2017-07-13

DTMF Audio Communication for Nao Robots

Kyle Poore
University of Miami, kyle@cs.miami.edu

UNIVERSITY OF MIAMI

DTMF AUDIO COMMUNICATION FOR NAO ROBOTS

By

Kyle Poore

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

August 2017

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

DTMF AUDIO COMMUNICATION FOR NAO ROBOTS

Kyle Poore

Approved:

Ubbo Visser, Ph.D. Associate Professor of Computer Science	Dilip Sarkar, Ph.D. Associate Professor of Computer Science
Mei-Ling Shyu, Ph.D. Professor of Electrical and Computer Engineering	Guilliermo Prado, Ph.D. Dean of the Graduate School

POORE, KYLE (M.S., Computer Science)
DTMF Audio Communication for NAO Robots (August 2017)

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Ubbo Visser.
No. of pages in text. (63)

We propose an alternative to Wi-Fi for robotic communication, as its increased use in a competition environment has lead to highly overlapping and interfering networks. This interference often causes unreliable transmission of data, which affects teams' ability to coordinate complex behaviors. Our method uses fixed length Dual Tone Multi Frequency (DTMF) messages and uses a basic packet structure designed to reduce data corruption as a result of noise. We conducted twelve different experiments varying the distance between robots and message format, as well as whether the robots are walking or sitting silently. Methods for scheduling messages to avoid crosstalk were also developed and tested. The results show that while this method appears to be sensitive to motor noise, room reverberation and multipath effects, it has very low data corruption rates, which makes it suitable for use in some applications.

Acknowledgements

I would like to thank, foremost, my advisor, Dr. Ubbo Visser for his many hours of advice and motivation, his patience and persistence, and overall, his guidance to keep me on track despite my numerous shortcomings and deviations. His desire to see me succeed, and his belief that I could succeed has at times seemed inconceivable, but even after a short visit with my advisor, I found that I could dive into my work with renewed confidence and excitement. Without Dr. Visser's guidance, this work simply could not exist.

Next, I must thank the members of my committee, Dr. Dilip Sarkar and Dr. Mei-Ling Shyu for their unquestioning willingness to help, and to give witness to my work on such short notice.

My colleagues in the Computer Science Department were instrumental in completing this work, and our many conversations of myriad topics has contributed greatly to my development in the field. I especially thank the members of the RoboCanes robotics team, and in particular I must express my gratitude for the contributions to this work made by Joseph Masterjohn and Dr. Andreas Seekircher.

Finally, I thank my family for their unwavering support and confidence in my ability to see this work through, and for enduring hours upon hours of my ramblings about research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For most autonomous robotics tasks involving multiple robots, some level of communication is crucial for coordinating complex actions to achieve a desired goal. A popular method of communication between robots (especially in RoboCup) is Wi-Fi. With hundreds of robots communicating on several independent overlapping Wi-Fi networks, interference is rampant, often causing network delays of several seconds, and in the worst cases, dropping entire connections altogether. It is therefore desirable to have an alternative method of communication, which, while possibly inferior to a strong Wi-Fi connection, is useful in situations where Wi-Fi has become unusable. Of course, the usefulness of an alternative communication method extends beyond just Wi-Fi. Any system which might encounter a catastrophic level of interference could benefit from having a backup communication scheme.

There are a few potential candidates to consider when looking for ways to transmit data from one robot to another. One possibility is using visual communication, controlling LEDs on the robot to send data through the visible or infrared spectrum.

This approach might actually have the greatest bandwidth between robots, as noise in the infrared spectrum tends to be relatively low, and the sensors would not be sensitive to vibrations of the robot; but these positive attributes come at a cost. The NAO robot, a humanoid robot engineered by Aldebaran (SoftBank) robotics, has infrared LEDs and sensors only on the front of it's head, meaning that any communicating pair of robots must keep their faces aligned towards each other, much like directional antennas. This also makes it difficult to communicate between more than one robot at a time. Panfir et al. (2013) were able to use these infrared sensors to communicate between two NAO robots in order to coordinate their actions for manipulating large objects. The major caveat to visual communication is that it is line-of-sight only; if one robot moves in front of another or turns so that its LEDs are no longer visible, the channel is interrupted. Other kinds of visual communication might even involve using the robot's physical movements to encode data in some way, and then recover the pattern using a camera. This is even more problematic than the infrared communication scheme as it would require complex vision algorithms for detecting signals, which are often not suitable when processing power is limited, not to mention the extremely low bandwidth afforded by articulating motors and body parts. Additionally, all of these methods suffer from the inability to transmit information among all robots simultaneously, which in the fast-paced domain of robotic soccer, is certainly desired.

Another, more favorable possibility for an alternative communication system is to encode data as audio, and broadcast it via loudspeaker. This does not suffer from the same problems as visual communication because sound waves travel around obstacles, and analyzing audio samples can be done fairly efficiently. The caveat to this approach is that there tends to be much more noise in ambient sound waves than there are in electromagnetic waves, especially when the microphones are attached to a moving and vibrating robot. The channel could also be negatively affected by multi-path effects; if the same message echoes off of a distant wall, the same message may arrive at a receiving robot at slightly different times, causing self-interference. The method would, however, enable robots to broadcast their state information simultaneously to all of the other robots, without placing too many constraints on robot positioning or alignment (the only requirement being that robots should not be too far away). While broadcasting allows one robot to transmit to many receiving robots, it does not permit multiple robots to share their states at the same time; messages must be carefully timed so that they do not collide (making both messages irrecoverable).

