Master Thesis
On the Performance of Sound Bubbles: An Analysis of Directional Performance and Underlying Acoustic Principles

Student: Emil Lunau Bentsen
Company responsibles: Jesper Jensen & Svend Feldt
Supervisor: Zheng-Hua Tan

Spring 2025

DEPARTMENT OF ELECTRONIC SYSTEMS
AALBORG UNIVERSITY

Abstract

This master thesis investigates the performance of the audio system proposed in the paper *Hearable Devices with Sound Bubbles*. The analysis focuses on three key objectives: (1) how the sharpness of the sound bubble boundary depends on the configured bubble size, (2) how system performance varies with the angle between the listener and the speaker, and (3) whether the neural network employed in the paper implicitly leverages the inverse distance law to estimate speaker distance.

The work includes a technical review of the original paper, a MATLAB-based simulation using the inverse distance law to model idealized bubble behaviour, and a systematic evaluation of the trained neural network using custom-generated test data. The custom data was created under simplified conditions to isolate distance and angle as variables, assuming near-anechoic environments and using single-speaker test cases.

The results suggest that the neural network does utilize cues consistent with the inverse distance law, as its performance patterns closely match those of the MATLAB simulation. However, due to simplifications in the test setupincluding low environmental variation, no reverberation, and idealized signal assumptions, the findings must be interpreted with caution. More realistic testing conditions are needed to assess the generalizability of the system to real-world deployments.

Contents

1 Introduction

1.1 Motivation

The rapid acceleration of processing power in electronics has made hearable devices today more accessible and capable of a lot of things which were previously limited to bulky computer systems, which were not practical to carry around. In open acoustics environments and public spaces like open-plan study-environments, libraries, museums or cafés, it can be difficult for people to filter out background noise or concentrate on the task at hand with different voices fighting for your attention [1][2]. This poses a challenge which can be attempted solved with better electronic devices, capable of things like separating interfering speakers from the ones you want to hear, amplifying speakers close to you or simply reducing background noise. With an increased use of wearable electronics, this could become easier to implement in the near future.

One way of approaching this problem, is to develop a system which can create a *sound bubble* [3] around the listener, which acts as a boundary beyond which voices and noise is filtered away or attenuated. This system could have the potential to improve working conditions at workplaces, help facilitate group work in educational institutions or even improve the quality-of-life for people wearing hearing aids. An illustration of the concept is shown in figure 1.

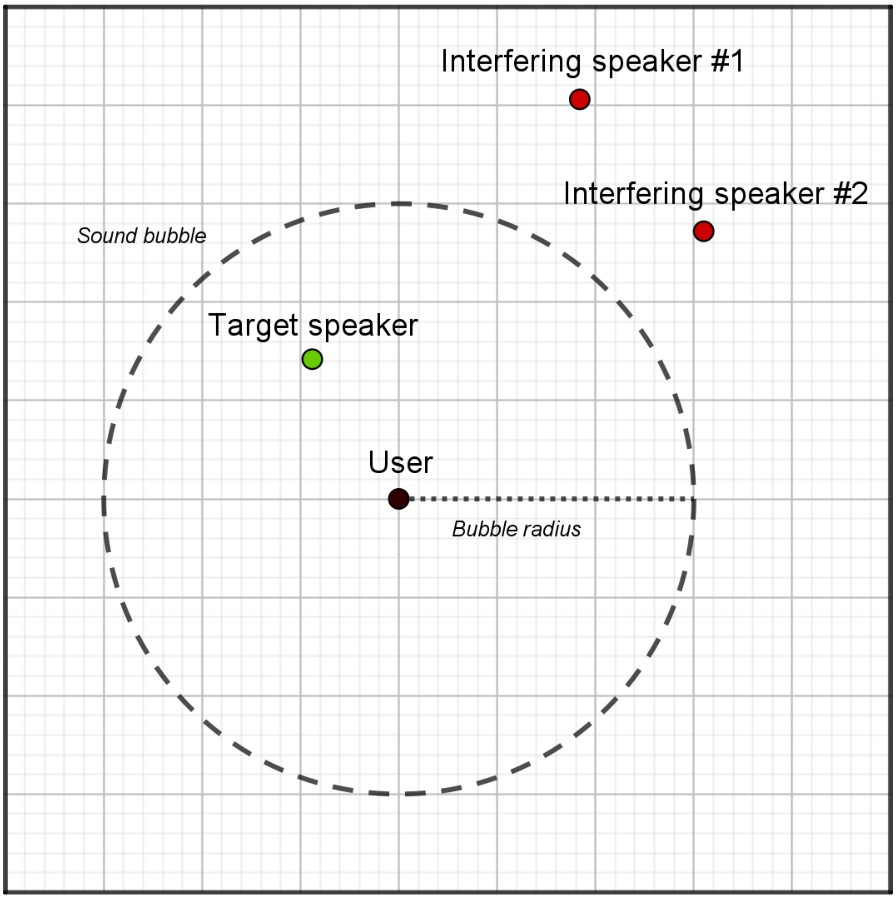


Figure 1: Illustration of a sound bubble, with a user (in the middle), a target speaker within the sound bubble (above to the left of the user) and two interfering speakers outside the sound bubble (top right).

The figure shows a target speaker inside the sound bubble and two interfering speakers outside the sound bubble. In this case the target speaker inside should **not** be attenuated and the two interfering speakers outside **should** be attenuated.

This exact solution has been attempted by a research group (Chen et. al.) at Washington University[3]. Their approach was to train a deep neural network (DNN) to separate detected speech signals, identify whether each speaker was inside or outside the sound bubble, and attenuate the speakers outside the bubble.
The test setup for this project was made by mounting a number of small microphones on the headband and ear cups of an off-the-shelf noise-cancelling headset. The noise-cancelling functionality of the headset was the kind seen in many consumer-grade headsets used today to suppress noise coming from outside the headset. The microphones mounted on the headset were used to capture the surrounding audio and feed it to a small processing unit which was used to run the DNN with the received audio. The resulting output was then played back through the speakers of the headset, while the noise-cancelling headphones suppressed incoming sounds to ensure the listener only received the filtered audio.

Depending on the performance of the system, the sound bubbles paper [3] proposes a system with a wide range of possibilities. To investigate whether the system is viable for use in real-life scenarios, the performance needs to better understood.
As it is in the nature of DNNs to have hidden layers, it is unknown exactly what information the DNN presented in the sound bubble paper uses to filter the audio, how it uses it and what implicit signal processing is going on behind the scenes. This project aims to test and evaluate the performance of the system presented in the sound bubbles paper[3] and investigate which acoustic principles are utilised by the DNN model, and how the performance of the model depends on the acoustic environment in which the model is tested.

1.2 Project Goal

The primary objective of this project is to evaluate the performance of the *sound bubbles* presented by Chen et. al. and to investigate which acoustic principles their DNN utilises. Specifically, the project will involve:

- Understanding and describing the architecture and operation of the provided sound bubble system [3].
- Developing a MATLAB simulation to model some of the theoretical acoustic limitations of a sound bubble.
- Designing and generating a custom dataset to systematically test the system's behaviour and performance.
- Analysing the directional performance of the model with the custom dataset and comparing it to the MATLAB simulation.
- Discussing the performance and limitations of the sound bubble system and possible improvements for future implementations.

The outcome of the project is intended to provide a better understanding of how the proposed sound bubbles work, and the directional performance.

1.3 Scope

This project uses the publicly available data and models provided by Chen et. al on GitHub. [4]. The research group has only made a model available which is trained on synthetically generated data. Thus the scope of the project will be limited to a theoretical and simulation-based evaluation. This means that all test samples and environments are simulated, and tested on a model trained with synthetic data, and all tests are performed without the physical test setup used in the original paper. This project aims to provide a clear understanding of certain aspects of the systems performance, which is why all simulations will be made under controlled assumptions to avoid interfering elements introduced by factors such as interfering speakers, noise and reflections. Thus the project scope is limited to:

- A DNN model trained with synthetic data.
- Synthetic microphone signals simulated in a digital room.
- Highly absorptive low-noise simulated environments.
- Single speaker scenarios without interfering speakers.
- Theoretical modelling of sound sound bubbles using MATLAB.

2 Background and Analysis

In this section the theoretical and technical foundation necessary to understand the project will be presented. It consists of three parts: first a description of some of the fundamental deep learning concepts, which will help understand the work done by Chen et al; second, a discussion of key acoustic principles which are used in the project's simulation and evaluation; and third, an exploration of some more project-specific aspects such as some theorised physical limitations of a sound bubble, and evaluation metrics used to assess speech quality and intelligibility in both the original paper and in this project.

2.1 Deep Learning and Neural Networks

Deep learning models are a special type of machine learning models, which can be taught to model complex behaviour, otherwise difficult to obtain with traditional methods. The models are called *deep neural networks* (DNNs) and are built with multiple layers of interconnected computational units, loosely inspired by the structure and functioning of biological neural networks.

Why deep learning is used over traditional signal processing

Traditional signal processing methods for spatial audio tasks often rely on explicitly defined models of sound propagation and require manual tuning of parameters. While they can be very effective, more complex environments with a lot of reverberation or noise might degrade the performance.

To overcome these limitations, DNNs can be used to model complex acoustic behaviour otherwise difficult to handle with traditional signal processing methods. The goal when creating a sound bubble is to only extract the audio sources located within the bubble. Traditional signal processing methods would require precise knowledge of source positions, noise profile, and microphone array geometry to separate and mute unwanted audio. In contrast, a trained DNN might be able to learn to associate patterns of phase and amplitude differences (such as interchannel level difference (ILD) and interchannel phase difference (IPD)) with speaker positions, without explicitly using commonly used acoustic principles or classical signal processing methods. This way it can learn to categorise speakers as inside or outside and attenuate the signal based on the categorisation.

To solve specific tasks, DNNs need to be trained and are well-suited to use structured data such as the multichannel microphone signals used for inputs in the sound bubble paper. By training a DNN with a wide variety of data it can be taught to be more flexible and better at adapting to variations caused by a change in position (eg. when the listener is moving), environmental noise, or array configurations. Through exposure to large and diverse training data, the model can learn to adapt to these variations, making it potentially more suitable for real world implementation.

DNN Structure

A typical DNN architecture consists of three main components: an *input layer*, one or more *hidden layers*, and an *output layer*. Each layer is composed of multiple units known as *neurons*. An example of a DNN architecture is shown in figure 2.

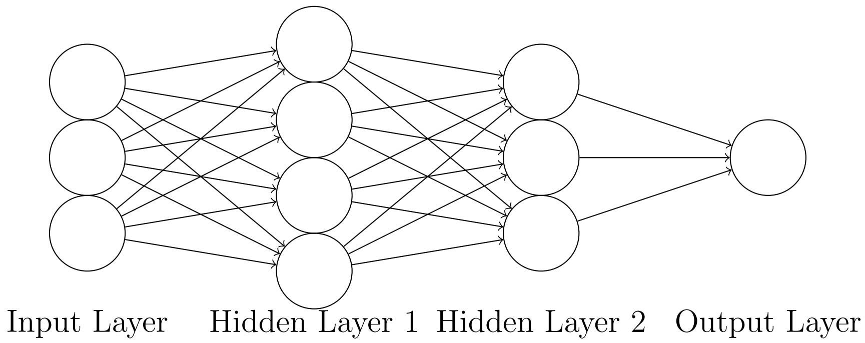
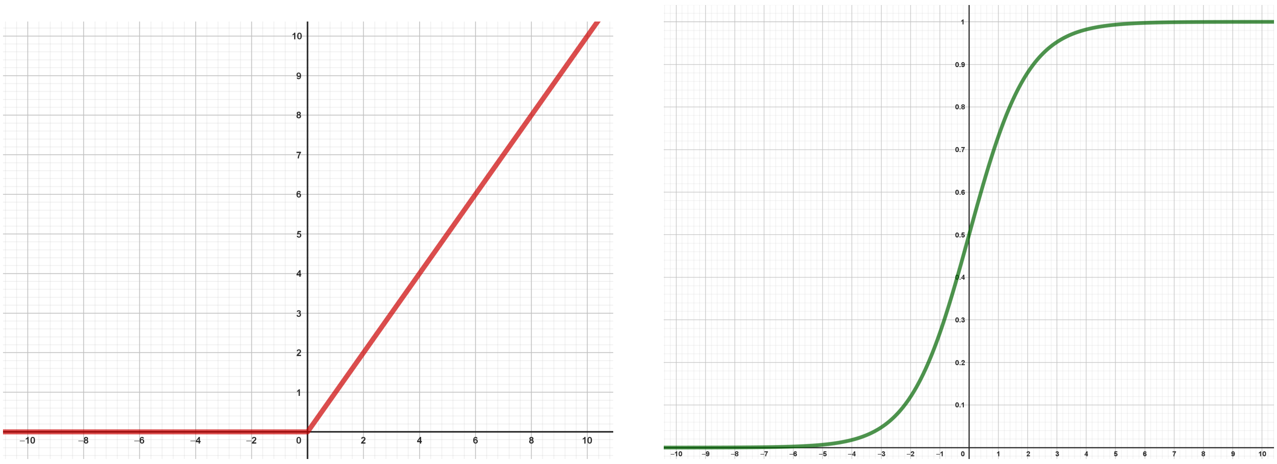


Figure 2: Structure of a fully connected deep neural network (DNN) with two hidden layers. Each node represents a neuron, and arrows indicate weighted connections.

Each layer in a DNN takes an input from the previous layer and transforms it into increasingly abstract representations, which allow the network to learn complex patterns and relationships [5]. Each neuron in a layer receives a vector of inputs from the previous layer and then computes a weighted sum of those inputs. As these weights can be both positive and negative so can the weighted sum. Then a non-linear function called an *activation function* is applied to the result. Two examples of activation functions can be seen in figure 3. The one in figure 3b is the sigmoid function, which squeezes the result into a range between 0 and 1, where more negative outputs will be closer to 0 and more positive outputs will be closer to 1. The other one in figure 3a is the Rectified Linear Unit, or *ReLU* function, which outputs 0 for all negative inputs, and a linearly increasing output for increasingly positive inputs.



(a) ReLU function.

(b) Sigmoid function.

Figure 3: Two of the most commenly used activation functions, ReLU (left) and Sigmoid (right).

Before the activation function is applied, a *bias term* can be added to the weighted sum. This can make sure to skew the result such that neurons are only activated if the weighted sum is large enough. Mathematically, the activation of a single neuron **a** in a layer l can be expressed as:

$$a^{(l)} = \phi(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases of layer l, ϕ is the activation function, and $\mathbf{a}^{(l-1)}$ are the activations from the previous layer.

Training a neural network

The values of the weights and biases in a neural network are very important as they determine the behaviour of the DNN and thus are the “knobs and dials” which can be adjusted to tune the model’s performance. Training a neural network is done by feeding the network with data and then using the output to adjust the weights and biases. As neural networks can become very large, the tuning is not done manually and is usually done using a *loss function*. The loss function is used to compute how bad the output of the network is compared to some desired output. The loss function can be designed by the person training the network to fit the performance of the network to specific use cases. After the loss function is calculated, a method called *back-propagation* is used to go back through the model’s layers via the output layer and adjust the weights and biases to minimised the loss function, and thus improving the performance of the network.

2.2 Inverse Distance Law

The inverse distance law is a fundamental principle in acoustics that describes how sound pressure decreases as a function of distance. If ideal conditions are assumed, such as no reflections, a fully absorptive environment and a sound coming from a point source, then the sound pressure is inversely proportional to the distance from the sound source. Mathematically this can be expressed as:

$$p(r) \propto \frac{1}{r}$$

With p being the sound pressure and r being the distance from the source.
Sound Pressure Level (SPL) in decibels is defined as:

$$L_p = 20\log_{10}\left(\frac{p}{p_0}\right)$$

With p_0 being the reference pressure (the lowest pressure detectable by humans) and p being the measured pressure. According to the inverse distance law, if the pressure at a certain distance from a sound source is p, then if the distance is doubled (i.e., $r \to 2r$) then the pressure becomes $p/2$. This has the following effect on the SPL:

$$\Delta L_p = 20\log_{10}\left(\frac{p/2}{p}\right) = 20\log_{10}\left(\frac{1}{2}\right) \approx -6.02\,\mathrm{dB}$$

Thus, each doubling of the distance to the source results in approximately a 6 dB decrease in SPL.
It is important to note that the inverse distance law assumes a point source and no loss due to the environment. In real environments, absorption by the air and the room’s surfaces, as well as reflections, will cause it to deviate from this ideal behaviour. To minimize such effects, this project simulates the sound field in an almost anechoic virtual room with high absorption coefficients, thereby closely approximating the ideal conditions of the inverse distance model.

2.3 Distance Estimation

Two concepts which are important to distinguish are absolute sound pressure versus relative amplitude difference. These are concepts are used throughout this project as they can provide information about sound sources in an acoustic environment.

Absolute Sound Pressure

Absolute sound pressure refers to the magnitude of a sound signal or the sound pressure at a specific location. It is typically either measured in pascals or expressed in decibels (dB SPL) and indicates how “loud” the sound is at that point. In a single-microphone system, absolute pressure is often the only available metric. As sound pressure decays by distance, the inverse distance law described above along with the absolute sound pressure can be used to calculate how far away a sound source is, if it is already known how loud the sound was at its source (typically defined as 1m away from the source). For a sound bubble, the goal is to estimate the distance to an unknown source and to asses whether or not it is inside or outside the sound bubble. It is also the ambition of the research group behind the sound bubbles paper (as mentioned in pg. 1 of [3]) that the sound bubble *“must work when the speakers outside the bubble are louder and there-fore cannot use only the amplitude information”* [3], pg. 1047.

Relative Amplitude Difference

When using more than one microphone, instead of measuring the absolute pressure, the relative amplitude can be used. When measuring the difference in amplitude between two microphones, due to the inverse distance law, it is possible to extract information from that amplitude difference. For example if you have two microphones and a sound source is placed a distance away along the line between the two microphones, then audio from the sound source will arrive with a delay between the two microphones. For this scenario the following equations can be set up:

$$\begin{aligned} r_1 &= r - \frac{d}{2} & \qquad r_2 &= r + \frac{d}{2} \\ p_1 &= \frac{p_{source}}{r_1} & \qquad p_2 &= \frac{p_{source}}{r_2} \end{aligned}$$

Here r_1 and r_2 are the distances to the two microphones, r is the distance from the speaker to the midpoint between the microphones and d is the distance between them. As per the inverse distance law, the pressure is inversely proportional to the distance, and the sound pressure at each microphone can be written as in the equations above. Combining the equations for pressure and distance, results in an equation which can be used to estimate the speaker distance by only using the distance between the microphones and the pressure measured by them.

$$\frac{p_1}{p_2} = \frac{\frac{p_{source}}{r_1}}{\frac{p_{source}}{r_2}} = \frac{r_2}{r_1} = \frac{r + \frac{d}{2}}{r - \frac{d}{2}} \quad \Rightarrow \quad r_{est} = \frac{d(p_1 + p_2)}{2(p_1 - p_2)}$$

The distance estimator can be used to estimate the distance r_{est} to a source by knowing the sound pressure at two different microphones with a known distance between them. This assumes that there is only one sound source and that it is placed along the line going through the two microphones.

If, however, the source was to be placed at an angle to the line going through the microphones, then the difference in distance and thus the time delay between the microphones would change. The difference in distance from the microphones to the source can be written as $\Delta l = r_2 - r_1 = d \cdot cos(\theta)$. An illustration of this can be seen in figure 4.