There are numerous methods of modulating waves to carry digital information. Some methods, such as Pase Shift Keying (PSK) and Frequency Shift Keying (FSK) operate by modulating intrinsic properties of a carrier wave. PSK switches between different phases of the same frequency to signal bits, and is therefore sensitive to sudden changes that affect the perceived phase of the wave, such as the distance between the transmitter and receiver or reflections of the signal off of nearby walls. Unfortunately, robots tend to be moving around constantly, which makes this method less suitable than others. FSK signals bits by switching between two (or more) frequencies, and while it is less sensitive to motion, it is sensitive to unpredictably noisy environments such as the internal noise of a robot. For our work, we choose a different method, Dual-Tone Multi-Frequency (DTMF) signalling, because it does not exhibit the same sensitivity to sudden phase shifts as PSK, but should also be less sensitive to noise, since it requires correlating multiple simultaneous frequencies to construct a single symbol. Our research shows that the DTMF method, while unreliable in certain conditions, can be used to broadcast messages with low probability of message corruption.

Much research has been devoted to audio signals featuring humanoid robots, especially in the past decade. Audio signals can be important sensory information as they can be used for various purposes, whether for the communication between multiple robots, the detection of audio cues in the environment or game events such as whistles, using the audio signals to improve self-localization, or even detecting problems with the robot's own hardware. A demonstration within the RoboCup Standard Platform League (SPL) in 2013 in Eindhoven by the team RoboEireann revealed how difficult it is to communicate between NAOs on the soccer field in a noisy environment.

This thesis is organized as follows: we discuss relevant work in the next chapter and describe our approach in Chapter 3. A more detailed description of the implementation process is described in Chapter 4. Our experimental set-up and the conducted robot tests is explained in Chapter 5. In Chapter 6 we briefly summarize our work in 6.1, discuss the pros and cons of our method and its results in Section 6.2, and conclude and outline future work in the remaining Section 6.3.

Chapter 2

Related Work

When consulting the literature, one finds a number of research papers that relate to our work. We include work that is not only related to communication, but also work that develops audio processing techniques for sensing the environment, since communication and sensing are inherently related. Saxena and Ng (2009) present a learning approach for the problem of estimating the incident angle of a sound using just one microphone not connected to a mobile robot. The experimental results show that their approach is able to accurately localize a wide range of sounds, such as human speech, dog barking, or a waterfall. Sound-source localization is an important function in robot audition. Most existing research investigates sound-source localization using static microphone arrays. Hu et al. (2011) propose a method that is able to simultaneously localize a mobile robot and in addition to an unknown number of multiple sound sources in the vicinity. The method is based on a combinatorial algorithm of Difference of Arrival (DOA) estimation and bearing-only Simultaneous Localization and Mapping (SLAM). Experimental results with an eight-channel microphone array on a wheeled robot show the effectiveness of the proposed method. Navigation is part of another study where the authors developed an audio-based robot navigation system for a rescue robot. It is developed using tetrahedral microphone array to guide a robot finding the target shouting for help in a rescue scenario (Sun et al. (2011)). The approach uses speech recognition technology and a Time DOA (TDOA) method. The authors claim that the system meets the desired outcome.

Another recent application of audio based communication has been developed by Sauer et al. (2014) for the purpose of facilitating control and adjustment of hearing aids by using high frequency audio signals sent from a smart phone. Their technique involves fully redundant transmission by sending the same control signals across multiple different frequencies, so that in the case of environmental noise masking one frequency, there is a higher chance that the control codes can still be recovered by the earpiece.

Mullins et al. (2012) describe a method for robot navigation in a swarm using Diffusion Limited aggregation (DLA). Their approach is inspired from the foraging behavior of Escherichia coli bacteria, which performs a gradient search based on the diffusion of nutrients in its environment. In order to apply this technique to their e-puck robots, they used audio signals as the diffusion medium; the greater the distance between two robots, the more diffuse the signals between them become. These signals were then used in a protocol designed to allow robots with a low battery to get assistance from neighboring robots in order to find a path back to a charging station. Due to hardware constraints, they performed large scale tests in a simulated audio environment. The authors offer this method as a proof of concept, showing that single-agent bacterial search and DLA-based collaborative search can be useful methods in a distributed system and that audio is an acceptable medium for these methods.

Audio communication is also useful in some underwater applications as shown by Wills et al. (2006), who developed an acoustic modem for low power communication of underwater seismic sensor networks. Their approach involves using an Atmel ATmega 128L microcontroller and other specialized hardware to send packets by modulating underwater acoustic waves using Frequency Shift Keying (FSK). In order to adhere to their low power constraint, they use a wake-up signal of 18kHz so that they do not need to run the controller continuously, but only when an incoming message is imminent.

Nakamura et al. (2012) present a framework of 3D Sound Source Localization (SSL) by Multiple Signal Classification achieving both high-resolution and real-time processing and apply it to a robot. They use a trilinear interpolation to their previously published Frequency- and Time-Domain Linear Interpolation. This extension generates transfer functions in 3D space with desired resolution by a small number of pre-measured 3D transfer functions in low resolution. Athanasopoulos et al. (2012) describe a TDOA-based sound-source localization method that successfully addresses the influence of a robot's shape on the sound-source localization. The evaluation is made with the humanoid robot NAO. The authors state that this approach allows to achieve reliable sound-source location.

ASIMO, the remarkable humanoid developed by HONDA also uses the auditory system for its tasks. An early paper from 2002 introduces the use of a commercial speech recognition and synthesis system on ASIMO. The authors state that the audio quality and intonation of voice need more work and that they are not yet satisfactory for use on the robot (Sakagami et al. (2002)). Okuno et al. (2011) present a later version of ASIMO's ability to use the auditory system for tasks at hand. They use the HARK open-source robot audition software (Nakadai et al. (2010)) and made experiments with speech and music. The authors claim that the active audition improves the localization of the robot with regard to the periphery.