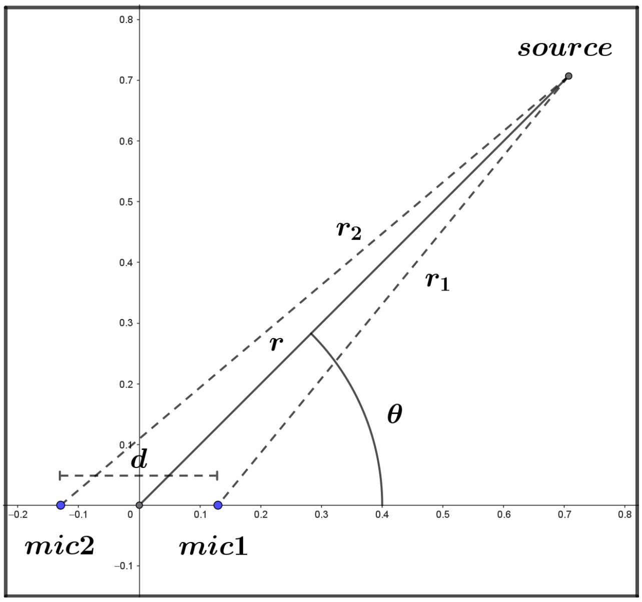


Figure 4: Illustration of a source placed at an angle to the microphone axis. Due to the angle, the distance difference between the microphones change ($\Delta l = r_2 - r_1 = d \cdot \cos(\theta)$)

If the sound source is placed at 90°directly in front of the midpoint between the microphones, then both microphones are an equal distance from the source ($\Delta l = d \cdot (90) = 0$) and the sound would reach each microphone at the same time. This would cause the r_{est} formula above to longer be applicable. This suggests, that the performance of this estimator decreases as the angle approaches 90°.

2.4 Reflections and Absorption

In real acoustic environments, sound does not propagate in a purely direct path from source to receiver. In a room sound will interact with surfaces which will cause absorption, reflection, diffusion, and scattering. These effects can play a significant role in how the sound is perceived by the listener.

Reflections.

When sound waves encounter hard surfaces such as walls, floors, or ceilings they are partially reflected. These reflected waves can arrive at the microphone array with varying delays and at different angles, resulting in constructive or destructive interference. This interferences can alter the perceived direction or loudness of a source.

Absorption.

Absorption occurs when part of the sound energy is absorbed by the surface the sound hits. Soft materials like curtains, carpets, and acoustic foam tend to absorb more sound, especially at high frequencies. The degree of absorption is often described by an absorption coefficient between 0 (fully reflective) and 1 (fully absorptive).

Impact on Sound Bubble

For a system such as the one proposed in the sound bubble paper [3], a high number of reflections and poor absorption can have a big impact on the performance of the model.

Reflections might reach the listener from unintended directions, making it harder for the system to estimate the distance to the source. In contrast, an anechoic or highly absorptive environment helps preserve direct-path audio, making sure only the intended sound reaches the listener.

In this project, simulations assume highly absorptive, non-reverberant environments to isolate the directional behaviour of the system. While this is useful for understanding the fundamental performance, testing the system with more realistic room conditions are a requirement before the system can be deployed under real world conditions.

2.5 Limitations to Bubble size

If the sound bubble in the reference paper is based upon the inverse distance law, and the principles described in this chapter then that would put some limitations to the possible performance.

The sound pressure level (SPL) of normal speech is usually around 60dB measured 1 meter from the speaker[6]. At a reference air pressure of $p_0 \approx 20\mu Pa$ in air at 20°, this is equivalent to a sound pressure in Pa of:

$$L_p = 20\log_{10}\left(\frac{p_{speak}}{p_0}\right) \quad \Rightarrow \quad p_{speak} = p_0 \cdot 10^{\frac{60dB}{20}} = 20mPa$$

Then if two microphones are placed 25.9 centimetres apart (the furthest distance between two microphones in the sound bubbles paper), with an audio source at 60dB at 1m, then the difference in sound pressure between the microphones at different distances would be:

$$\begin{aligned}
1m :\Delta p_1 &= p_1 - p_2 \approx 22.98mPa - 17.71mPa &&\approx 5.27mPa\\
2m :\Delta p_2 &= p_1 - p_2 \approx 10.69mPa - 9.39mPa &&\approx 1.30mPa\\
4m :\Delta p_4 &= p_1 - p_2 \approx 5.17mPa - 4.84mPa &&\approx 0.32mPa\\
8m :\Delta p_8 &= p_1 - p_2 \approx 2.54mPa - 2.46mPa &&\approx 0.08mPa
\end{aligned}$$

From the equations above it is clear that the requirements for the signal to noise ratio (SNR) of the microphone setup used increases a lot as the bubble size is increased. Thus the precision of the microphone setup and the amount of noise will likely be a limiting factor to the performance of the system.

The *ReSpeaker six-channel microphone* used in the sound bubble paper has an SNR of 59dB [7]. This equates to microphone noise of: $94dB - 59dB = 35dB$ which is equivalent to a sound pressure of $p_{micnoise} = p_0 \cdot 10^{\frac{35}{20}} = 2mPa$. This means that this microphone would only just be able to detect the pressure difference at 2m in theory. Practically you would need the noise to be a lot lower than the signal to get more accurate measurements. This could be achieved by denoising algorithms or by using microphones with a better SNR.

In the sound bubbles paper, they have both 2-, 4- and 6-microphone setups, but the results presented in the paper are mostly focused on the 6-microphone setup. Increasing the number of microphones from 2 to 6 would likely make the system able to detect much lower differences, as the amplitude difference between multiple microphone pairs could be compared to each other to eliminate noise.

2.6 Speech Evaluation Metrics

To evaluate the performance of the sound bubble system and its neural network, four metrics are used: signal-to-noise ratio (SNR), scale-invariant signal-to-distortion ratio (SI-SDR), short-time objective intelligibility (STOI), and perceptual evaluation of speech quality (PESQ). These metrics assess the perceived quality and intelligibility of the estimate signal output from the model

2.6.1 SNR (Signal-to-noise ratio)

SNR is the signal-to-noise ratio, and it tells how the power of an original signal relates to the power of the difference between the original signal and an estimate. This difference is often referred to as noise. For SNR higher usually means better as a higher SNR means a stronger signal compared to the noise. SNR as a speech evaluation metric can used to determine how much noise is present in an processed speech signal compared to a clean reference signal. SNR is measured in decibels (dB) and is calculated in the following way:

$$SNR_{dB} = 10 \cdot \log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 10 \cdot \log_{10}\left(\frac{\|s(t)\|^2}{\|s(t) - \hat{s}(t)\|^2}\right)$$

Here P_{signal} and P_noise represent the power of the signals, $s(t)$ represents the original "clean" reference signal and $\hat{s}(t)$ represents the estimate. In the sound bubble paper, as well as in this project, SNR is used to evaluate the output sound bubble model. This means that $s(t)$ is the synthetically generated audio signal and $\hat{s}(t)$ is the output estimate of the model. The SNR can be used to tell how much the model has attenuated the reference signal. If the model has not attenuated the signal, then the SNR will likely be high, as the only difference should be the noise introduced by the model. If the model DO attenuate the signal, then there will be a large difference between the reference and the estimate, and thus the SNR should become lower the more attenuated the signal is.

2.6.2 SI-SDR (Scale-invariant signal-to-distortion ratio)

SI-SDR[8] is scale invariant signal-to-distortion ratio. Regular SDR is used to measure the relationship between a signal and the distortion present, much like SNR. The scale invariance (SI) means that the SDR is unaffected by the amplitude differences of the two signals. This way only the "shape" of the signal is being evaluated and not any gain or attenuation. This is the primary difference from the SNR. SI-SDR is calculated as seen below:

$$\text{SI-SDR} = 10 \cdot \log_{10}\left(\frac{\|\alpha \cdot s\|^2}{\|\alpha \cdot s - \hat{s}\|^2}\right), \quad \alpha = \frac{\langle \hat{s}, s\rangle}{\|s\|^2}$$

As with SNR, SI-SDR is measured in decibels (dB) and higher is better. A high SI-SDR means that the distortion is low compared to the signal. The difference is the scaling factor α. This factor scales the reference signal such that it matches the amplitude of the estimate. Due to the scale-invariance SI-SDR cannot be used to measure how much the model attenuates each sample, but can instead be used to measure how much noise the model introduces and how much it distorts the sample.

2.6.3 PESQ (Perceptual Evaluation of Speech Quality)

PESQ[9] stands for Perceptual Evaluation of Sound Quality and is an objective speech quality assessment model. It is developed to estimate the *Mean Opinion Score (MOS)* which is a subjective rating of perceived quality of speech signals, given by human listeners on a scale of

0 (bad) to 5 (excellent). MOS-testing is time-consuming and resource-intensive and thus PESQ was developed as an objective approximation of MOS. In the approximation PESQ takes into account parameters such as audio sharpness, call volume, background noise, variable latency, audio lag, clipping, and interference
The resulting PESQ score usually ranges between 1.0 and 4.5, with higher values indicating better quality.

2.6.4 STOI (Short-Time Objective Intelligibility)

STOI[10] is a objective metric which estimates how intelligible speech will be perceived by a typical listener. STOI works by splitting the reference signal and the degraded or noisy signal into time- and frequency-bands. These are then compared with and evaluated across all bands. The final result of the comparison is a STOI score between -1 and 1, where a higher score means the distorted signal is easier to understand.

2.7 Motivation for Metric Selection

Each of the chosen metrics targets a different aspect of audio quality:

- **SNR** measures basic signal fidelity and, in this project, is used to evaluate if the model has attenuated the signal or not.
- **SI-SDR** provides a more robust, improvement over SNR in terms of signal fidelity, as it is unaffected by scaling differences. it is used to evaluate how mich noise and distorstion the model introduces.
- **STOI** estimates intelligibility, and is used to approximate how intelligible the model estimate will be perceived by a regular human listener. It is used because of the high correlation with human perception.
- **PESQ** approximates subjective human assessment of audio quality. It is used to relate the performance to how the it would be perceived in a real world scenario.

The combination of these metrics is chosen, as it combines basic comparison techniques (SNR and SI-SDR) with more advanced perception aimed evaluation metrics. These metrics ensures that both physical signal properties (SNR/SI-SDR) and perceptual outcomes (STOI/PESQ) are evaluated, offering a more complete picture of the systems performance.

3 The Sound Bubble Paper

Hearable Devices with Sound Bubbles [3] is a paper seeking to create a bubble around a given listener inside which all sounds will be let through to the listener, and all sounds outside will be attenuated or ideally cut off completely. The approach used in the paper is using a deep neural network to filter the sound by separating the different speech signals present in the measured sound, and then estimating the distance and filtering the sound based on the bubble size.
Specifically, the research group (Chen et. al.) sets out the create programmable sound bubbles using real-time neural networks running on embedded CPUs mounted on headsets. This process involved constructing a custom test setup, modifying existing neural networks for speech separation and generating data (both synthetic and real world) for training, testing and evaluation.

In the paper the goal is defined as meeting four key requirements:

- "process incoming audio in real time and on the device on an embedded CPU, and achieve an end-to-end latency of less than 2030ms between the audio and visual scenes"
- "adapt to diverse reverberant conditions and wearers"
- "support configurable bubble sizes"
- "work when the farther speakers are louder than those within the bubble, as well as when there are multiple or no speakers inside or outside the bubble."

The paper focuses a lot on the performance of the system with diverse conditions. From varying testing conditions such as reverberation and room size, to number of speakers, speaker placement and adding background noise and interfering speakers.
An important aspect of the paper is, that it requires the system to work when farther speakers are louder than those within the bubble. This makes the system more complex, and makes sure that more than just the absolute sound pressure of an audio source is utilised in the creation of the sound bubbles.

Multiple different test setups were used in the paper. The primary setup was a pair of headphones with six microphones mounted on it along with an embedded CPU to hold the neural network and process the audio. The paper includes multiple other test setups, but as most of the results in the paper primarily focuses on the 6-microphone setup mounted on a pair of headphones, that will also be the focus of this project.

3.1 Network architecture

The core of the sound bubble paper is the real-time neural network. It consists of four different modules which together create a model which can create the proposed sound bubbles. The model works by taking the incoming audio from the array of microphones and passing them through the model. The output of the model is the processed audio, which will be what the user hears. The model's four modules are a feature encoder, a distance-embedding network, a separation module and a feature decoder, and will be described in this chapter.

Feature Encoder

The feature encoder works by sampling the multi-channel audio input (a channel for each microphone) in 8ms audio chunks at a time with a 4ms lookahead. This is done at 24kHz.

This audio chunk is then converted to a time-frequency (TF) representation with a short-time Fourier transform (STFT). Then the interchannel phase difference (IPD) and interchannel level difference (ILD) features are extracted and combined with the TF-representation. These provide information about the phase- and level-differences between the channels.

Separation Module

After the feature encoder, the signal is fed to the separation module. This is the most computational heavy part of the model. The separation module processes the input signal and separates the individual speakers present. It is conditioned through training and the distance embedding mask to only preserve and let through the speakers located inside the bubble. Sources coming from outside the bubble are implicitly suppressed as part of the learned separation behaviour.

The separation module is based on the state-of-the-art speech separation neural network TF-GridNet. However, the TF-GridNet network cannot run on small embedded CPUs (like a Raspberry Pi). For this reason, Chen et. al. took multiple actions to reduce the computational complexity of the network like removing computationally heavy parts of the network and compressing the signal.

Distance-Embedding Network

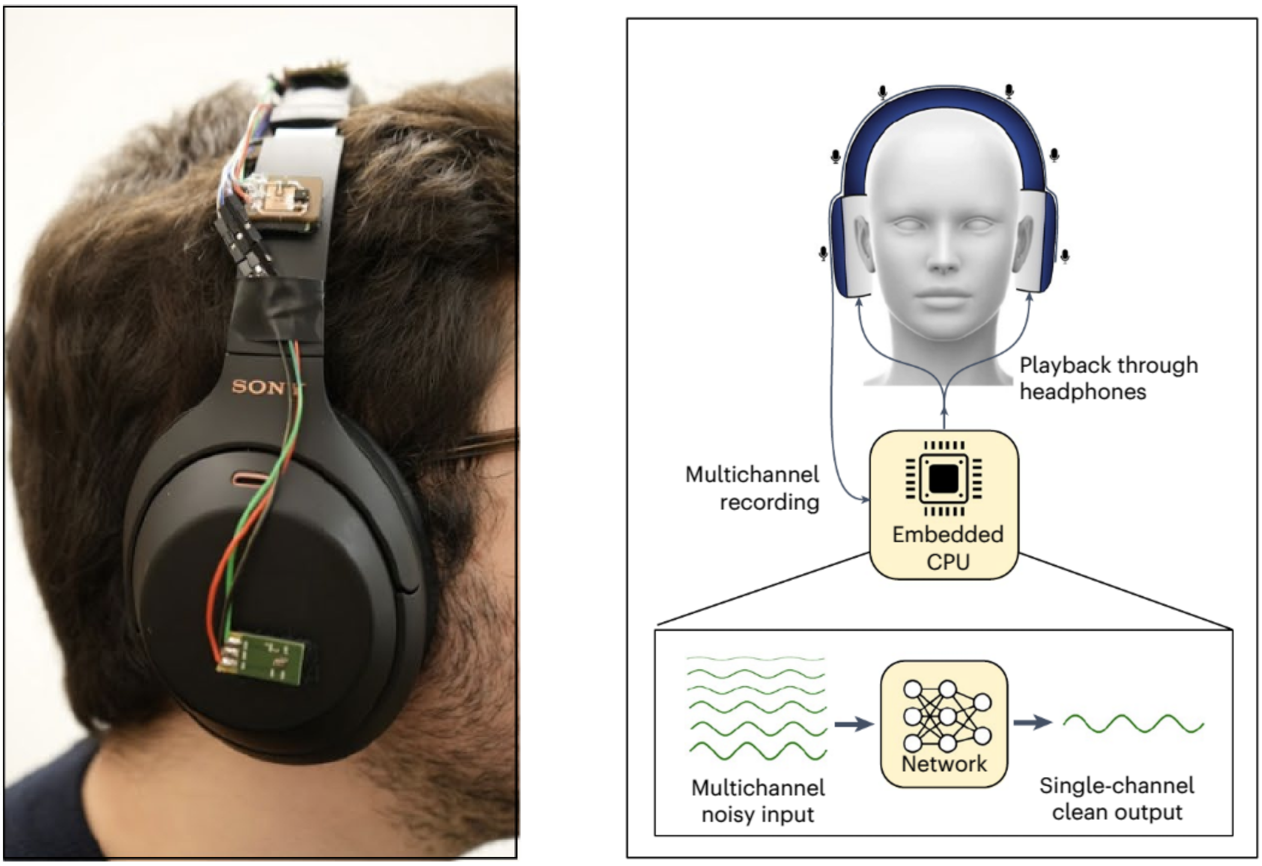
The system is designed to work with three configurable bubble radii of 1.0 m, 1.5 m and 2.0 m. These are coded into a distance-embedding mask which is applied inside the separation module.

Feature Decoder

The feature decoder takes the TF-representations from the separation module (only containing the speakers inside the bubble) and converts them back into the time domain with a STFT decoder (an inverse STFT). This time domain audio signal is the output of the model.

3.2 Test Setup

The test setup primarily consisted of a pair of headphones (Sony WH-1000XM4) with regular noise-cancelling, with an array of six microphones (ReSpeaker six-channel microphone [7]) mounted on it. The array of microphones were connected to an embedded CPU (Raspberry Pi 4B) which held the model. After receiving and processing the audio, the processing unit played back the processed audio output from the model through the speakers of the headphones. The noise-cancelling of the headphones was used during testing to suppress all sounds to make sure that only the processed audio reached the user's ears. In figure 5 a picture and an illustration of the test setup from the paper can be seen.



(a) Picture of the microphones mounted on a pair of headphones.

(b) Illustration of the physical test setup and how the parts are connected to each other.

Figure 5: Figures from the Sound Bubble Paper of the physical test setup with headphones. In the paper subfigure a (left) is figure 1b and subfigure b (right) is 1c [3].

3.3 Data generation

Data generation was a crucial part of the sound bubbles paper. The research group both generated synthetic data and collected real world data for the paper, with a wide combination of acoustic environments, number of speakers, speaker angles and noise conditions. After generating the data, it was split into non-overlapping training-, validation- and testing subsets. The validation and training of the model is not within this project scope, and as such the descriptions in this chapter will focus on the generation of test data.

Synthetic Data Generation

The synthetic data generation for the sound bubbles paper generates microphone audio data by simulating acoustic environments and calculating the expected output at the microphone array. The data generation script `generate_adaptive_data.py` generates 6-channel audio data by using the python library “Pyroomacoustics” (PRA). PRA is used to generate a virtual room with user defined acoustics. Speakers and microphones can then be placed inside the room and the acoustic environment can be simulated to estimate the resulting signal received by each microphone. The full script can be found in the paper’s GitHub [4].

To generate the audio in each test sample they used three open-source datasets. For speech utterances they used the LibriTTS[11] and VCTK[12] datasets. For background noise they used the WHAM![13] dataset. Each sample was 5 seconds long and consisted of a combination of these three datasets. All three datasets were split into non-overlapping subsets for training, validation and testing.

Room Generation

When data is generated for the sound bubbles paper, it is simulated in a virtual room with random parameters. This means that both the size, absorption and number of reflections can vary widely from sample to sample. For each sample, the bubble size was defined as either 1.0, 1.5 or 2.0 and PRA was used to generate a random room with dimensions ranging from 5x4x2 to 8x8x4 m^3. The absorption of the room was uniformly sampled from 0.1 to 0.9 and the maximum number of reflections off of surfaces was also sampled uniformly between 10 and 72.

When training and testing a model like the one used in the sound bubbles paper, it is important that the generated data has a lot of variety. This way the model can be trained to learn to better generalise to different scenarios and acoustic environments.

Speaker- and microphone placement

After generating the room, a 3D 6-microphone array with the same measurements as the physical setup (figure 5a) was then randomly placed somewhere in the room at a random height between 1.2m and 1.8m. Then 0-2 target speakers were placed at random positions inside the bubble radius and 1-2 interfering speakers were placed outside the bubble with a 50% chance of adding background noise from the WHAM! dataset. All speakers got assigned a random 5s speech signal from the VCTK or LibriTTS datasets. The height of these speakers was also randomly sampled at $\pm 0.4m$ from the height of the microphone array. In 50% of the samples the interfering speakers (outside the bubble) were louder than the targeted speakers (inside the bubble).