Speech/dialogue based approaches for the NAO also exist. Kruijff-Korbayová et al. (2011), e.g., present a conversational system using an event-based approach for integrating a conversational Human-Robot Interaction (HRI) system. The approach has been instantiated on a NAO robot and is used as a test bed for investigating child-robot interaction. The authors come to the conclusion that the fully autonomous system is not yet mature enough for end-to-end usability evaluation.

More recent work, such as the paper by Wrede et al. (2013) suggest that significant background noise presented in a real HRI setting makes auditory tasks challenging. The authors introduced a conversational HRI dataset with a robot inducing interactive behavior with and between humans. The paper however does not discuss the auditory methods used in detail. We assume that the authors use the standard auditory recognition that comes with the NAO.

Carrara and Adams (2014) have shown that audio communication between machines is possible and practical for covert transmission of data in an office environment between computers in a manner which is imperceptible to humans. By using frequencies just above the human hearing range of about 20kHz - 20.5kHz they were able to transmit data at a rate of 140 bps, and were able to achieve 6.7 kbps when using audible frequencies between 500Hz and 18kHz.

Very recent work by Guri et al. (2016) shows how data can be exfiltrated from an air-gapped computer with no installed audio hardware by controlling the armature of the computer's hard drive. The disk's read/write head is oscillated to produce audible frequencies, which can be picked up by a nearby mobile device. This was achieved by examining the anatomy of the hard drive and analyzing its acoustical properties. The authors were able to transmit data at a rate of 180 bits per minute over a distance up to two meters away.

Kirovski and Malvar (2001) have developed a robust, covert communication method over a public audio channel for the purposes of watermarking. Since there are numerous methods of trying to remove a watermark from a digital file, the authors wanted to make a watermark embedding and detection method that could still recover the information after any of these distortions had been applied. They accomplished this by using both a spread-spectrum technique, and psycho-acoustic frequency masking.

All mentioned approaches and techniques so far differ from our approach (a) in the method used, (b) in the application of the audio recognition, and (c) the RoboCanes framework, a robotics framework developed by the RoboCanes robotic soccer team at the University of Miami. Here, all audio modules have been implemented from scratch and run within the framework's system loop.

Nguyen and Bushnell (2004) have suggested that acoustic communication using DTMF is, in general not, recommended for mobile robot applications due to the unreliability in acoustical integrity of the signal during transmission. While their transmission methods are similar, there are key differences in the recognition methods used: the frequencies used in their experiments are the generic set of frequencies used in telecommunications (which lie in a range prone to environmental noise). We sought to experiment with different sets of frequencies, chosen for specific empirical reasons, and to overcome signal degradation through robust filters.

Other uses of DTMF technology for robotic communication have been explored apart from acoustical environments. Srivastava et al. (2014) and Aswath et al. (2013) have used DTMF with mobile phones for long range control of robots. Each of these works uses a mobile phone directly connected to embedded hardware and the signal is transmitted through RF, not acoustically.

Chapter 3

Approach

3.1 General Approach

There are many methods for transferring data over analog media, but most are not suitable for communicating over the open air waves with a moving, noisy robot. Some of the challenges presented by this domain are relatively high noise levels (which can be unpredictable, especially in a robotics competition environment), interference from the internal vibrations of the robot (such as motors, fans, and stressed plastic), and unknown/changing distance between communicating robots. The latter of the above challenges means that using a Phase Shift Keying (PSK) method would likely perform poorly, since the data is embedded in the phase of the carrier wave. As a robot moves closer or further away from the sender, the distance – and thus the phase of a signal – will drift. This effect would be particularly prominent in systems which use high frequency carrier waves, as their wavelengths are very short and thus sensitive to small changes in distance. Frequency Shift Keying (FSK) handles the problem of the robots moving around since humanoid robots rarely reach speeds that would affect the frequencies via the Doppler effect, but they have a different weakness since single tones are modulated between two or more frequencies. Interference on one of those frequencies is likely to cause a data corruption error.

The Dual-Tone Multi-Frequency (DTMF) signaling method was developed at Bell Labs and was used in push-button telephones starting in 1963. It uses eight different frequencies, divided equally into two groups: four low pitched tones and four high pitched tones (Dodd (2002)). Symbols are transmitted by combining one frequency from the set of low frequency tones and one frequency from the set of high frequency tones using additive synthesis followed by a short period of silence and playing the resulting signal through a loudspeaker. The number of bits which can be transmitted through one symbol for a *generalized* DTMF scheme with a frequency groups, each with b frequencies is $\log_2 b^a$. Since we use the typical two groups of four frequencies each, we can send $\log_2 4^2 = 4$ bits at a time. Sending arbitrary bytes of data is convenient, since bytes can be broken into two four-bit codes, which allow a direct mapping to the sixteen possible symbols of DTMF.

This method is less prone to errors than FSK in noisy environments because if the probability of random noise coinciding with a chosen frequency f is $P(f)$, then probability of random noise emulating two chosen frequencies f_1 and f_2 simultaneously is $P(f_1) \cdot P(f_2)$. This assumes that the probability of random noise producing each of the two frequencies is independent of each other. However, frequencies in open-air environments are often not independent; while many sounds have a fundamental frequency, they are often accompanied by harmonic frequencies. Usually, the higher the multiple of the fundamental frequency a harmonic is, the lower the amplitude of that harmonic. This means that the most highly correlated frequencies in any noisy environment tend to be ratios whose numerator and denominator are small integers; since either f is the fundamental frequency, or f is some harmonic f_n of some other fundamental frequency $f_1 = \frac{1}{n} f_n$, and is related to the other harmonics of f_1 by $f_k = \frac{k}{n} f_n$.