The placement of the speakers and the microphone array is important when testing the model. As mentioned under *Room Generation* above, the room varies in absorption and number of reflections. This can have very different consequences, as the reflections from walls, floor and ceiling can introduce delays and disturbing elements to the audio signals, altering the perceived distance or loudness of the speaker.

Simulation

After the complete test scenario was set up, the room was simulated to generate the resulting audio at each microphone. This process was repeated for each sample.

Real world data generation

To record the real world data, a robotic platform was made with a mannequin head wearing the headphones. The robotic platform could turn, raise and lower the head to record audio in different conditions. Audio was recorded at more than 60 different head orientations and three speaker heights. For a given speaker distance, all angle and height combinations were tested, before the speaker was moved to a new distance. This was done for four different acoustic environments with 8-11 distances per environment. The variation of each sample can help train the model to generalise better to realistic environments.

This project focuses on synthetically generated test data. For this reason the rest of the paper will assume synthetic data generation.

3.4 Results Presented in Sound Bubbles Paper

This section will present the performance and conclusions of the sound bubbles paper, based on their synthetic data, as the model was trained with synthetic and real world data separately. The performance of the model was evaluated in multiple different ways. A selection of them is presented in figure 6.

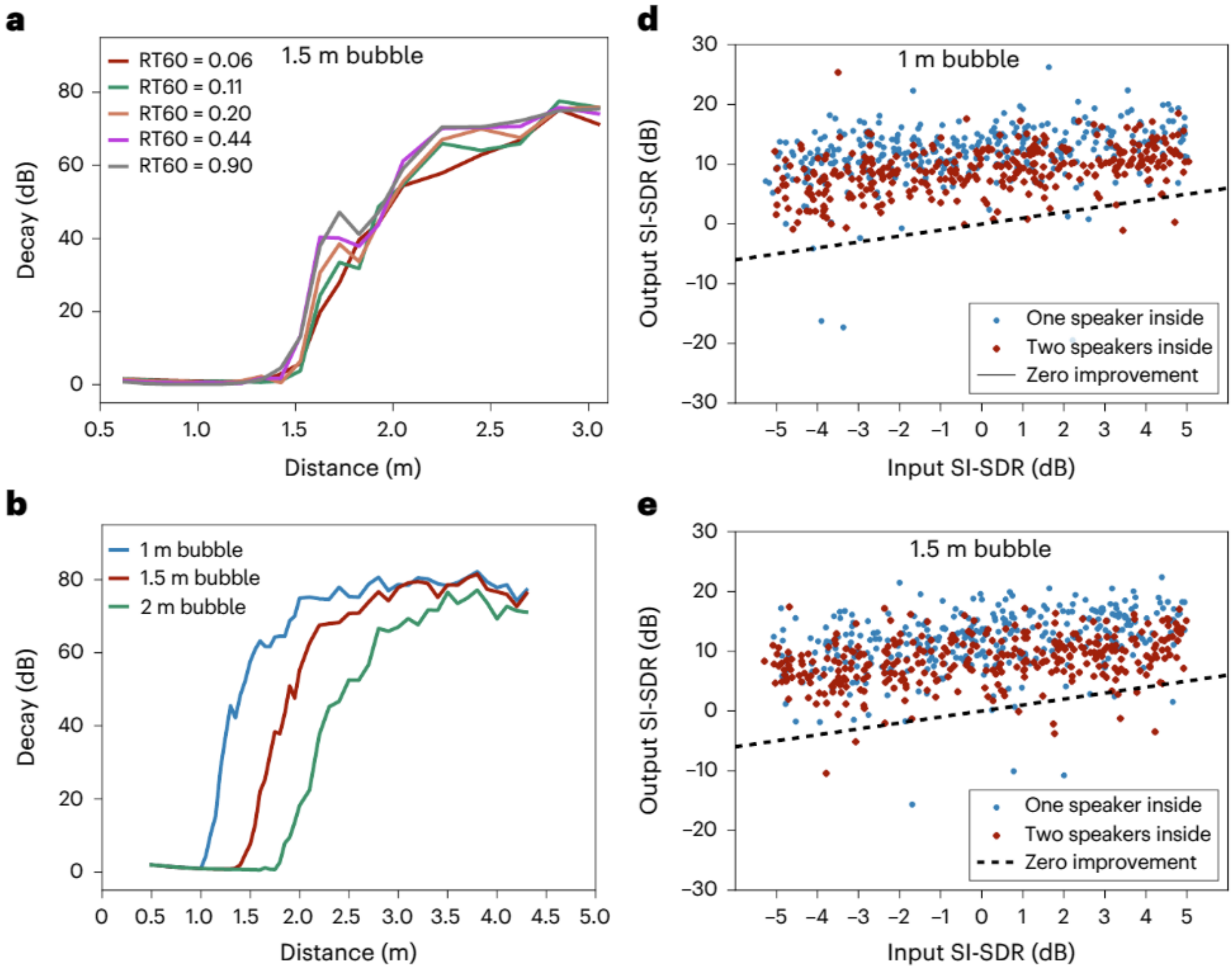


Figure 6: Figure 3a, 3b, 3d and 3e from the sound bubbles paper [3]. a and b show the decay as a function of distance for different RT60 values with a fixed bubble size (a) and different bubble sizes with fixed RT60 (b). d and e show the improvement in SI-SDR of the output of the model.

Figure 6a shows the suppression of speakers as a function of distance for a variety of reverberant conditions in a 1.5m bubble. The reverberation is shown as RT60 values, which is the time in seconds it takes for a sound to decay 60dB. This shows a fairly uniform decay between the different RT60 values after 1.5m, with the lowest RT60 values resulting in the best performance (less reflections and higher absorption lead to less noise and better suppression). At 2.0m all the signals have decayed around 50dB.

Figure 6b shows the decay as a function of distance for all three bubble size configurations. To generate the graph, a room with measurements of 8x8x3 m^3 and a fixed RT60 of 0.11s is generated. Then a single speaker is placed at increasing distances, computing the average decay of 30 samples per distance. The figure shows that for the bubble size of 1.0m, the signal starts decaying very close to the 1.0m bubble threshold. For the 1.5m and 2.0m bubbles, the signal starts decaying slightly before the speaker reaches the threshold, more so for 2.0m than 1.5m. The sharpness of the decay also seem to vary between the bubble sizes, with 1.0m being having the sharpest decay, and 2.0m having the slowest decay.

Figure 6d and 6e show the relation between output SI-SDR and input SI-SDR for the 1.0m and 1.5m bubbles. Plotted are both blue dots, representing when one speaker is inside the bubble, and red dots, representing two speakers. The x-axis shows the SI-SDR between the inside and outside speakers, meaning negative values represents when the interfering speakers (outside) are louder than the target speakers (inside). This shows that around 97.8%[3] of the samples showed an improved SI-SDR when passed through the model. With a single speaker inside the bubble, the model achieved SI-SDR improvements (SI-SDRi) of 12.35dB, 11.52dB and 9.85dB for bubble sizes of 1.0m, 1.5m and 2.0m respectively, and with two speakers inside the bubble, the network achieved SI-SDRi values of 9.02dB, 8.55dB and 6.94dB.

Conclusions

There are several conclusions drawn in the paper. One of the things that went well is the fact that they created a model which seems to create sound bubbles with configurable bubble radii. It seems like the transition between inside and outside the bubble becomes less sharp as the bubble size increases. This could impose an upper practical limit for bubble sizes. Another thing is that the model runs on an embedded CPU and does so with an average inference time of 6.36ms, which is well below the delay requirements defined at the start of the paper (20-30ms).

Some of the limitations mentioned are the prototype test setup and test environments. Relying on off-the-shelf noise-cancelling headphones might introduce unwanted noise, or not suppress the outside sounds completely. The data collection and simulation focused heavily on indoor environments. To ensure that the model works in different kinds of environments, a focus on outside simulations or data collection could be prioritised for future applications.

4 Project Objectives

Based on the 'Background and Analysis' and description of the Sound Bubble Paper in chapters 2 and 3 the primary goals of the project needs to be defined.

The primary goal of this project is to evaluate the performance of the model proposed in the sound bubbles paper [3], and investigate which acoustic principles are behind the performance of the neural network. As discussed in chapter 2, the inverse distance law can be utilised to estimate the distance to a sound source. This raises the question of whether the sound bubble model depends on the inverse distance law.
From this, the three primary objectives of this project can be defined.

1. Investigate how the sharpness of the sound bubble depends on the bubble size.
2. Investigate how the angle between the speaker and listener affects the performance.
3. Discuss whether the neural network in the sound bubble paper's model utilises the inverse distance law.

The following chapters will attempt to complete these three objectives through simulation, discussuion and model testing with custom generated data.

5 MATLAB Simulation

To investigate which acoustic principles the DNN model presented in the paper [3] utilises, an acoustic simulation of sound bubbles is made in MATLAB. The simulation is based on the inverse distance law described in chapter 2.2, and evaluates the performance of the distance estimator r_{est} from chapter 2.3 for a two microphone system. This is done by simulating a speech signal at various speaker angles and distances, and estimating whether a given bubble size would suppress the signal or not.

5.1 Simulation Setup

The MATLAB simulation is divided into two different simulations. One where only the distance to the microphone is varied, and another where both the distance and angle is varied. The simulation setup and method for both simulations are the same. The overall sequence of the simulation is as shown below.

1. Define parameters
2. For each bubble size:
    (a) For each speaker distance and angle
        i. Do 1000 times
            A. Simulate samples with given distance and angle
            B. Estimate distance
            C. Check if distance is inside bubble
        ii. calculate the let_through_percentage

The simulated setup has two microphones with a distance between them of d = 25.9cm (the same as the two microphones at the bottom of the microphone array in the sound bubbles paper) amd the microphones have an SNR of 59 (the same as the microphones used in the sound bubbles paper[7]).
The "speaker" is simulated as a sine wave at 500Hz. (The first formant (F1) of typical speech for vowel sounds lies at 300Hz - 900Hz with an average of ≈ 500Hz. This makes 500Hz a key frequency for identifying vowel sounds [14]). A normal conversation lies around 60dB [6]. The loudness of the simulated speaker is defined as a random value sampled uniformly between 55dB and 65dB to add variation. A code snippet of this can be seen below.

```
% Random speaker loudness between 55  65 dB
speaker_dB = 55 + rand() * 10;
% RMS sound pressure at 1m from speaker
p_speaker = p0 * 10^(speaker_dB / 20);
% Peak pressure for sine wave
p_peak = p_speaker * sqrt(2);
% Sine wave source
signal = p_peak * sin(2*pi*f*t);
```

Here the speaker loudness is used to calculate the peak amplitude of the sine wave. The sine wave is sampled at each microphone with a 24kHz sample rate (the same sample rate as in the paper), over a window of 8ms. This window size is chosen to resemble the inference time of the model in the paper (chapter 3.4). The inverse distance law (chapter 2.2) is then used to calculate the resulting signal amplitude at each microphone:

```
% Apply inverse distance law to get pressure at distance for
   each mic
signal_1 = signal / r1;     % sampled sound pressure at mic 1
signal_2 = signal / r2;     % sampled sound pressure at mic 2
```

The signal at this point is clean and without any noise, so to simulate a more realistic scenario, normally distributed microphone noise is added. The SNR of the microphones used was 59dB, so the microphone noise is $94dB - 59dB = 35dB$. This is converted to a resulting sound pressure in pascal: $mic\_noise = p_0 \cdot 10^{\left(\frac{35dB}{20}\right)} \approx 1.12mPa$. This microphone noise is then multiplied with a random number according to the normal distribution, and is added to each sample of the signal.

The RMS value of both noisy microphone signals is then calculated (over 8ms) to get a single value. This also averages out the added noise, reducing its impact on the signal. The RMS values are then used to estimate the distance with the distance estimator r_{est} found in chapter 2.3. This estimate is then compared to the bubble size.

```
% Estimate distance to speaker
r_est = d*(s1_meas+s2_meas)/(2*(s1_meas-s2_meas));

% Accept sound if it appears to come from within the bubble
if r_est <= bubble_radius(k)% && r_est > (d/2)
    pass_count = pass_count + 1;
end
```

If the estimate is within the bubble radius, then the variable pass_count is incremented. After each angle-distance combination is simulated a 1000 times, the percentage of samples let through (estimate inside bubble threshold) is calculated. This let_through_percentage is plotted as a function of either speaker distance or speaker angle for in the two simulations.

5.2 Varying Speaker Distance

With the approach described in chapter 5.1, the simulation of varying speaker distances was made. All samples are placed along the line going through the two microphones, with no change in angle (all 0°). Five different bubble sizes were simulated: 1.0m, 1.5m, 2.0m, 3.5m and 5.0m. 3.5 and 5.0 were included to see the impact of increasing the bubble size outside of the three sizes defined in the paper. The distance was sampled from 0.5m to 10.0m with a 0.05m step, and each distance was simulated 1.000 times. For each bubble size the let_through_percentage was plotted as a function of the true distance to the speaker. The results for the three bubble sizes used in the paper, 1.0m, 1.5m and 2.0m, are shown in figures 7, 8 and 9.

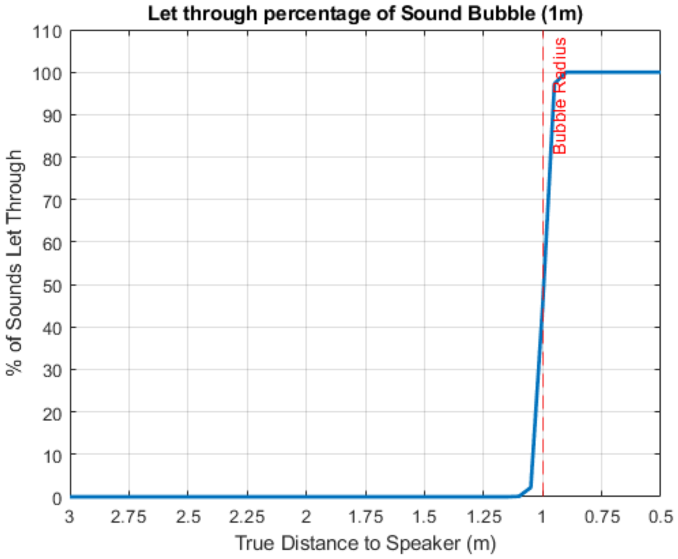


Figure 7: Performance of MATLAB simulated let-through-percentage as a function of true speaker distance (bubble size of 1.0m).

For all three bubble sizes it seems like the let-through-percentage is very close to 50% at the bubble threshold. The drop in `let_through_percentage` starts at 0.9m for the 1.0m bubble, 1.3m for the 1.5m bubble and 1.6m for the 2.0m bubble. A common way to describe the fall-time of a signal is to measure how long it takes for the signal to drop from 90% to 10%. In this case the fall occurs over distance instead. For the three first bubble sizes the fall-distance is $< 0.1m$, $0.2m$ and $\approx 0.5m$ respectively. This is similar to the results seen in the sound bubbles paper, where the sharpness of the bubble also seemed to lessen as the bubble size got bigger.

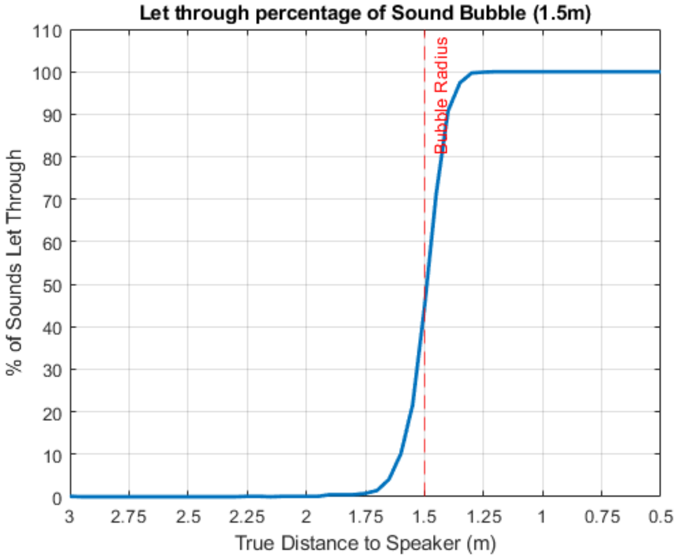


Figure 8: Performance of MATLAB simulated let-through-percentage as a function of true speaker distance (bubble size of 1.5m).

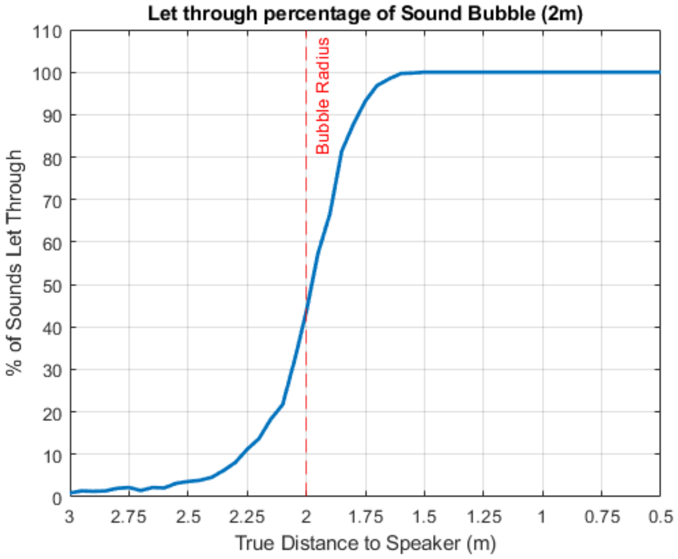
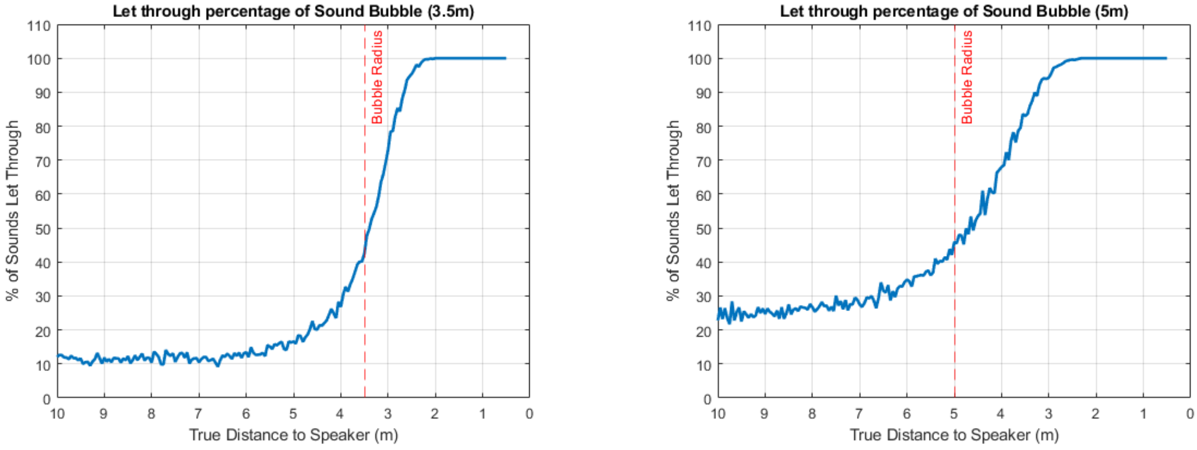


Figure 9: Performance of MATLAB simulated let-through-percentage as a function of true speaker distance (bubble size of 2.0m).

On figure 10 the performance of the two big bubble sizes (3.5m and 5.0m) are shown. These are included to see the effects of increasing bubble size even further. As seen in figure 10a a bubble size of 3.5m already has a big impact on the performance. The performance has degraded a lot. At 2.65m the let-through-percentage is below 90%, which is 0.85m from the intended threshold, and the let-through-percentage never really reaches 10% as it settles around 10%. Looking at the 5.0m bubble the performance worsens even more, with the let-through-percentage reaching 90% at 3.3m, which is 1.7m from the intended bubble threshold. Here the performance settles at around 25%.



(a) Performance of MATLAB simulated let-through-percentage as a function of true speaker distance (bubble size of 3.5m).

(b) Performance of MATLAB simulated let-through-percentage as a function of true speaker distance (bubble size of 5.0m).

Figure 10: Let-through-percentage simulated in MATLAB as a function of true speaker distance for bubbles sizes 3.5m and 5.0m.

The performance of the different bubble sizes clearly indicate that bigger bubble sizes introduces big estimation errors when utilising the inverse distance law. The reason why becomes clear when taking a look at the estimator:

$$r_{est} = \frac{d(p_1 + p_2)}{2(p_1 - p_2)}$$

As speaker distances grow the sound pressure at each microphone becomes smaller, due to the inverse distance law. At some point the pressure difference becomes so small that the estimate becomes dominated by the microphone noise. As the noise is normally distributed the resulting distance estimate starts to follow that distribution. To combat this, one could implement sanity checks or noise thresholds below which all signals will be discarded. More complex signal processing methods could be used to lower the noise or filter the signal, however this simulation is made with assumptions of a very low noise environment with no disturbances due to the test environment. For this reason, the performance would likely be worse in a more realistic scenario where interfering speakers, higher background noise, and actual voice data would complicate the estimation.

Another thing to keep in mind, is that the simulation was made with two microphones. It is assumed, that a 6-microphone system, like the one in the sound bubbles paper, would perform better, but to keep things simple and provide a proof of concept, the 2-microphone system was simulated.

5.3 Varying Speaker Angle

The next simulation was made to determine how the distance estimator, based on the inverse distance law, performed at varying speaker angles. The approach used is largely the same as in chapter 5.2, though in this simulation only the three bubble sizes from the sound bubbles paper were used. Instead of just sampling a number of distances, this simulation sampled all angle-distance combinations. The distance was sampled at every 0.1m in the interval [0.5m ; 3.0m] and the angles were sampled in the interval [0°; 180°] in 5°steps. The sample spread can be seen in figure 11.

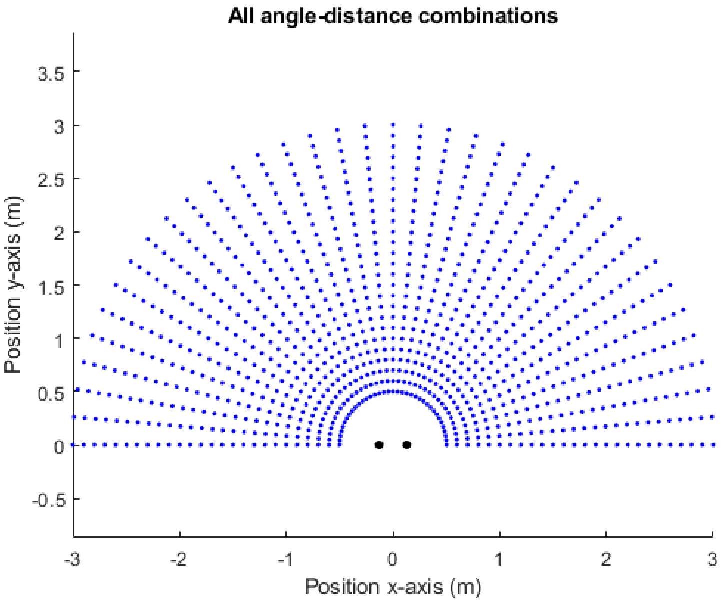


Figure 11: Position plot of all angle-distance combinations. Each dot represents a simulated speaker position. There are 26 different distances and 37 different angles for a total of $26 \cdot 37 = 962$ speaker positions.

As shown in figure 4 changing the speaker angle also changes the distance to the two microphones as well. By increasing the angle, the difference in distance between the two microphones becomes smaller, and thus a performance decline as the angle approaches 90°is expected.

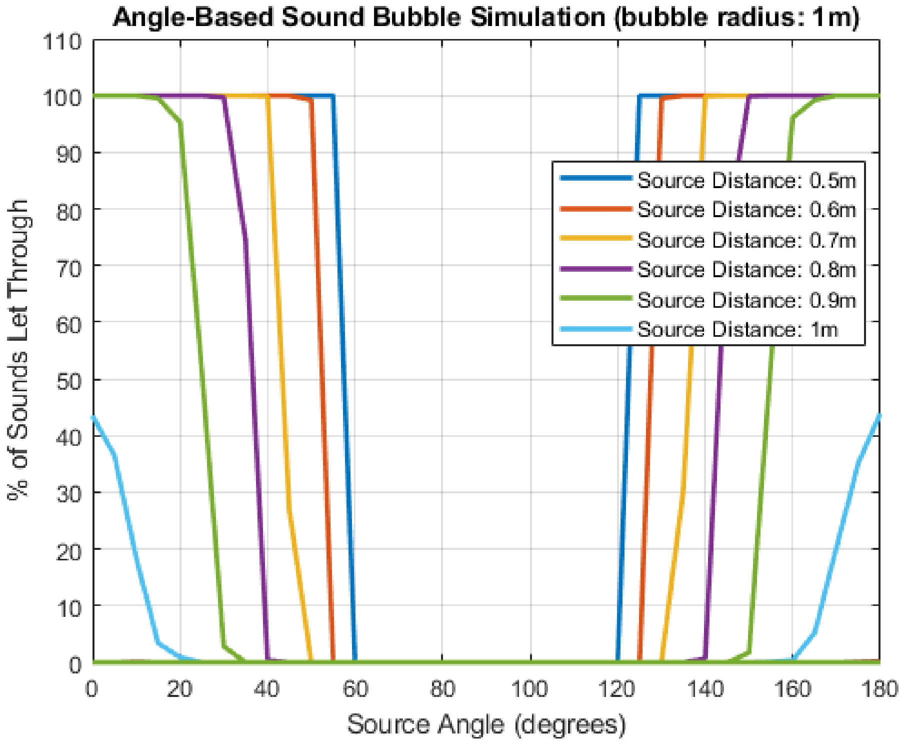


Figure 12: Performance of MATLAB simulated sound bubble as a function of speaker angle for different distances (bubble size: 1.0m).

In figure 12 the angle-performance of the 1.0m sound bubble is shown. Even at the lowest source distance (0.5m) no samples are let through between 60°and 120°. This means the simulation has a "dead" zone at 90° ± 30°. As the source distance increases, the performance becomes worse, and the "dead" zone widens letting no samples through at lower and lower angles. At any source distance above 1.0m nothing is let through. This corresponds well to the sharpness of figure 7.

Looking at the performance of the 1.5m bubble in figure 13, the simulation performs better at higher angles. At the lowest source distance (0.5m) the dead zone is between 70°and 110°, corresponding to a "dead" interval of 90° ± 20°.

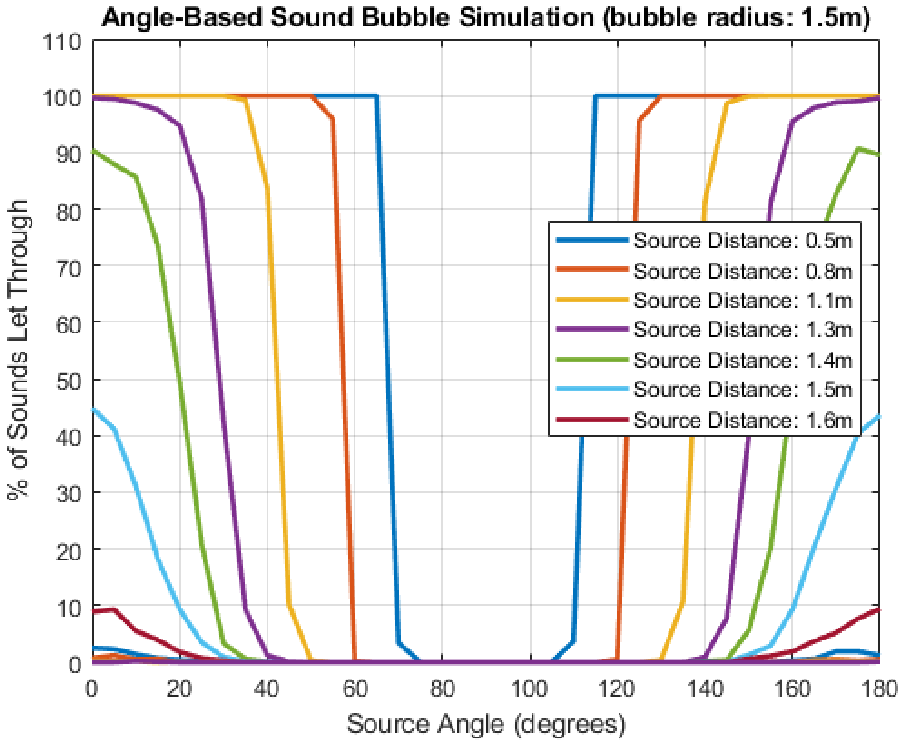


Figure 13: Performance of MATLAB simulated sound bubble as a function of speaker angle for different distances (bubble size: 1.5m)

The performance of the 2.0m bubble follows the same pattern as the previous two. The angle related performance is even better than the two smaller bubble sizes, with a “dead” zone at 90° ± 15°.

The better performance at high angles for bigger bubble sizes can be explained by looking at the estimator again along with figure 4.

$$r_{est} = \frac{d(p_1 + p_2)}{2(p_1 - p_2)}$$

As the angle increases, the difference in pressure between the microphones decreases and the denominator of r_{rest} becomes smaller. This means that the estimate increases along with the angle. For small bubble sizes, there is not much room for error, as a small increase in the distance estimate might put the estimate outside the bubble. This makes bigger bubbles more “forgiving” of this estimation error, which is why the angle-dependent performance of the model increases as the bubble size increases.

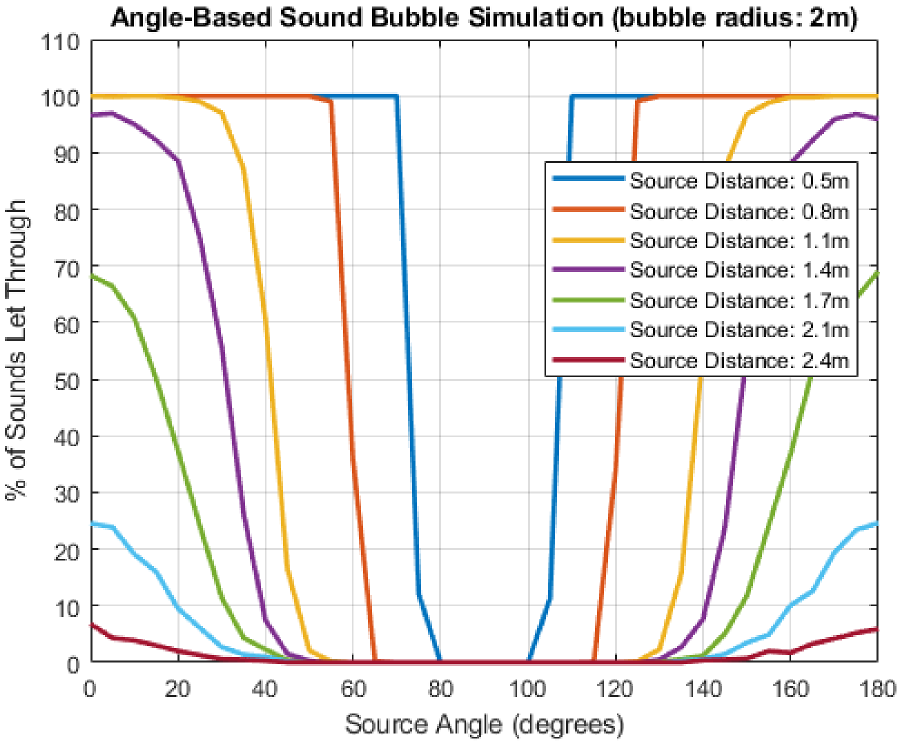


Figure 14: Performance of MATLAB simulated sound bubble as a function of speaker angle for different distances (bubble size: 2.0m)

6 Custom Data Generation

To test the specific performance metrics defined in the project objectives in chapter 4 (bubble sharpness and angle dependency), a new custom data generation script was made, to make sure it was possible to control as many variables as possible and have full control over how the data was generated.
The custom data generation script is made by heavily altering the `generate_adaptive_data.py` from the sound bubbles paper. The structure of the script is kept the same, but most functionality is rewritten to fit the project needs. The new script contains multiple different functions and loops. An overview of the custom data generation process is shown below.

1. Call script: `generate_custom_dataset.py`
2. Get list of speakers (`vctk_spit.json`)
3. Define combination of speaker distances and angles
4. For each angle and distance combination:
    (a) Call function: `generate_sample()`
        i. Call function: `select_speakers()`
        ii. Generate room with fixed dimensions
        iii. Call function: `get_mic_positions()`
        iv. Call function: `get_custom_speaker_positions()`
        v. Set absorption and max_order
        vi. Call function: `generate_data_scenario()`
        vii. Call function: `write_data()`

When the script `generate_custom_dataset.py` is called it is done so with a number of required and non-required inputs. The function prototype below shows the ones which are used in the script:
`generate_custom_dataset(input_voice_dir, output_path, --split_path, --duration)`
Here `input_voice_dir` is the directory pointing to the VCTK library and `output_path` tells the script where to put the newly generated files. Both of these are required for the script to work. `--split_path` and `--duration` are non-required inputs with default values if omitted. The `--split_path` is the path pointing to the list of speaker IDs to be used from the VCTK dataset. `--duration` is the combined duration of each generated test sample.

6.1 Choosing test audio: `select_speakers()`

Instead of using multiple different voice data sets, like in the sound bubble paper, for the custom data only the VCTK speech data is used. This is done to eliminate variation caused by the background noise added with the WHAM! dataset in the original script and to simplify the data generation. In `select_speakers()` a list of VCTK speaker IDs is copied from the `split_path` file (`vctk_split.json`). This is a list of speaker IDs divided into training, evaluation and test. From this list all "test" speaker IDs are chosen.

From this list of VCTK test speakers, `select_speakers()` choses 20 different speakers and imports an audio clip for each one. The clips are then trimmed to minimize the amount of

leading or trailing silence and are then all concatenated into one long audio clip. The audio clips in the VCTK data set vary a lot between speakers, so putting a lot of them together is done so in an attempt to avoid letting the performance depend on only one audio file. A single audio file might perform really well, where another might perform really badly. By concatenating 20 audio files, it is the average performance of the audio files which is being evaluated in the end.
The selection of speakers is made based on parameters such as the gender of the speaker and the pitch of the voice. This is to make sure that the test sample contains both light and deep voices, as the performance of the model might change depending on the type of voice.
The order in which the 20 voices are used and the chosen audio clip for each speaker is the same for every test sample. Normally this would not be optimal as a higher variety in test samples would provide a better picture of the model's ability to adapt to a variety of data. This is only done to minimize variation across samples to get consistent performance across multiple tests.

6.2 Speaker angles and -distances

The list of speaker angles and -distances is the most important variation in each test sample. This combination can be customized to achieve varying levels of resolution by varying the step size and range of the two variables. Figure 15 shows illustration of how the angles and distances are represented relative to the microphone array.

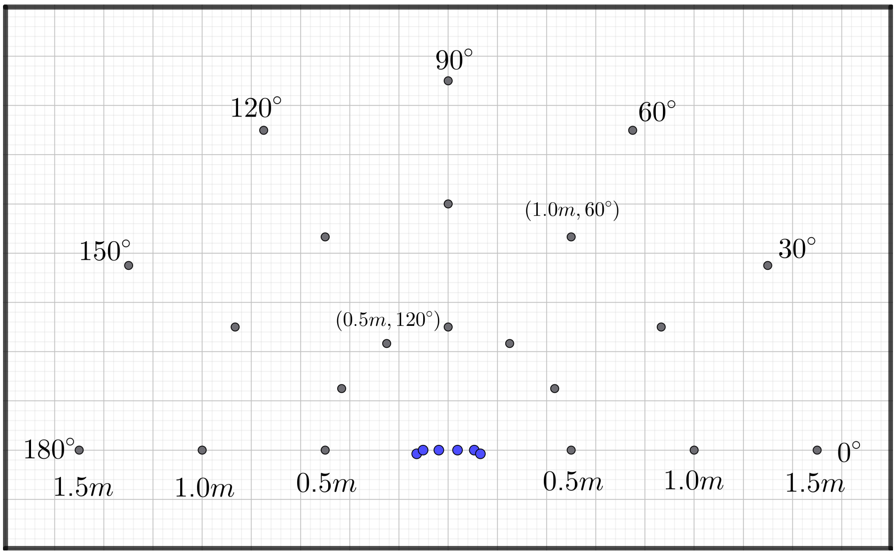


Figure 15: Illustration of how the angles-distance combinations are positioned in the script. The distance is measured in a straight line from the midpoint between the two outermost microphones of the microphone array. The angle is defined as 0°directly to the right of the microphone array along the line going through same two microphones. The angle increases from the right at 0°, to directly in front of the midpoint at 90°and directly to the left at 180°.

Directly to the right of the microphone array is 0°, 90°is directly in front, and 180°is directly to the left. The distances go from near the speaker and away at a fixed step size.

6.3 Creating the test environment

Instead of randomly generating rooms as in the data generation in the sound bubbles paper, the custom data generation generates a room with fixed dimensions. The room is defined as

12x12x3 m3. By fixing the room size, variation from different testing conditions is reduced, which will make it easier to test specific objectives such as sharpness and angle dependency. The function `get_mic_positions()` then places the microphone array the middle of the room and the function `get_custom_speaker_positions()` places the speaker at the given angle and distance from the microphone array. Both the microphone array and the speaker is placed at the same height. This is done to ensure that the microphone setup does not benefit from a height difference. If the model does indeed utilise the amplitude difference between microphones to estimate distance, then a difference in height would mean a bigger spatial diversity, which could affect the performance and make it harder to evaluate.
The absorption and max_order of the room is then set 0.99 and 0 respectively. PRA does not allow absorption of 1.0. When setting the absorption of the room, all surfaces are simulated at that same absorption. A max_order of 0 means there will be no reflections, and PRA will only simulate the direct-path audio. These parameters are chosen to eliminate as much unwanted variation as possible. After setting up the room, the function `generate_data_scenario()` simulates the room and calculates the resulting microphone signals at the six microphones in the microphone array. `write_data()` then outputs the resulting audio from the six microphones to a 6-channel audio file and creates a metadata file which contains all the information about angle, distance and room parameters.

Reducing variation

In the custom data generation, the variation besides distance and angle is reduced as much as possible. To accurately assess the model's performance for these parameters, the data needs to be predictable and repeatable.
The sound bubbles paper focuses on the performance of the model when multiple speakers are present, both inside and outside the bubble. This is great when attempting to simulate real-life scenarios. When assessing the performance of the model relative to distance and angle between microphone array and speaker, it can be difficult to determine what the effect of multiple speakers is, which is why the custom data generation only uses a single speaker.

The placement of the speakers and the microphone array is also important when generating data. In the data generation for the sound bubble paper (chapter 3.3), the room varies in absorption and number of reflections. This can have very different consequences depending on were the speaker or microphone array is in relation to the surfaces in the room.
Another thing is the way the microphone array is constructed. Most of the spatial difference between the microphones is across the x-axis (from left to right of the speaker) and the z-axis (up and down), where as the maximum variation in the y-axis (back to front) is quite small (1.5cm). As described in chapter 3.1, the model uses interchannel phase difference (IPD) and interchannel level difference (ILD). Both of these features are hindered from contributing to the model performance along the y-axis, because of the small spatial variation along that axis.

Testing the model with multiple speakers, placing the speaker and microphone array at different heights and generating different rooms with varying absorption and max_order could be tested in another project to asses the performance impact.