Since the frequencies present in general noisy environments are not known beforehand, we cannot choose frequencies which avoid the harmonics present in the signals. We can, however choose frequencies whose ratios are not fractions with a small numerator and denominator. To achieve this, we divide our window length w_s by prime numbers p in the range of $2 \leq p \leq \frac{w_s}{2}$ in order to determine the appropriate number of samples per period for the low and high tones. The frequency is then calculated by dividing the sample rate by the number of samples per wavelength. This method of selecting frequencies ensures that they share only distant harmonics while making sure that there are at least two periods of each frequency per window.

Another limitation of which frequencies can be used is the Nyquist limit, which is the maximum representable frequency based on a certain sample rate (Nyquist (1928)). Nyquist himself was interested in the maximum transmission rate of telegraph messages The Nyquist limit is precisely half of the sample rate, since to represent an oscillation, at least one sample is needed for a crest, and another for a trough; each wavelength must therefore be composed of at least two samples. Since our sample rate is 48kHz, the Nyquist limit is 24kHz. This is why the upper limit for usable primes is $\frac{w_s}{2}$, as the buffer cannot represent frequencies with wavelengths that are shorter than two samples. As can be seen in Table 3.1, our highest frequency was less than 13kHz, because the microphones in our NAO robots have poor frequency response above that range.

Table 3.1 also shows the chosen prime numbers for dividing the window length, which is 2,400 samples, as well as the number of samples per wavelength, μ. Notice that the frequencies are well below the Nyquist limit, thus the chosen frequencies can be properly represented at the chosen sample rate. Also, with between 500 and 600 wavelengths per window, we should expect to get accurate readings for our magnitudes.

f	p	μ	Hz
l_1	503	4.771	10,060
l_2	521	4.606	10,420
l_3	541	4.436	10,820
l_4	557	4.309	11,140
h_1	571	4.203	11,420
h_2	587	4.089	11,740
h_3	599	4.007	11,980
h_4	607	3.954	12,140

Table 3.1: Frequencies Used for the Low and High Frequency Groups

3.2 DTMF Transmitter

The transmitter sends encoded data as DTMF symbols through open-air sound waves. The encoded data can easily be converted to playable waveforms by isolating four bit segments of the message (conveniently represented as a hexadecimal-digit) and then using table 3.2 to determine the appropriate frequency pair for that symbol. The two frequencies are combined using additive synthesis. Each waveform is pre-computed and simply indexed by each four bit section of the message. The gain of each DTMF tone is gently faded at the end as to avoid discontinuities which result in undesirable pops and clicks in the resulting audio signal resulting from waveforms ending with non-zero samples.

	h_1	h_2	h_3	h_4
l_1	0	1	2	3
l_2	4	5	6	7
l_3	8	9	A	B
l_4	C	D	E	F

Table 3.2: Frequency/Hexadecimal Encoding/Decoding Table
l_i and h_i indicate the low and high frequency groups.

3.3 DTMF Receiver

The receiver works very differently from the transmitter. This is mostly due to the fact that everything has to work in reverse: going from audio samples to decoded digital data. The process is illustrated in Fig 3.1. Recorded audio is passed in windows of samples through a series of Goertzel filters (Goertzel (1958)) to isolate the desired frequency magnitudes. This is done continuously on the new recorded samples, which generates a stream of vectors of frequency magnitudes (see Fig. 3.2). This stream of Goertzel responses is then analyzed in two more steps. First, we need to identify sequences in the magnitudes that have enough similarity with a message. If a sequence has been identified as message, the symbols are decoded and the robot has received a message.

The first step is to capture audio data from the robot's microphones. We acquire our audio samples from the Advanced Linux Sound Architecture (ALSA) (van de Pol et al. (n.d.)), and use a 2,400 sample sliding window, advancing the window by half

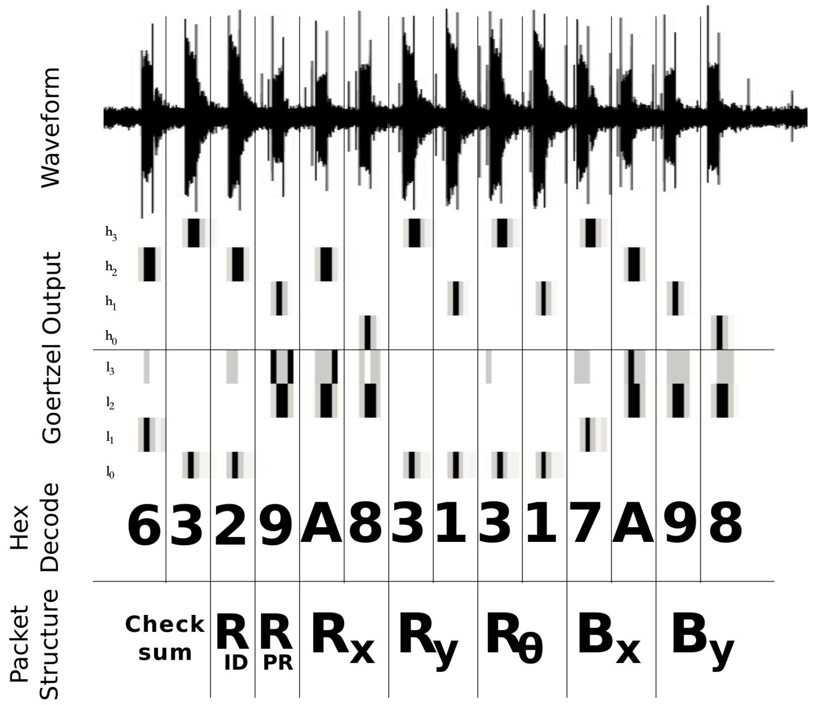


Figure 3.1: Illustration of the decoding process

of its length after each analysis step. With a sample rate of 48,000, this means we are effectively performing 40 analysis steps per second of audio data. The window length was chosen such that it would not overlap two separate DTMF symbols. The part of the waveform corresponding to a symbol is referred to as the *mark*, the other part of the waveform is called the *space*. We have chosen the lengths of the mark and space to each have durations of 100ms, so both take up 4,800 samples. Thus, the window lies either completely in *mark*, completely in *space*, or some combination of the two, but will never span the gap and include samples from unrelated symbols. Since the window advances by half of its width after each step, we are guaranteed that at least one window will be completely filled with samples from the *mark*.

Before measuring the frequencies, we pass the samples through a Hann windowing function to reduce aliasing, which is when two different frequencies, sampled periodically, yield the same magnitudes (Harris (1978)). Such signals would be indistinguishable to a Discrete Cosine Transform (DCT), so if one of these frequencies is present in an incoming signal, the DCT would measure higher magnitudes for both frequencies. Without a window function, the effective window is rectangular. Since the DCT needs to fit this non-periodic shape with periodic functions, many higher frequencies must be used together to approximate the rectangular shape, even though those frequencies might not exist in the raw continuous signal. The Hann window is very effective at reducing aliasing, which is why we chose it for this application.

The next step is to perform frequency analysis on the windowed data. It would be sufficient to perform a DCT on the sample window, but there are only eight frequencies used in the messages. Therefore, it is more efficient to use multiple passes of the Goertzel algorithm, one for each frequency. This leaves us with a vector of magnitudes for the eight frequencies.

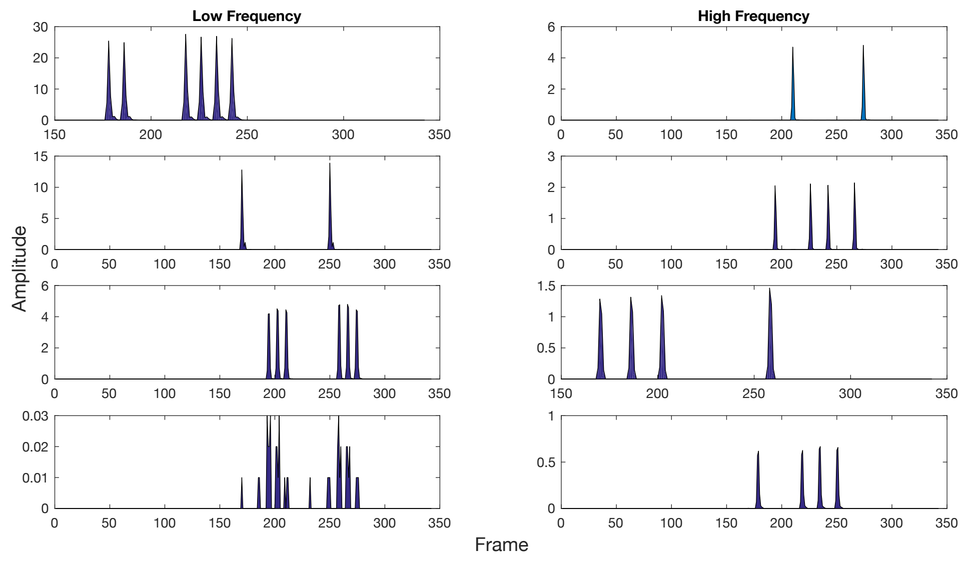


Figure 3.2: Magnitudes for individual frequencies

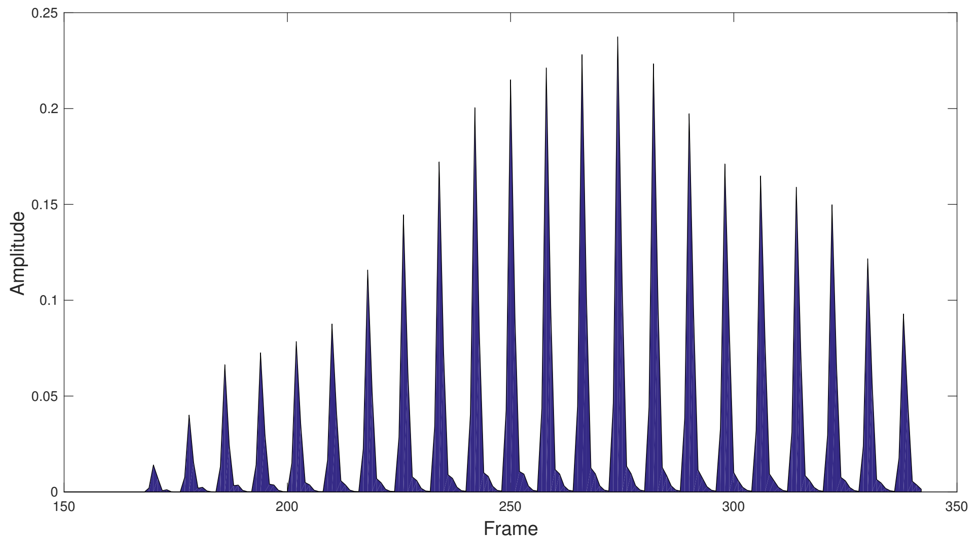


Figure 3.3: Peaks corresponding to output of the comb filter

We keep a history of the last several seconds of magnitudes and also keep a record of the sum of the magnitudes of each frequency over time. We use these frequency sums in order to detect the presence of a message before using the individual frequency magnitudes to decode the message. A comb filter with delay equal to the separation between DTMF symbols and length equal to the number of symbols per message is used to detect when a message is heard. The comb filter is a Finite Impulse Response filter which adds a delayed version of the signal to itself (Smith (2010)). This is a filter that has the strongest response if the sum of the magnitudes of the eight frequencies are high in exactly the positions selected by the comb filter.