7 Evaluation of Sound Bubble Model

As defined in chapter 4, this project has three primary objectives. The first two of which seeks to evaluate the directional performance of the sound bubble model from the paper [3]. The two performance metrics tested in this chapter are the sharpness of the sound bubble and how it relates to bubble size, as well as the impact changes in speaker angle has on the performance of the model.

Test Procedure

To test the two objectives, data was generated as described in chapter 6. All three bubble sizes use the same custom generated dataset. The combination of speaker angles and -distances used was [0°; 180°] with a step size of 5°, and [0.5m ; 3.0m] with a stepsize of 0.05m. This resulted in $51 \cdot 37 = 1887$ test samples. The speaker position of each sample is illustrated figure 16.

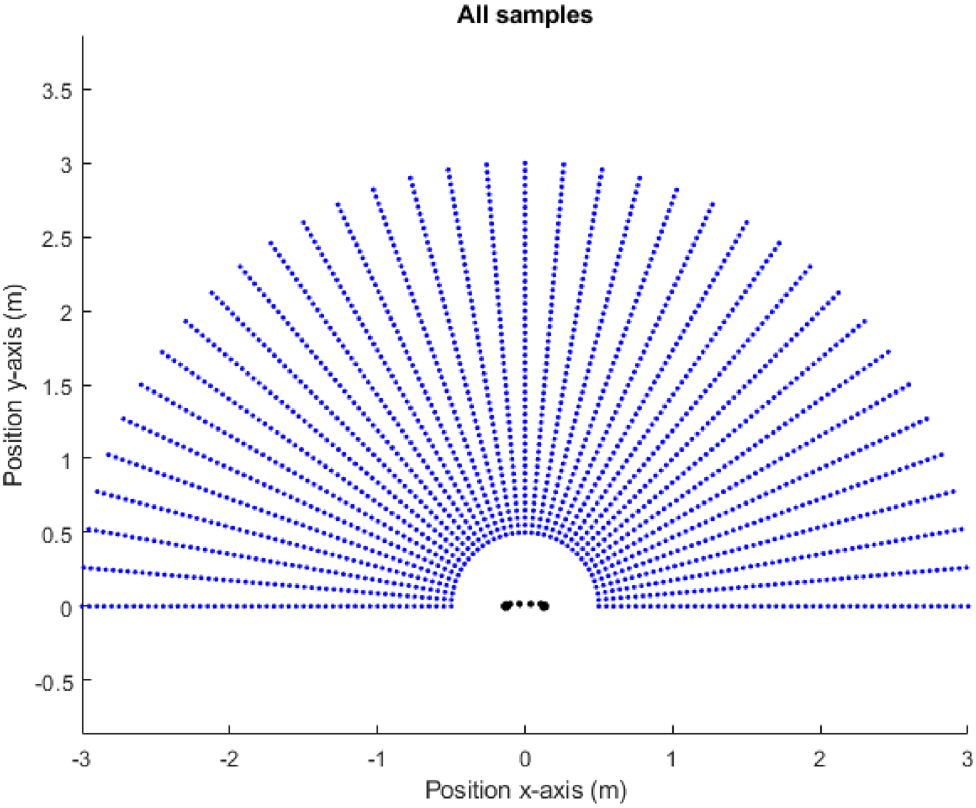


Figure 16: Speaker positions for all 1887 custom generated test samples used to evaluate the performance of the DNN model.

All the test samples are run through the DNN model for each of the three bubble sizes. To do this a modified version of the script `Sound_Bubble-main/src/test_samples.py` from the sound bubble GitHub[4] was created. The original script was made to run some test samples provided by the Chen et. al. research group. To do this Chen et. al. provided a checkpoint for synthetic datasets (which can be found in the sound bubble GitHub). The custom data was tested on the model with this checkpoint to avoid retraining.

The modified script `test_custom_samples.py` script was made to fit the custom generated data. The new script was stripped of functionality only needed for the original test data and changed the expected folder structure of the input as well as customised the structure of the output. When run, the model outputs an estimate audio file for each test sample. These estimates were stored in the folder `Sound_Bubble-main/results/est/`. Information about bubble size, positional data as well as the performance for each estimate was stored in the .csv file `Sound_Bubble-main/results/sound_bubble_metrics.csv`. A full overview of the files used in this project, as well as the folder structure can be seen in appendix A.

Plotting results

To present the resulting performance, a python script `plot_results.py` is created and placed in the `../results/` folder. The purpose of this script is to illustrate the results in a graphical manner which serves to provide a better understanding of how the model performs. The primary evaluation metric used is the signal-to-noise ratio (SNR) described in chapter 2.6. The SNR is calculated as the difference between the reference signal (the generated audio sample) and the model estimate (the output of the model). The SNR is used to measure any noise or distortion introduced by the model, but also any level difference between the two signals. This is important, as the model is expected to attenuate signals if they are identified as coming from outside the sound bubble threshold.

The results for each bubble size are presented in two ways. The first one is SNR as a function of speaker distance. In this plot the SNR/distance relation is plotted for each speaker angle. The second way the data is presented is SNR as a function of speaker angle plotted for each speaker distance. With 51 different speaker distances and 37 different speaker angles the full data plots are not easily interpretable. For this reason, all plots in chapter 7.1 contain a selected number of angles and distances. The selection of data is made in an attempt to best characterise the observed performance of the system in terms of the project objectives defined in 4. For the full plots with all data, see appendix B.

7.1 Bubble Sharpness

The first data representation is the SNR as a function of speaker distance. Starting with a bubble size of 1.5m, the bubble sharpness can be seen in figure 17.

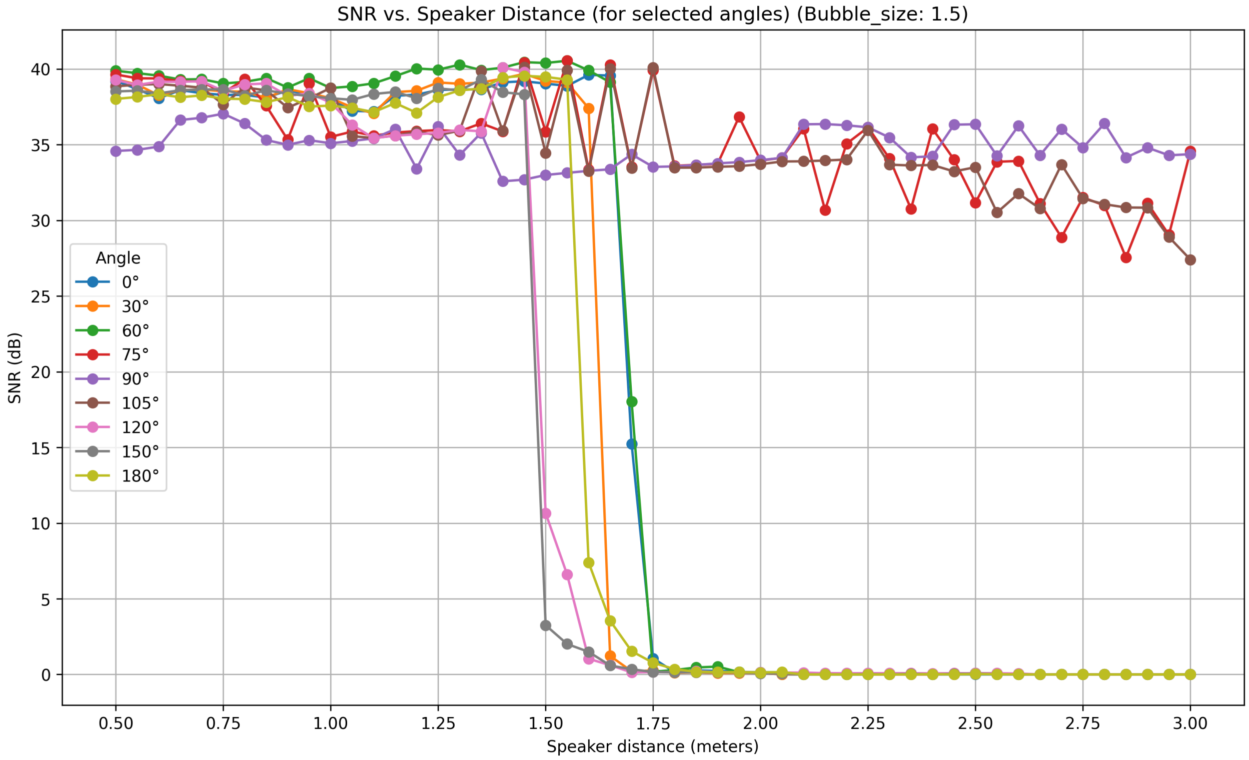


Figure 17: The SNR between the reference signal and the model output as a function of speaker distance, plotted for various angles. (Bubble size = 1.5m)

The plot shows a SNR of between 35dB and 40dB for all angles at 0.5m. This indicates that the model only introduces a small amount of noise. If the angles close to perpendicular to the listener, 90°, 75°and 105°, are ignored for a moment, the other angles seem to all drop drastically in SNR between 1.5m and 1.75m. When the SNR drops down to near zero, it means that there is suddenly a big difference between the reference and the model output. It could be that the model introduces a lot of noise, which makes the output unrecognisable compared to the reference, or it could be that the sample is attenuated. When listening to the estimates output by the model, it is clear that they are attenuated. Most of the low SNR samples are either completely quiet, or only has small residual bites of the original audio signal. The attenuation happens within 3-4 samples (15cm-20cm) for the angles in question. It seems like, however, that there is a pretty big difference between when the attenuation happens, as there is a span of 0.25m where the attenuation happens. It is difficult to identify a dependency between the angle and when the attenuation starts, however the three angles with the earliest drop off all occur at above 90°and the three angles with the latest drop off occurs below 90°. This could indicate that the model has a asymmetric performance, where it performs better with sounds coming from one side than the other. In appendix B figure 23 the full plot is shown. It shows the same picture, of a rapid but varying drop off between 1.5m and 1.75m.

Going back to the three angles 90°, 75°and 105°they show an entirely different picture. At these angles the samples are never attenuated. This could indicate that the model does indeed perform worse at angles close to 90°, like in the Matlab simulation in chapter 5.3.

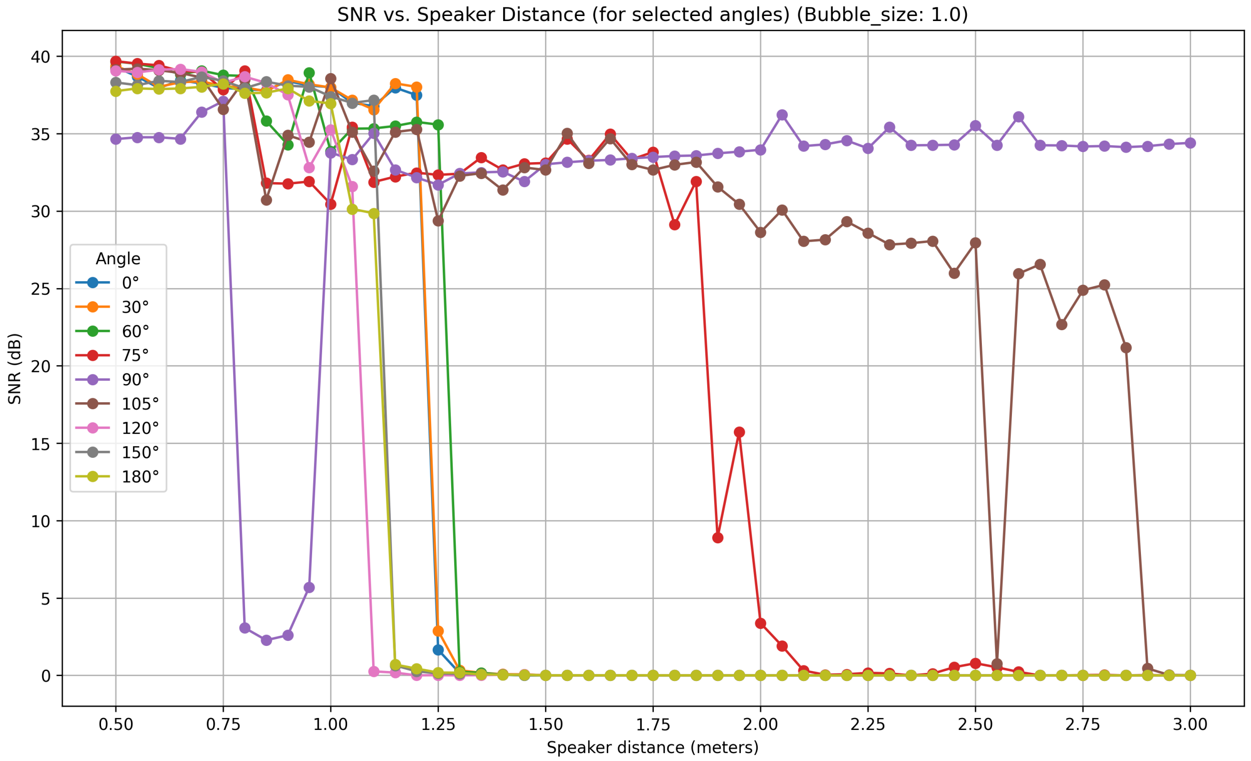


Figure 18: The SNR between the reference signal and the model output as a function of speaker distance, plotted for various angles. (Bubble size = 1.0m)

Moving on to a bubble size of 1.0m in figure 17, the performance is somewhat similar to the 1.5m bubble. The drop off happens between 1.1m and 1.35m, all within one or two samples (5cm-10cm). At the three angles at which the model behaved differently before, 90°, 75°and 105°, the results are seemingly more random. For 90°there is a sudden drop off in SNR before the bubble threshold (between 0.75m and 1.0m), and for 75°and 105°the model seems to attenuate the samples, but not nearly as early or as much as the other angles.

For the last bubble size, 2.0m, figure 19 shows similar attenuation as the two other bubble sizes, with similar drop off for angles close to 0°and 180°, and worse attenuation as the angle approaches 90°. The figure shows that the sharpness at a bubble size of 2.0m is the lowest of the three, with the drop off happening in 5-6 samples (25cm-30cm). The drop off “window” seems to be slightly wider as well. This performance resembles the Matlab simulation in chapter 5.2, as the sharpness of the bubble was negatively correlated with the bubble size. Looking at the full plot in appendix B figure 25, the drop off happens roughly between 1.70m and 2.15m.
In this case the performance at 105°(brown) shows unexpected drop off multiple places in the plot, and the performance at 60°(green) is not properly attenuated.

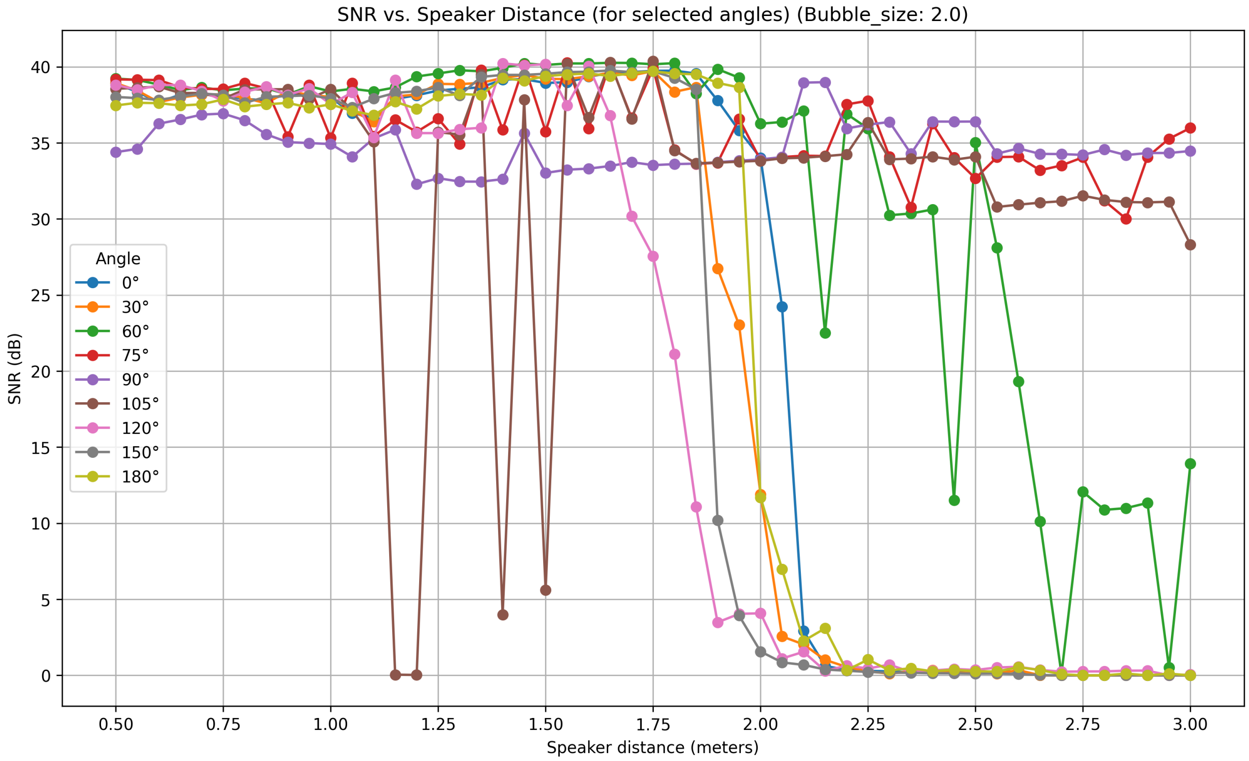


Figure 19: The SNR between the reference signal and the model output as a function of speaker distance, plotted for various angles. (Bubble size = 2.0m)

Evaluation

There seem to be a clear relation between sharpness and bubble size, where a smaller bubble sizes results in a sharper boundary. This corresponds well with the observed relation between bubble size and sharpness seen in the Matlab simulation in chapter 5.2. When looking at the full plots in appendix B, the 1.0m and 2.0m bubbles seem to have more unexpected behaviour, where the signal is attenuated in a short ranges of distances away from the bubble boundary. What causes this is uncertain.

7.2 Angle Dependency

The next part of the evaluation is the plot of SNR as a function of speaker angle plotted for each distance. As with the bubble sharpness, the plots in this chapter contains a selected number of distances for each bubble size. For the full plots see appendix B.
The plot for a bubble size of 1.5m can be seen in figure 20. The selection of speaker distances was made to best represent what happens at distances close to the bubble boundary. Like the figures in chapter 7.1 the SNR is close to 40dB for most distances at most angles. For the distance 1.40m the SNR stays around 40dB except for between 75°and 110°. In this interval the model seem to perform very similar for all distances, dropping 5dB-10dB. At distances between 1.50m and 1.70m the SNR seem to fluctuate throughout all angles except the ones close to 90°. This seems to be a pattern; that the model fluctuates around the bubble threshold. For distances of 1.75m and above the signal is almost completely attenuated until between 60°and 120°. In this interval, 90° ± 30°, all of the samples have a high SNR and are only slightly attenuated. This suggests that the performance of the model is significantly hindered at angles close to 90°. This performance again resembles the angle dependency seen in the Matlab simulation in chapter 5.3.

By taking a look at the full plot, appendix B figure 26, it is clear the most samples at 95°are attenuated, some more than others. By plotting different combinations of distances, it was discovered that all of the distances which were the most attenuated at 95°were either significantly above or below the bubble threshold. It is difficult to determine what the cause of this is.
It is also interesting to note that this performance issue is clearly centred around 95°and not 90°. This suggests asymmetric performance, like noted in chapter 7.1.