Since the comb filter accumulates the magnitudes of the tones until the end of the message, when the comb filter begins to discard older tones, the sums develop a triangular pattern consisting of distinctive peaks in ascending and descending magnitudes as in Fig. 3.3. The overall shape formed by the peaks can be evaluated for symmetry. If one peak exceeds a defined threshold and the surrounding peaks are close enough to the triangular shape, the message is accepted.

The position of the maximum peak is exactly the end position of the message. We can find all measured frequency magnitudes at the message *marks* at the corresponding positions in the buffer relative to the end of the message. The appropriate magnitude vectors can be revisited and evaluated for the maximum low and high frequencies to decode the message.

Chapter 4

Implementation

4.1 Tools

Leading up to the full implementation of the communication system, a few tools are critical for prototyping and developing an audio based data channel. These include both external libraries and and tools which were created specifically for this work. Of primary importance in this regard, has been ALSA, which allows access to and control of any installed audio devices, including microphones and speakers. ALSA supports many advanced configurations of these devices, which unfortunately greatly complicates its interface. For this reason, we found it useful to create C++ wrapper classes to encapsulate the desired functionalities (reading microphone data and controlling the speakers) in order to avoid juxtaposing signal processing code with yet more complicated library calls. These wrapper classes were used throughout the project whenever control of an audio device was needed.

Another important tool is the "Fastest Fourier Transform in the West" (FFTW) library for frequency analysis of the incoming audio signal. FFTW is indeed currently the fastest implementation of the Fast Fourier Transform, which was desired due to the limited processing power of the NAO Robot, and the 48kHz sample rate required to clearly resolve the high frequency tones used for transmission.

The RoboCanes framework allowed for the integration of this communication system into several modules for the purpose of providing an backup for Wi-Fi. The framework is used as an organizational tool, but also serves the purpose of a platform for this system to run on. Indeed there is little need for a backup communication system if there isn't an agent to use it.

The tools discussed above had already contributed to the success of a previous project regarding the recognition of a whistle used to signal the beginning of play for robotic soccer matches in RoboCup's Standard Platform League. For this task, we used logistic regression, using the frequencies from FFTW as features. This worked very well, since whistles have many harmonic frequencies that allow them to be easily differentiated from other kinds of sounds. Following the success of that project, we decided that the same tools should be used to implement the communication system.

After gaining more familiarity with the problem of transmitting data over broadcast audio signals, we realized that the Goertzel Algorithm for computing individual DCT coefficients of a signal would be more efficient than a full FFT, since we only needed to analyze eight frequencies. The use of the Goertzel Algorithm for analyzing frequency amplitudes also improved the accuracy of the analysis, since it can compute the magnitude of a specific frequency band directly, instead of using the coefficient of the nearest FFT bucket, as would be required for the FFTW based approach.

4.2 A Standalone Program

Instead of developing the audio communication system directly on our robot's modular framework, we decided to start by creating a smaller stand-alone program designed specifically with audio processing in mind. This meant we could focus more on the issues specific to our goal, while being insulated from the extra complexity of developing and testing from within the RoboCanes framework. The resulting program was called ALSA_DTMF, combining the acronyms of the library used and the implemented method of transmission, Dual-Tone Multi-Frequency coding. ALSA_DTMF proved to be a quite versatile program, with various usage modes; it could behave as a transmitter and receiver, as well as numerous other functionalities such as recording and playing audio to and from files in the WAV format, or analyzing recorded audio files for messages.

The architecture of the ALSA_DTMF program was designed to have an input module and an output module. The input module's purpose was to provide the audio stream data; multiple implementations of this module were developed to be able to produce an audio signal from different sources: microphones, WAV files, and binary messages to be transmitted. The output module's purpose was to process the audio stream, and, similar to the input module, multiple implementations were made to cover range of functions, including playing the stream through the speakers, writing it to a file, or decoding the audio signal back into binary messages. Any pair of input and output modules can work together, allowing for numerous useful combinations. The program can behave as a transmitter by connecting the data input module with the audio output module, or as a receiver by using the microphone input module and the detector output module. A few other useful combinations exist, like the data input module paired with the detector output, which tests data recovery after converting it into an audio signal, but without the noise of broadcasting the signal over the air.

One problem arises from using these independent input and output modules though, and that is one of timing. The audio input module must repeatedly read data off of the audio device to prevent a buffer overrun, and similarly the audio output module must provide a signal to the speakers periodically to avoid a buffer underrun. Since the relative timings of these modules are independent of other modules which read from a file or analyze incoming signals, we decided that it would make sense to have the input and output modules run in separate threads. Using the producer/consumer model of concurrency, each module is given access to a common queue and a mutex which regulates which thread is allowed to modify the queue.

While some modules perform a fairly trivial task, such as reading from or writing data to a file, two modules are worth explaining further as they encapsulate the transmitter and receiver code. The former's purpose is to convert raw binary data into an audio signal that can be broadcast, and the latter's purpose is to perform the inverse: to reconstruct the original message from an incoming signal. However, before we can get into the inner workings of these modules, we must first take a look at some of the program's initialization.