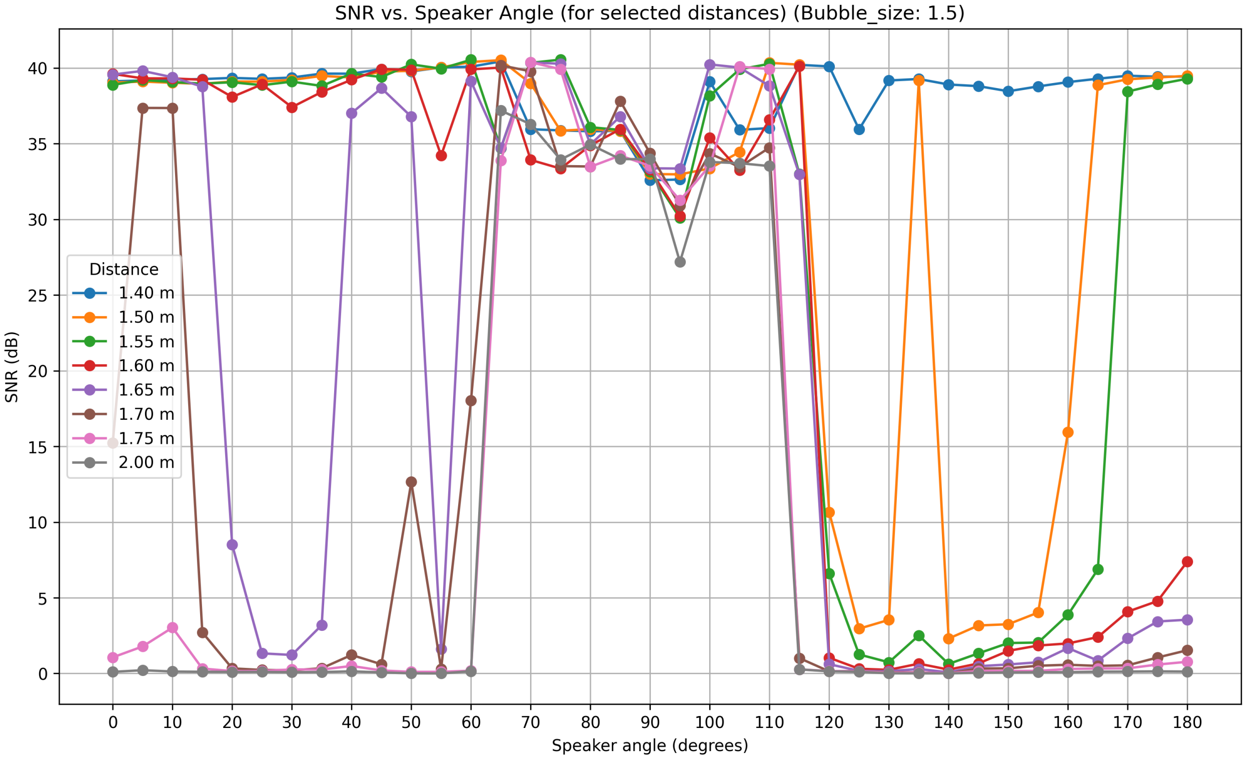


Figure 20: The signal to noise ratio between the reference signal and the model output as a function of speaker angle for a bubble size of 1.5m

The plot for 1.0m bubble size in figure 21 presents a less neat picture. Here there seems to be a big drop in SNR at around 95°as well. The drop in SNR is sustained for 1-2 samples more (5cm-10cm) which results in a bigger region where the model's ability to attenuate diminishes. As in figure 20 there seems to exist a region around the bubble threshold, where the performance of the model is less reliable, in this case between 1.0m and 1.3m. The region of angles where the distances outside the bubble threshold are not attenuated is slightly more narrow for a 1.0m bubble. Looking at the full plot in appendix B figure 27, the region seems to be between 70°and 110°.

With a bubble size of 2.0m, figure 22 shows a plot more similar to the 1.5m plot. There is a clearly defined region within which the samples are not attenuated at any distances. In the full plot in appendix B figure 28, there are a couple of samples which are attenuated a lot at between 95°and 105°, but a bubble size of 2.0m still has the least attenuation at angles close to 90°.

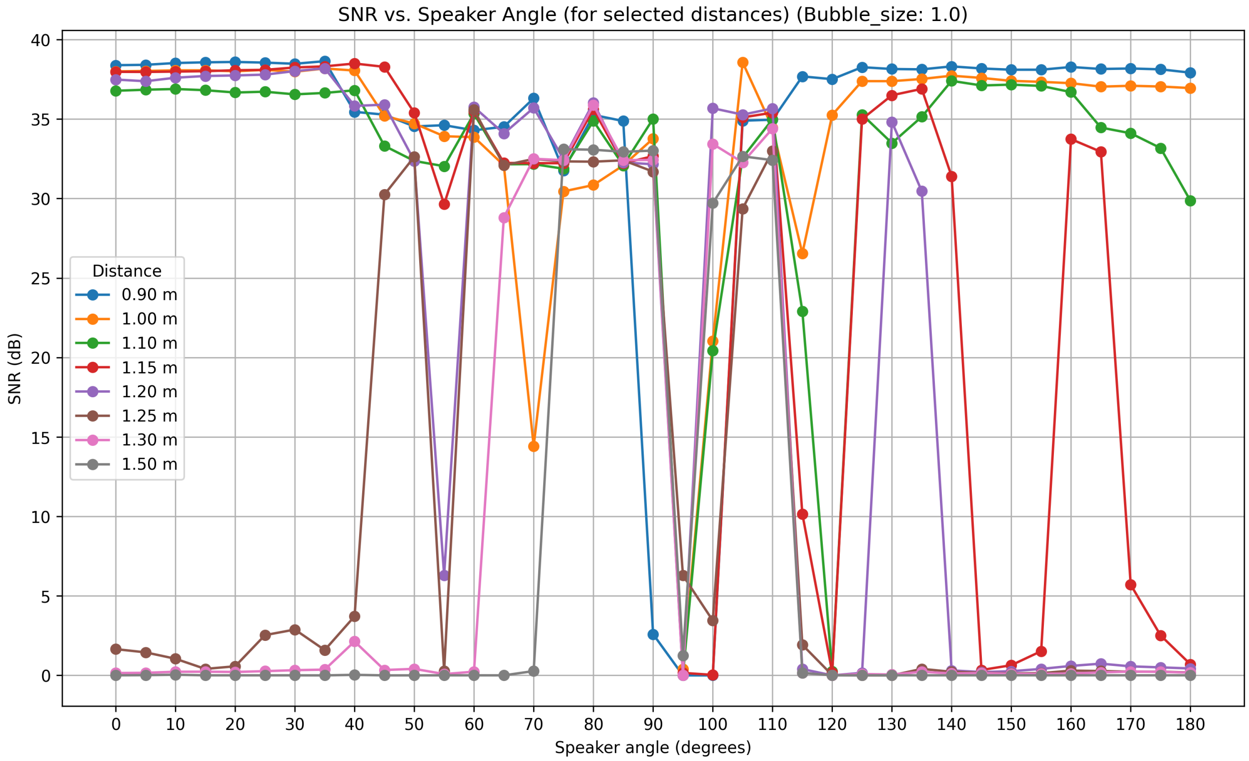


Figure 21: The signal to noise ratio between the reference signal and the model output as a function of speaker angle for a bubble size of 1.0m

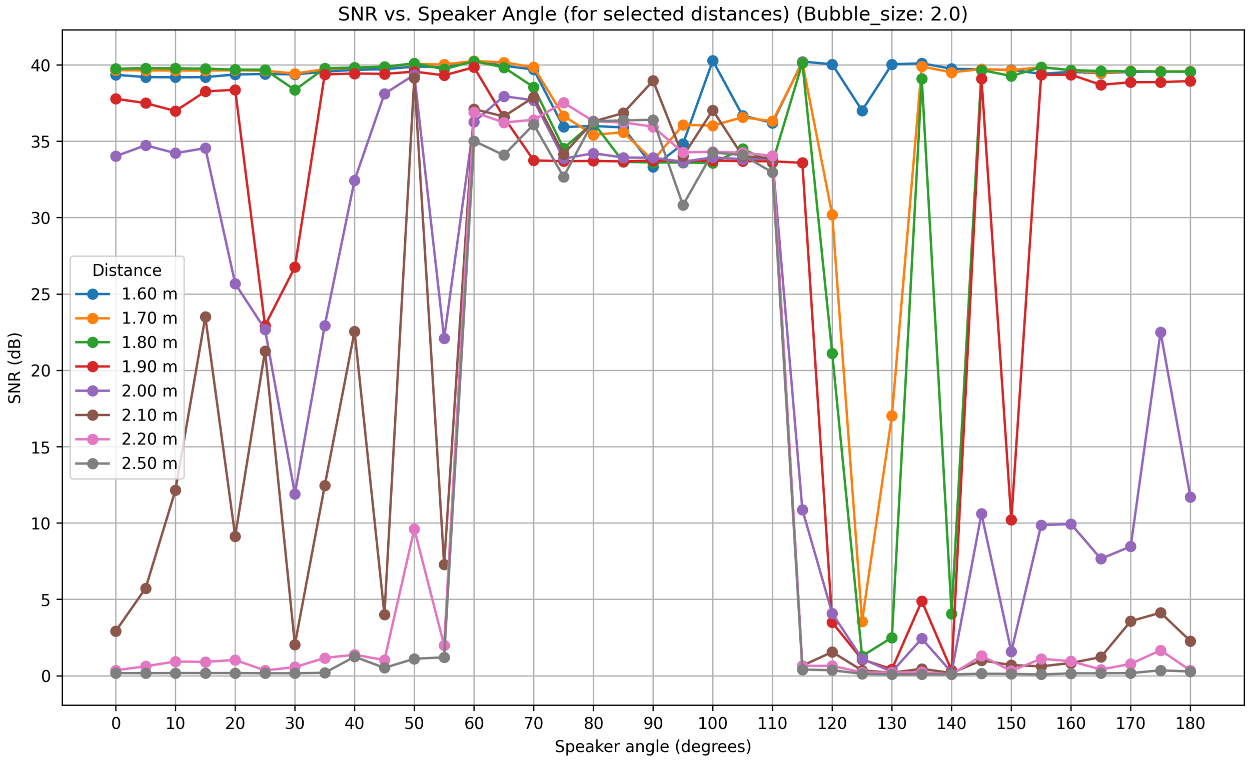


Figure 22: The signal to noise ratio between the reference signal and the model output as a function of speaker angle for a bubble size of 2.0m

Evaluation

The results presented in this chapter clearly indicate that the model performs worse when the speaker angle is close to 90°. There seems to be a consistent asymmetric performance, where the expected performance of the model is offset by 5°-10°. In the Matlab simulation, the performance at angles close to 90°became better as the bubble size increased. In this chapter the opposite seems to be the case, as lower bubble sizes seem to result in better performance for the model at angles close to 90°.

7.3 Peformance Table

In table 1, the performance of the different bubble sizes are compared.

Bubble Size	**Sharpness**	**Drop Interval**	**Worst Angles**	**Notes**
1.0m	5cm - 10cm	[1.10m - 1.35m]	75 - 110	High sharpness, best angular performance
1.5m	15cm - 20cm	[1.50m - 1.75m]	70 - 110	Medium sharpness, slightly worse angular performance
2.0m	25cm - 30cm	[1.70m - 2.15m]	60 - 115	Low sharpness, worst angular performance.

Table 1: Comparison of the performance of each bubble size.

8 Discussion

This project aimed to explore the behaviour and performance of the Sound Bubble system proposed by Chen et al. [3]. Key objectives were defined as investigating the bubble sharpness and angular dependency of the model as well as discussing whether the neural network utilises the inverse square law. To do this a thorough study of the paper and the associated code [4] was conducted, a simplified Matlab-based simulation was made and the model was tested with custom generated data.

8.1 Bubble Sharpness and Spatial Selectivity

One of the primary goals was to examine how sharp the transition was between inside and outside the bubble. This sharpness was approximated with the Matlab model based on the inverse distance law, and then evaluated for the DNN model by testing it with the custom data. The Matlab simulation (chapter 5.2) showed a clear boundary which became less sharp as the bubble size increased. This same behaviour was exhibited by the DNN model. The sharpness observed was around 40dB attenuation over 5-10 centimetres for the smallest bubble size (1.0m) and 25-30 centimetres for the biggest bubble size (2.0m). The precision of the drop-off is a range of 25cm for 1.0m and 1.5m and 45cm for 2.0m. For 1.5m there were angles where the drop-off happened immediately at the bubble threshold of 1.5m. For 1.0m there was a delay of around 10cm before some of the samples got attenuated, and for 2.0m the attenuation started 30cm before the bubble threshold. This could indicate that the accuracy of the drop-off is dependent on the bubble size. More bubble sizes need to be tested to be certain of the effect.

The similarity between the results could suggest that the inverse distance law plays a role in the performance of the DNN model. The results from the MATLAB simulation also indicate that even with a simple system, without complex machine learning, it is possible to create a sound bubble with a relatively sharp bubble boundary.

8.2 Angle Dependency of Performance

Another key parameter investigated was the angular dependency of the bubbles performance. The original paper does not show clearly how the performance of the system relates to the speaker angle. The Matlab simulation in chapter 5.3 showed a systematic performance loss for angles near 90°. This was expected, as the distance estimator was made relying on the pressure difference between the two microphones. It was seen that as the speaker angle moved closer to 90°, at some point, depending on the bubble size, there was a boundary beyond which the simulation always estimated the distance to be outside the bubble, reducing the chance of letting a sound through to near zero. This angular performance was worsened as the true distance approached the bubble boundary. This performance became worse as the bubble size decreased.

For the model tested with custom data the same angular dependency was observed. For all three bubble sizes there was a region on the SNR/angles plot around 90°where the samples were only either slightly attenuated or not at all. The same thing was observed on the SNR/distance plot, where the samples were not attenuated properly close to 90°. Unexpected behaviour was also observed at around 90°, as the signal seemed to fluctuate a lot more close to this angle. In the SNR/angle plot, there was a very strong attenuation at 95°-00°. This could indicate that the model has asymmetric performance with an offset of 5°- 10°.

For the three bubble sizes, it seemed like the degradation of performance due to angle was worst at higher bubble sizes, which does NOT correspond with the expected result from the

Matlab simulation (chapter 5.3). However, there is still a strong resemblance between the Matlab simulation and the performance of the DNN model.

8.3 Role of the Inverse Distance Law

One of the key questions raised in this project, was whether the inverse distance lawthe physical principle that sound pressure decreases proportionally to $1/r$plays a significant role in the DNN model's decision-making process. In the DNN model, the STFT and spatial features (ILD, IPD) implicitly encode distance cues, and the network is conditioned via a distance embedding mask. However, it is unclear which relations the speech separation module learns.

In the Matlab simulation, where signal strength follows the inverse distance law explicitly, sharp sound bubbles were produced, especially for small bubble radii. The same simulation showed poor angular performance at small bubble radii. The similarity between these results and the performance observed by the DNN model fed with custom data suggests that the inverse distance law do play a role. The performance also clearly indicate that the DNN utilises other concepts, and is not reliant on the inverse distance law alone. As the model has six microphones, it is most likely able to approximate the distance more precisely than the Matlab simulation.

8.4 Limitations of the Simplified Analysis

While the results in the Matlab simulation and model test with custom data generation are similar, several simplifications must be acknowledged. First, the Matlab simulation is done with perfectly non-reverberant conditions with deterministic noise and no reflections or interfering speakers. In practice, these environmental effects would likely significantly impact the performance. The simplified sine wave simulation of speech is also a significant assumption, as the simulation would need much more complexity with real audio signals.

Second, the custom data is generated at close to perfect non-reverberant conditions with little to no noise. To most accurately evaluate the sharpness and angular dependency, all test samples contained the exact same audio signals. Even though an attempt was made to approximate an average performance, by concatenating many audio signals, a real model would face a much more diverse sound profile. With no variation in room size or absorption/reflections the data most likely does not generalise to real-world settings.

Finally, the spatial diversity of the test setup was also minimised in the custom data generation. By fixing the height of the speaker to be equal the height of the microphone array, the spatial diversity along the y-axis (in front of the microphone array) was minimised. This was done in an attempt to force the model to show any dependency on the inverse distance law. Varying the height difference would most likely alter the performance of the model.

Despite these limitations, the results offer valuable insight into the performance and limitations of the Sound Bubble system proposed in the sound bubbles paper.

9 Conclusion

This project investigated the directional audio system proposed in the Sound Bubble paper through a combination of simplified simulation and custom test data generation. The focus was placed on understanding and evaluating the spatial behaviour of the system; specifically the sharpness of the bubble boundary, the angle-dependent performance, and the potential influence the inverse distance law.

A Matlab-based simulation, with a distance estimator based on the inverse distance law, was developed to approximate the behaviour of the original system. Although it simplified many aspects of the original model, such as omitting reverberation, background noise, and neural network complexity, the simulation was still able to reproduce a sharp transition between the inside and outside of a sound bubble. As expected, a degradation in performance at wider speaker angles were observed. These two metrics were then replicated with custom generated data which was fed through the DNN model of the original paper, with a bubble size dependent sharpness, and angular performance degradation at wider speaker angles.

The project explored whether the inverse distance law contributes to the models spatial performance. While not explicitly shown, the inverse distance law appears to influence the performance of the system, due to the resemblance between the simulated and tested results.

Overall, the findings suggest a somewhat dependency of the inverse distance law. However these results are produced with major simplifications such as non-reverberant environments, deterministic test data and low interference from background noise and interfering speakers. This limits the extent to which these results can be generalized to real-world settings. Future work could extend this analysis by incorporating more realistic room acoustics, speaker variability, and microphone imperfections.

10 Future Work

While this project has provided a simplified yet informative analysis of the Sound Bubble system, several directions remain open for further exploration and enhancement.

First, future work should incorporate more realistic acoustic environments into the simulation framework. The current assumption of an ideal, non-reverberant room limits the systems ability to model real-world variability. Introducing room reflections, more realistic absorption, diffuse noise, and microphone imperfections would allow for a more realistic evaluation.

Second, extending the custom test data generation to include multiple simultaneous speakers, varying speech content, and dynamic source movements could better evaluate the systems separation accuracy under complex conditions.

Finally, further training experiments could be conducted to assess whether the system remains effective under different microphone configurations or larger bubble sizes. This would support easier deployment in real-world audio applications where flexibility, latency, and robustness are critical.

By addressing these areas, future studies could advance both the theoretical understanding and the practical applicability of directional sound control using spatially aware neural networks.

References

[1] SP Banbury and DC Berry and. “Office noise and employee concentration: Identifying causes of disruption and potential improvements”. In: *Ergonomics* 48.1 (2005). PMID: 15764304, pp. 25–37. DOI: 10.1080/00140130412331311390. eprint: https://doi.org/10.1080/00140130412331311390. URL: https://doi.org/10.1080/00140130412331311390.

[2] P. Ella Braat-Eggen et al. “Noise disturbance in open-plan study environments: a field study on noise sources, student tasks and room acoustic parameters”. In: *Ergonomics* 60.9 (2017). PMID: 28287041, pp. 1297–1314. DOI: 10.1080/00140139.2017.1306631. eprint: https://doi.org/10.1080/00140139.2017.1306631. URL: https://doi.org/10.1080/00140139.2017.1306631.

[3] Tuochao Chen et al. “Hearable devices with sound bubbles”. In: *Nature Electronics* 7 (2024), pp. 1047–1058. URL: https://doi.org/10.1038/s41928-024-01276-z.

[4] Tuochao Chen et al. *Sound_Bubble GitHub*. https://github.com/chentuochao/Sound_Bubble/ (19/05-2025).

[5] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[6] International Noise Awareness Day. *Common Noise Levels - How loud is too loud?* https://noiseawareness.org/info-center/common-noise-levels/ (21/05-2025).

[7] Seeed Studio. *ReSpeaker 6-mic Circular Array*. https://wiki.seeedstudio.com/ReSpeaker_6-Mic_Circular_Array_kit_for_Raspberry_Pi/ (30/05-2025).

[8] Jonathan Le Roux et al. *SDR - half-baked or well done?* 2018. arXiv: 1811.02508 [cs.SD]. URL: https://arxiv.org/abs/1811.02508.

[9] Anthony W. Rix et al. “Perceptual evaluation of speech quality (PESQ)a new method for speech quality assessment of telephone networks and codecs”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* 2 (2001), pp. 749–752.