When the program is started, it reads a configuration file to determine which frequencies to use (four “low” frequencies and four “high” frequencies), how long it should play each symbol (this is called the “mark”), and how much space to leave between them (appropriately named, “space”). For efficiency, the symbols are pre-computed in this initialization step, since computing each sample of each waveform requires calling the trigonometric sine function which would be too slow to perform in real time. A symbol is made from two pure sine waves, one with a frequency selected from the low set of frequencies, and the other selected from the set of high frequencies. These sine waves are then added together; this process is called additive synthesis, and is what allows us to perceive multiple tones being sounded at the same time. This is enough to produce an audible tone, but when the symbol terminates, and the speaker begins playing silence, a loud click will be heard. This is because the last sample of the symbol leaves the speaker cone in a position that is probably far away from where it will be when it is playing silence, causing the cone to rapidly move to the “zero” position. It is this rapid adjustment that causes the click. We wish to remove this click, as it can interfere with frequency analysis, and can also be quite annoying to humans. A simple solution is to multiply each sample by an attenuation factor that is linearly interpolated from 1.0 to 0.0 near the end of the tone. This way, no matter what amplitude the wave form terminates with, the sample values will approach 0 by the end of the tone. Listing 4.1 shows the code that generates the waveforms corresponding to each symbol.

```
void Common::generate_tone_buffer() {
  // for each pair of low and high frequencies
  for(int i = 0; i < num_low_freqs; i++) {
    for(int j = 0; j < num_high_freqs; j++) {
      float f1 = lower_freq[i];
      float f2 = higher_freq[j];
      // for each sample in the current tone buffer (leaving space for
         channels)
      int total_samples = MARK_SAMPLES * CHANNELS
      for(int k = 0, x = 0; k < total_samples; k += CHANNELS, x++) {
        // calculate the angle for a 1hz signal
        float unit_hz = 2 * M_PI * x / SAMPLE_RATE;
        // compute k_th sample, the sum of both signals at time x
        // amp_low and amp_high are adjustable magnitude coefficients for
           each frequency
        tone_buffer[i][j][k] = (short)amp_low [i] * sin(f1 * unit_hz);
        tone_buffer[i][j][k] += (short)amp_high[j] * sin(f2 * unit_hz);
        // fade out samples near the end of the buffer
        float dist_to_end = total_samples - x;
        float fade = 1.0;
        if(dist_to_end < FADE_LENGTH){
          fade = dist_to_end/FADE_LENGTH;
        }
        tone_buffer[i][j][k] *= fade;
        // copy k_th sample to the remaining channels
        for(int l = k + 1; l < k + CHANNELS; l++) {
          tone_buffer[i][j][l] = tone_buffer[i][j][k];
        }
      }
    }
  }
}
```

Listing 4.1 Code that generates each of the 16 DTMF symbols

Now that we have explained how the waveforms for each symbol are initialized, the signal generation module is actually quite simple to implement. Since there are a total of sixteen DTMF symbols, each symbol represents exactly four bits of information. The module receives a byte of data from its input stream and breaks this byte into two four bit codes. The first code (with the most significant bits of the original byte) is used to index into the precomputed symbol buffer so that each code maps to a unique DTMF symbol. Following this first symbol is a period of silence before the whole process is repeated for the second half of the byte. This is repeated for every byte in the input stream until there are no more data to send.

The recovery of this signal is a much more difficult process, as we must filter out noise, obtain the amplitudes of the desired frequencies, detect impulses in those frequencies over time and correlate them with each other to determine the alignment of the message, and finally reconstruct the original stream of data. In the following paragraphs we will justify our chosen solutions to these challenges.

Filtering noise and performing frequency analysis are closely related. Since we wish to focus on the changes of specific frequencies over time, we could simply use our implementation of the Goertzel algorithm to extract that frequency's magnitude, but we will then have problems with what is called Aliasing. Aliasing is a problem that arises when sampling a signal; if two specific frequencies are sampled periodically, the sampled signals may appear to be identical, even though they are quite different in reality. Thankfully, there are methods to reduce aliasing, such as using a Hann window instead of the default rectangular window. A Hann window reduces aliasing at the expense of local frequency resolution; that is, the magnitudes of frequencies nearby to our desired pitch will leak into its measured magnitude, while frequencies that are far from the selected frequency will tend not to interfere. This reduces aliasing because signals which alias to the same sample values tend to be distant from each other in the frequency domain.

Having passed the time domain samples through the Hann window, it is now time to perform the frequency analysis, which we accomplish by using the aforementioned Goertzel algorithm. We show an unoptimized version of our implementation of the Goertzel algorithm in Listing 4.2. The Goertzel algorithm uses a type of Infinite Impulse Response filter, which means that all previously encountered samples (within the current frame of analysis) play at least some role in the filter's output, as opposed to a Finite Impulse Response filter, in which only a fixed number of samples contribute to the filter's output. The algorithm only needs to access each sample once, making it much more efficient than a full DCT. In our implementation of the Goertzel algorithm we attempt to further improve performance in practice by unrolling its loop by a factor of 50 to reduce the overhead involved in incrementing index variables and testing exit conditions.