[10] Cees H. Taal et al. “A short-time objective intelligibility measure for time-frequency weighted noisy speech”. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 4214–4217. DOI: 10.1109/ICASSP.2010.5495701.

[11] H. et al. Zen. “LibriTTS: a corpus derived from LibriSpeech for text-to-speech.” In: *In INTERSPEECH 15261530 (International Speech Communication Association)* (2019).

[12] Christophe Veaux, Junichi Yamagishi, and Kirsten MacDonald. “CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit”. In: *University of Edinburgh. The Centre for Speech Technology Research (CSTR)* (2017).

[13] Gordon Wichern et al. “WHAM!: Extending Speech Separation to Noisy Environments”. In: *Proc. Interspeech*. Sept. 2019.

[14] Peter Ladefoged, Keith Johnson, and Peter Ladefoged. *A course in phonetics*. Vol. 3. Thomson Wadsworth Boston, 2006.

A Full Folder Structure

An overview of the full folder structure is presented here.

```
Sound_Bubble-main/
├── src/
│   └── test_custom_samples.py
├── custom_test_samples/
│   ├── audio/
│   │   ├── sample_0000_d0.50_a000_source.wav
│   │   ├── sample_0001_d0.55_a000_source.wav
│   │   ├── sample_0000_d0.60_a000_source.wav
│   │   └── (...)
│   └── metadata/
│       ├── sample_0000_d0.50_a000_metadata.json
│       ├── sample_0001_d0.55_a000_metadata.json
│       ├── sample_0000_d0.60_a000_metadata.json
│       └── (...)
├── results/
│   ├── est/
│   │   ├── sample_0000_d0.50_a000_estimated.wav
│   │   ├── sample_0001_d0.55_a000_estimated.wav
│   │   ├── sample_0000_d0.60_a000_estimated.wav
│   │   └── (...)
│   ├── plot_results.py
│   └── sound_bubble_metrics.csv
└── generate_custom_dataset.py
```

B Full Test Data

This section presents the full unaltered data.

SNR as a function of distance for all angles:

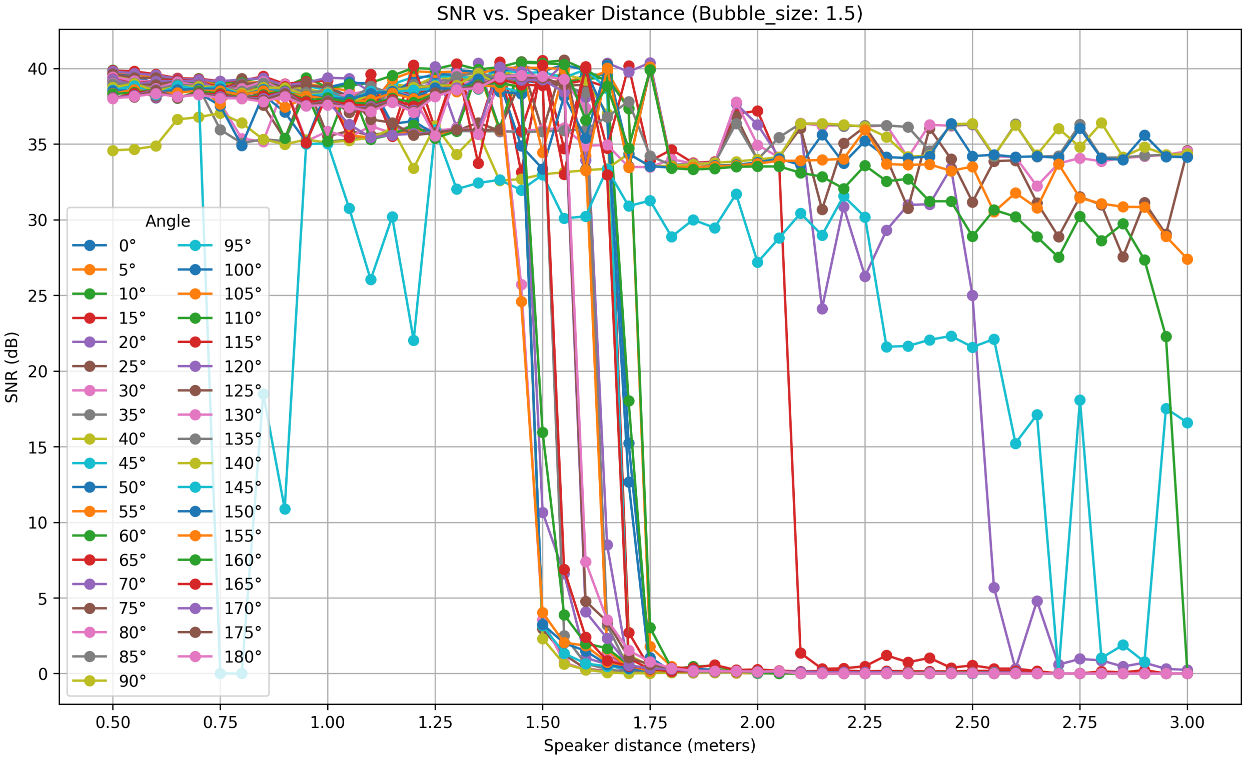


Figure 23: SNR as a function of distance for all angles (bubble size: 1.5)

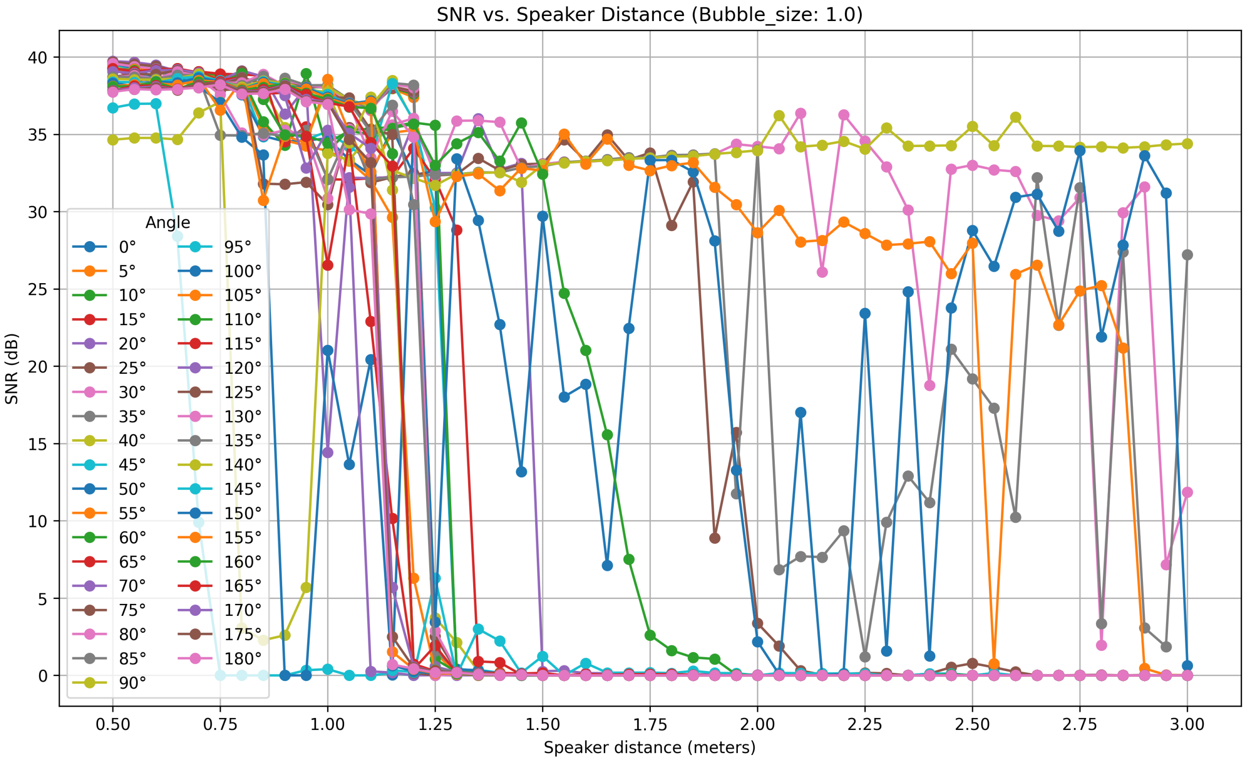


Figure 24: SNR as a function of distance for all angles (bubble size: 1.0)

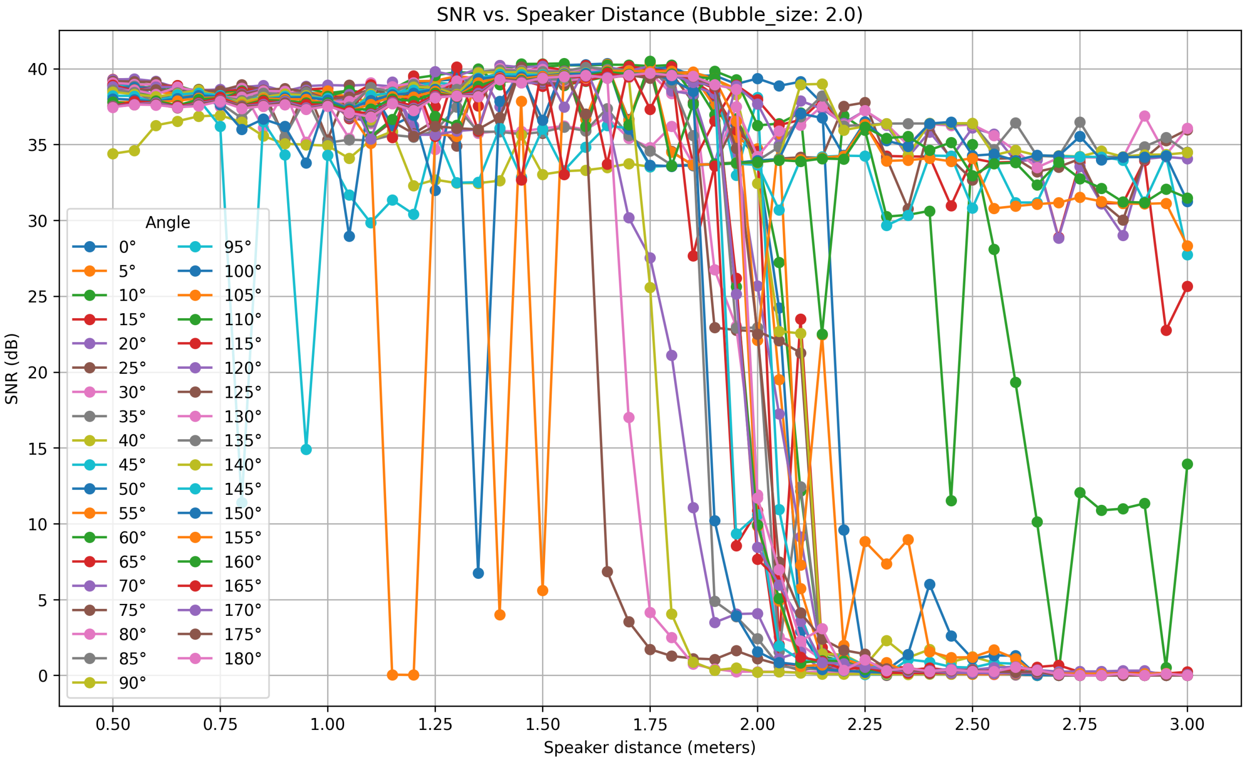


Figure 25: SNR as a function of distance for all angles (bubble size: 2.0)

SNR as a function of angle for all distances:

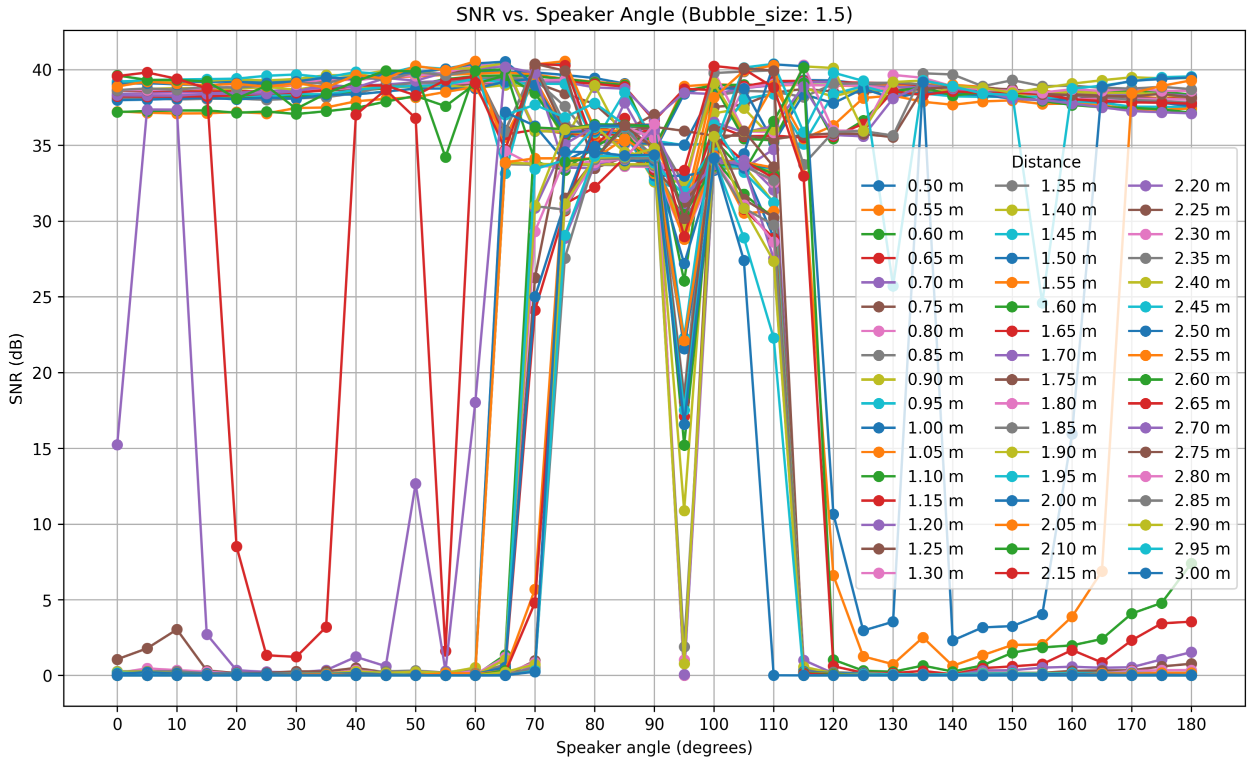


Figure 26: SNR as a function of angle for all distances (bubble size:)

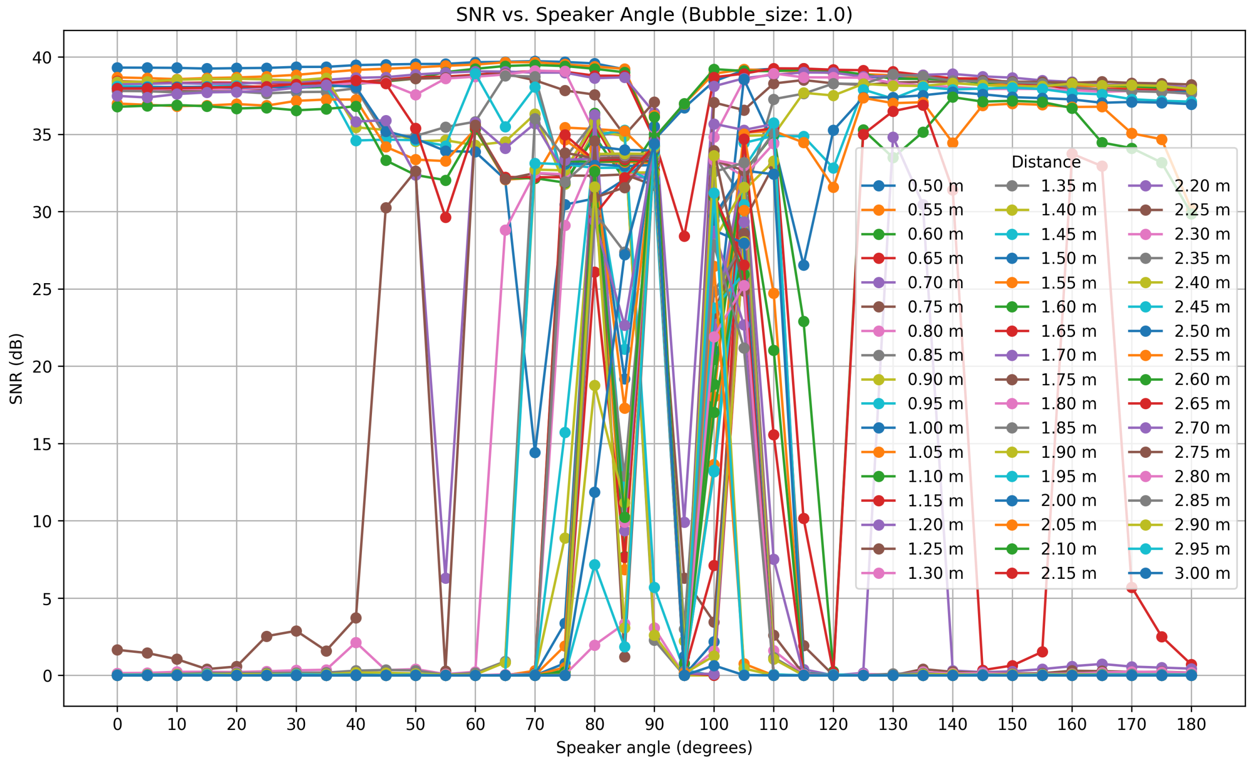


Figure 27: SNR as a function of angle for all distances (bubble size:)

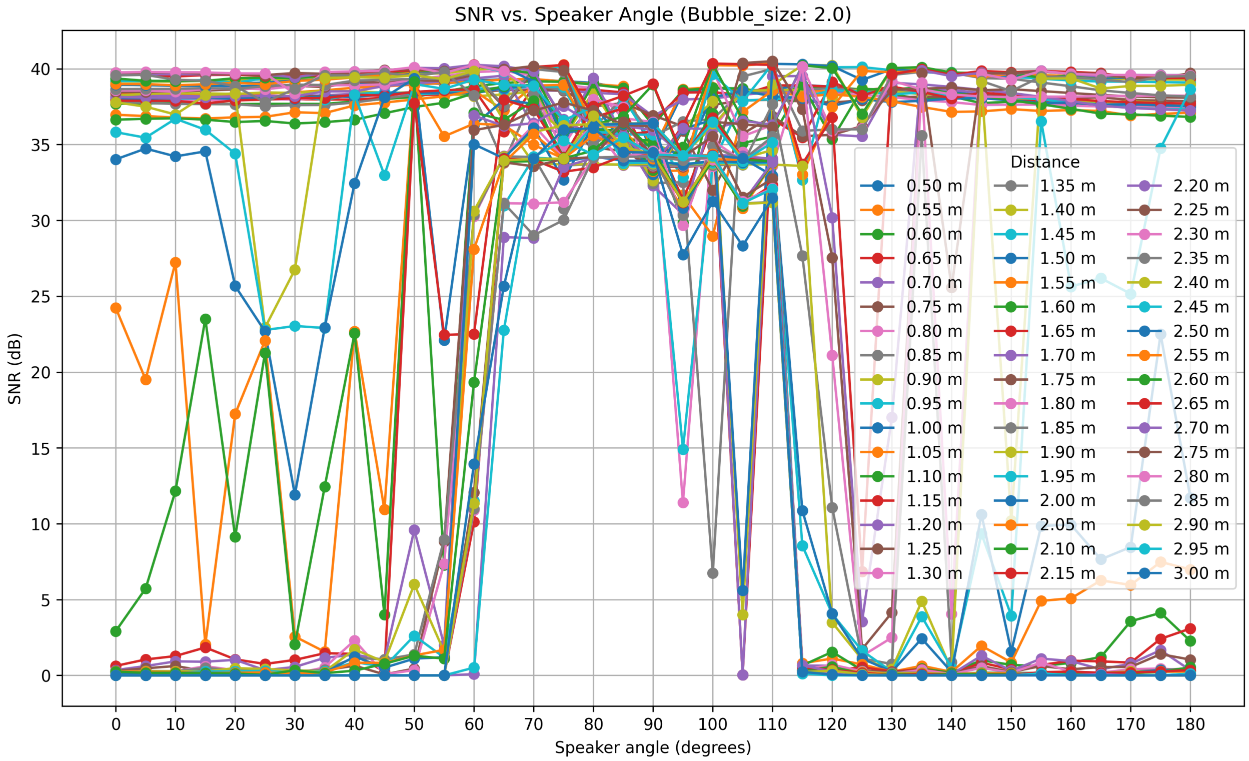


Figure 28: SNR as a function of angle for all distances (bubble size:)

STOI as a function of angle for all distances:

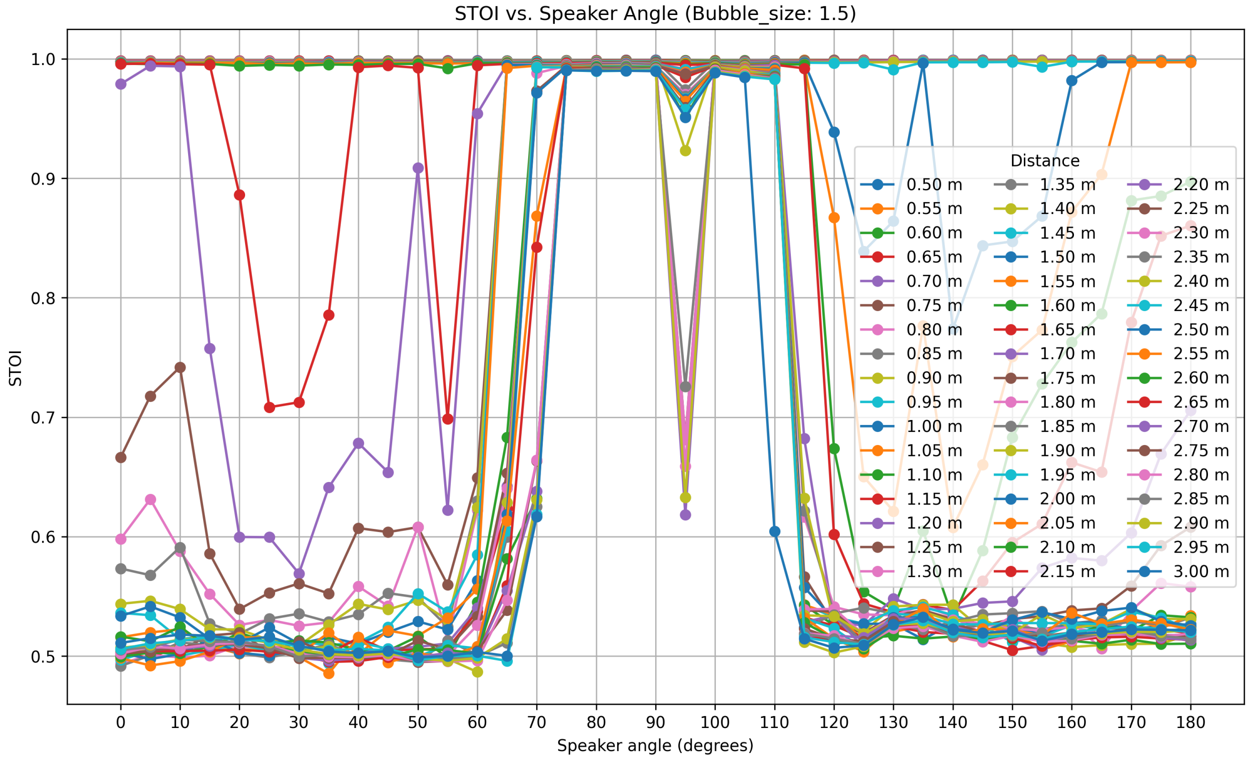


Figure 29: STOI as a function of angle for all distances. (bubble size: 1.5)

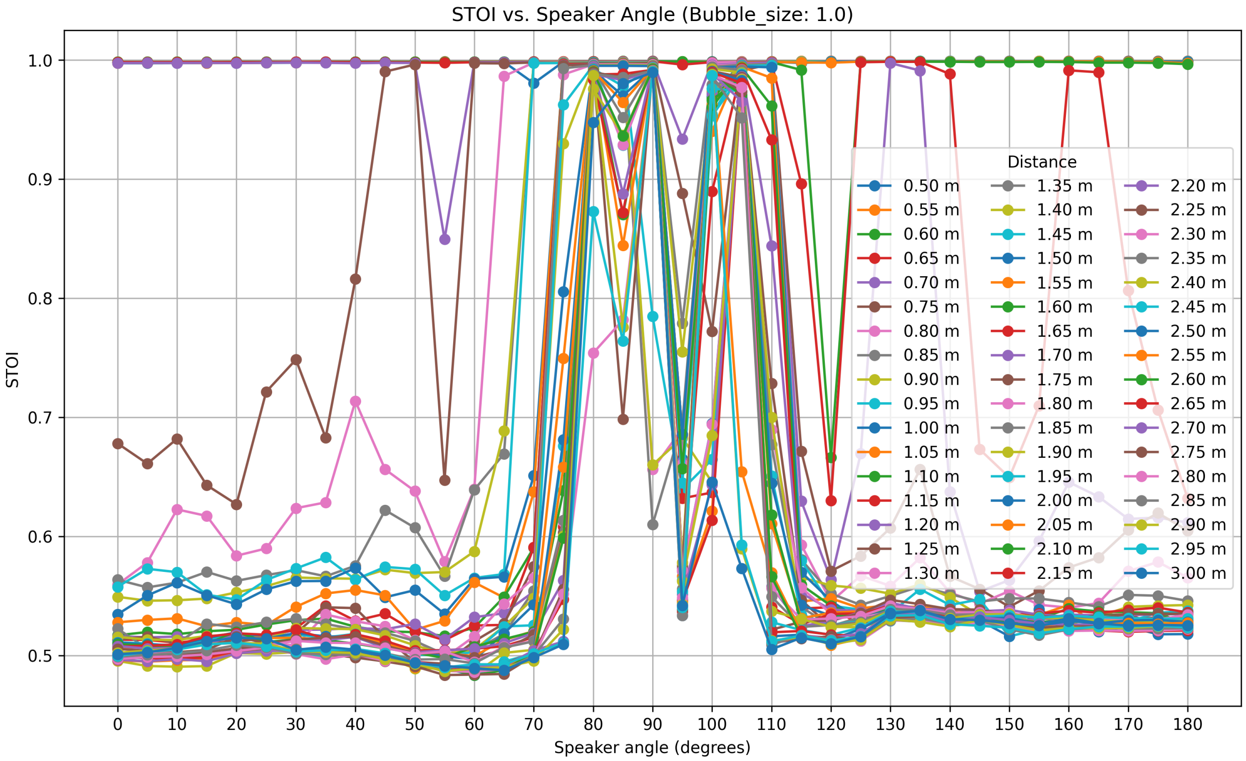


Figure 30: STOI as a function of angle for all distances. (bubble size: 1.0)

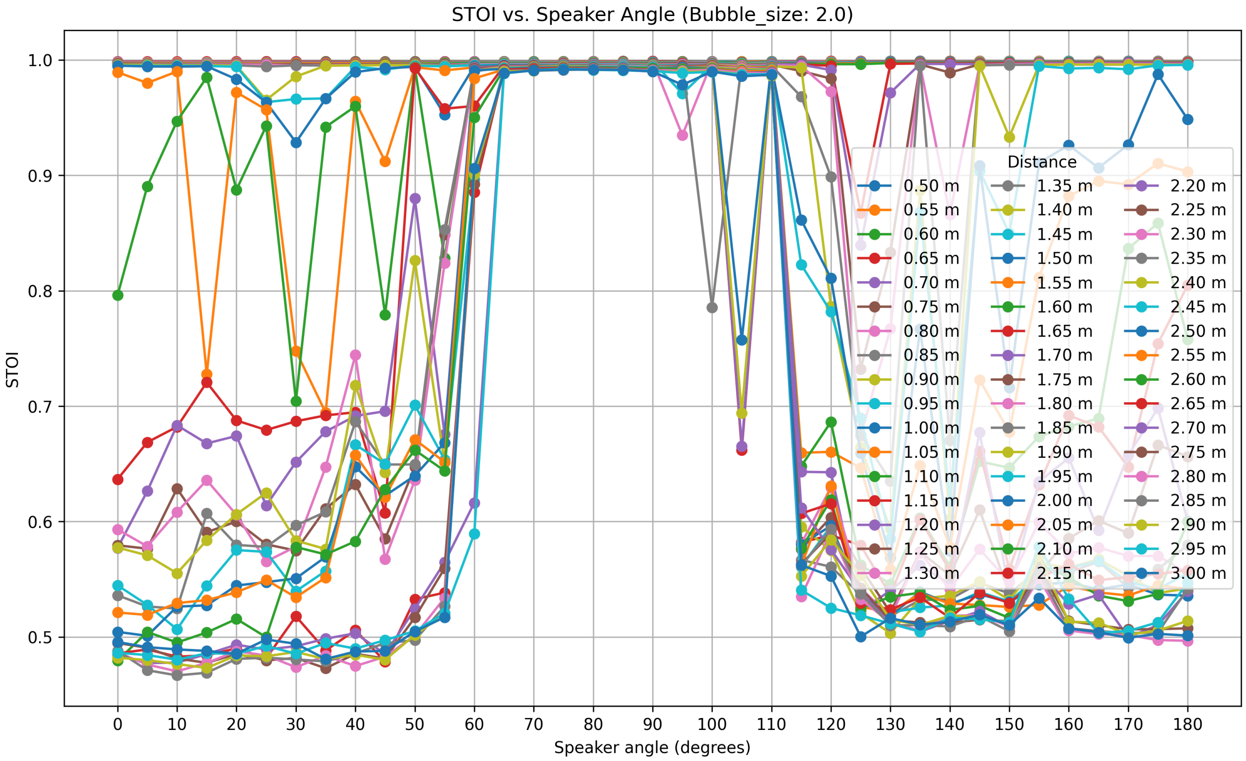


Figure 31: STOI as a function of angle for all distances. (bubble size: 2.0)

PESQ as a function of angle for all distances:

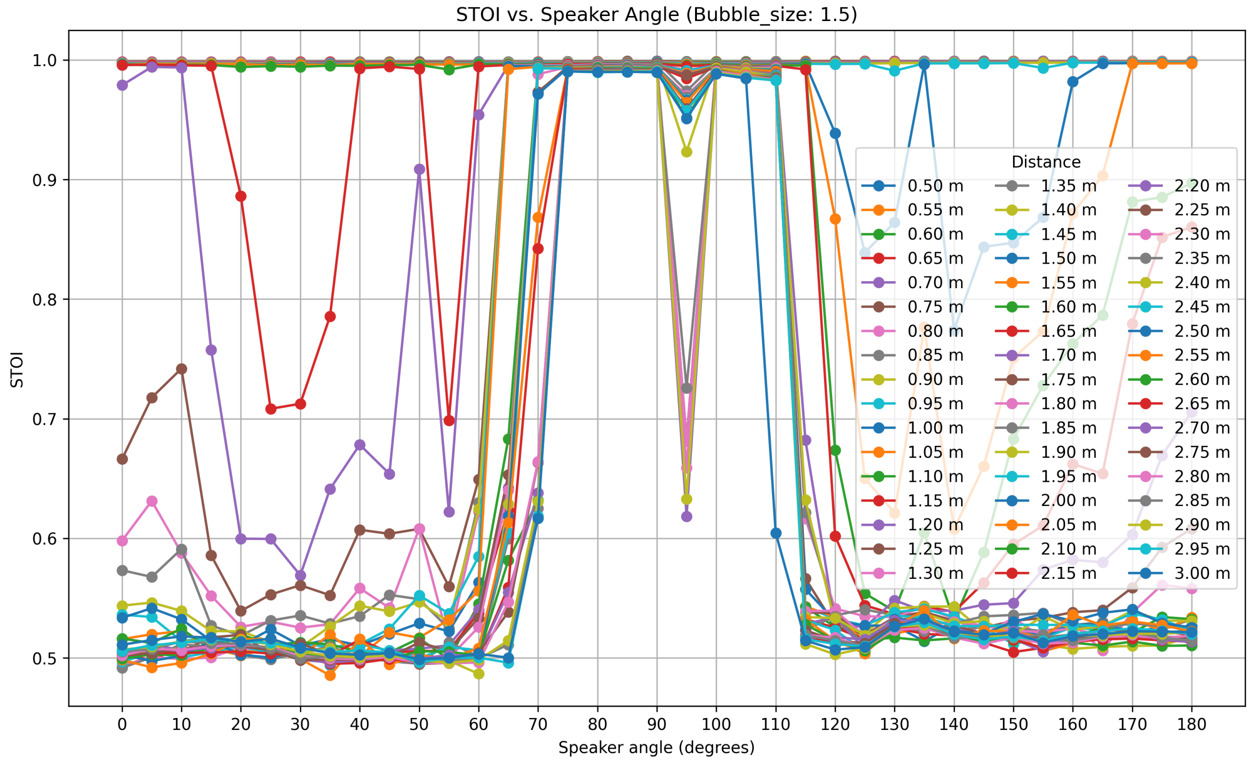


Figure 32: PESQ as a function of angle for all distances. (bubble size: 1.5)

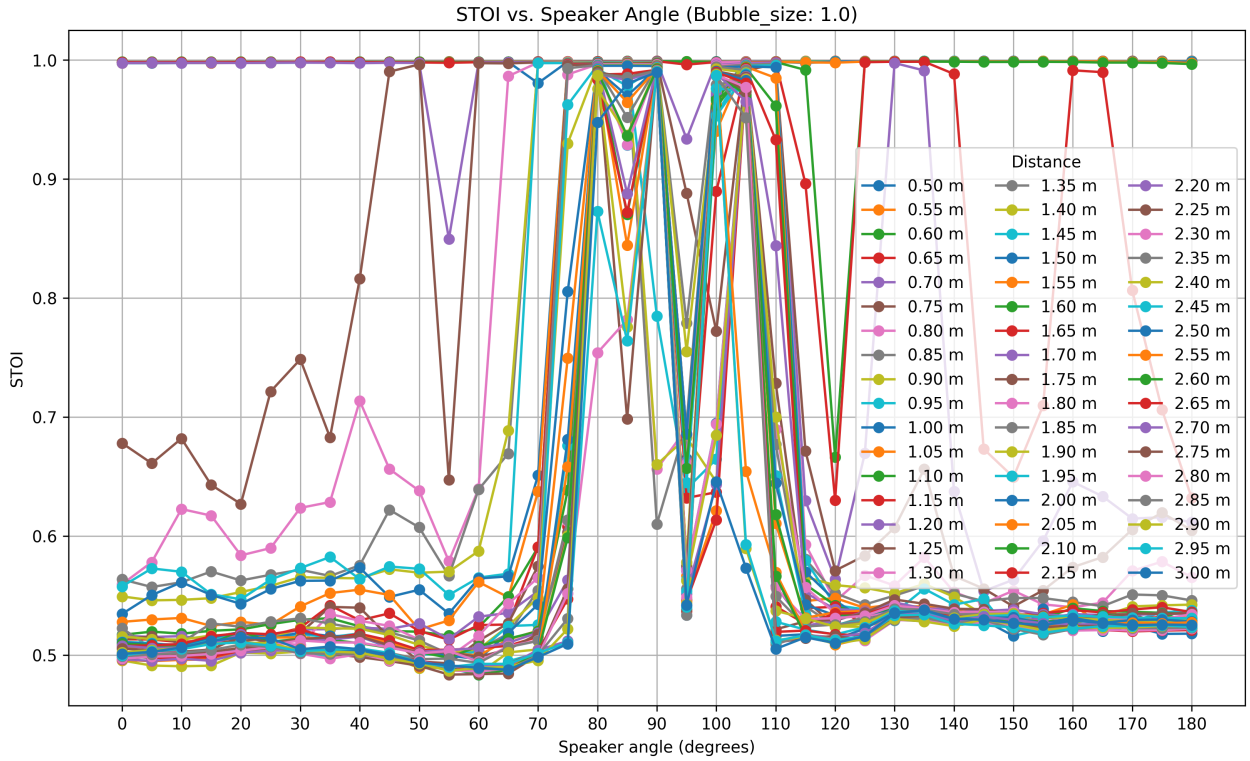


Figure 33: PESQ as a function of angle for all distances. (bubble size: 1.0)

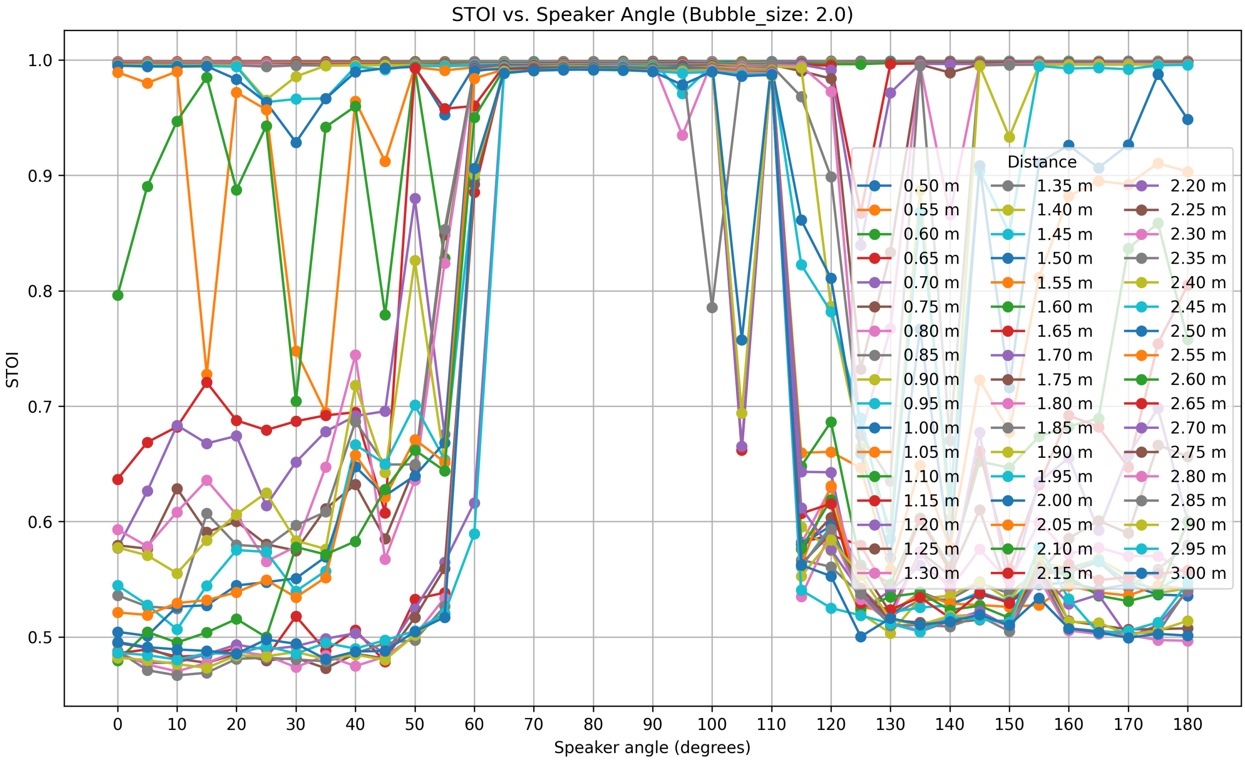


Figure 34: PESQ as a function of angle for all distances. (bubble size: 2.0)