The Goertzel filter is applied to a set of samples, produces the magnitude of the desired frequency, and then is applied again with the window shifted forward in time by half of the window length. This is to ensure some continuity in the output magnitudes and also improves accuracy, since parts of the signal near the edges of the window are rather severely attenuated by the Hann window function. This is because samples near the end of the window in one frame of analysis end up in the middle of the window for the next frame once the window has been advanced.

```
double Detector::goertzel(TwoBuffer &buf, double frequency, int x, int
    length) {
  // variables for the Infinite Impulse Response filter (IIR)
  double q0, q1, q2;
  q0 = q1 = q2 = 0;
  // coefficient derived from frequency
  double coef = 2 * cos((PI_2 / SAMPLE_RATE) * frequency);

  double scale = messageLength / 2.0;
  double min = 100000;
  double max = -100000;
  double val;
  // find the min and max sample magnitudes in the window
  // these are used to transform the range of sample magnitudes
  // to between 0.0 and 1.0
  for(int i = x; i < x + length; i++) {
    val = buf.get(i * CHANNELS);
    min = (val < min ? val : min);
    max = (val > max ? val : max);
  }
  // pass the IIR filter over the sample buffer
  for(int i = x; i < x + length; i++) {
    q0 = ((buf.get(i * CHANNELS) - min) / (max - min)) + (coef * q1) - q2;
    q2 = q1;
    q1 = q0;
  }
  // final calculation of magnitude
  return ((q1*q1 + q2*q2 - q1*q2*coef) / (scale*scale));
}
```

Listing 4.2 Our unoptimized implementation of Goertzel's algorithm

The detection of impulses in the incoming signal is performed by dividing the frequency magnitudes obtained from Goertzel's algorithm by the maximum magnitude for that frequency in the last ten message lengths. This is done because the magnitudes of the different frequencies often have different scales, and assuming that the loudest magnitude in the buffer was caused by a DTMF tone, magnitudes near that maximum will result in a value close to (and at most) 1.0, indicating that it is also probably a DTMF tone. This data can be more easily thresholded, since all values are on a scale from 0 to 1, though a loud impulse of noise on one of the frequencies, will likely reduce its sensitivity temporarily. These calculated values are summed up across each frequency and stored in another ring-buffer for the purpose of discovering the message alignment.

In order to obtain the message alignment, we pass this ring-buffer through a comb filter, summing values separated by the amount of time between symbols for the number of symbols expected in a message. As a message comes into alignment with the filter, the filter output higher and higher values, until it reaches a maximum when the message and filter are optimally aligned. Subsequently, the filter output reduces gradually as the message moves further from its proper alignment. The output of this filter as it passes over a message therefore takes a triangular shape. We detect the peak of this triangular shape by measuring the symmetry of the surrounding values. The symmetry of the values is measured by first taking a discrete derivative of the last five samples, and negating the second half of the resulting four values. Then the error between corresponding symmetric elements (values that are opposite each other across the center of the triangle) is obtained by taking the sum of the absolute values of their differences. We assume that the error in symmetry is Gaussian, so we pass the error value through a Gaussian filter with a standard deviation of 0.5. We then multiply this value by the product of all of the derivative values. This final number is somewhat of a confidence factor; if the symmetry error is high, then the Gaussian function will give a very low value, ensuring the current alignment will not be considered. However, if the values do not take a triangular shape and instead are flat (with very little deviation), the Gaussian filter is likely to give a high value, which is why we multiply by the product of the differences of the magnitudes; if the differences are small, the confidence value will also be very low. When a peak is detected in the last three confidence values, and the peak is above a threshold of 0.25, then we consider the message to be properly aligned.

Once the message alignment has been determined, the only remaining step is to recover the encoded data into binary. For each set of magnitudes that align with a symbol in the message, the two largest values and their corresponding frequencies are determined. If the frequencies correspond to a valid DTMF tone combination – that is, one of the lower frequencies combined with one of the upper frequencies – then the four bit value is placed in the next sequential position in the output buffer. In the case of the ALSA_DTMF program, this output buffer is simply printed to the standard output, but for this method to be useful to RoboCanes, the output needs to be decoded so that it can update different representations and beliefs. Similarly, the data to be transmitted via the audio signal must be determined from within the agent.

4.3 Integration into the RoboCanes Framework

Integrating the modules from ALSA_DTMF into the RoboCanes framework proved to have its own challenges. One problem is that if multiple robots attempt to transmit information at the same time, then the messages will interfere with each other, and often neither message will be received by the other robots. It is important to mitigate this source of interference, so we have tried two scheduling strategies. The first approach does not assume time synchronization between all robots. Instead each robot calculates its next transmission time based on the ID of the most recent message's sender. The second strategy assigns each robot a dedicated time slot in which only that robot is allowed to transmit. This second strategy does require time synchronization before it can work properly, but it does ensure that the robots will not transmit at the same time.

The first scheduling strategy defines an transmission queue ordered by the robot's ID, and selects an initial robot as the first transmitter. After a short delay to ensure all robots have initialized, the first robot transmits a message. Ideally, all of the other robots hear this transmission and, based on the ID of the sending robot, can calculate the expected time before it is their turn to transmit. For example, if the third robot receives a message from the first robot in the schedule, it knows that it needs to wait for the second robot's transmission before it can safely send a message. Even if the third robot does not receive the second robot's transmission, it can still transmit at the appropriate time. This process continues through the list of robots and wraps back around to the first robot once all other robots have transmitted. One of the benefits to this solution is that it takes little setup; there is not synchronization required. Though the robots sometimes miss messages or transmit at inappropriate times, the schedule tends to self correct. Often, enough robots are able to hear enough correlated messages to establish a new schedule if the old schedule has been violated.

The second and more successful scheduling method is actually simpler, but requires that the robots' clocks be synchronized before playing. In this case, the robots can directly calculate a shared transmission schedule, again based on the robot's ID, but even if a robot's microphones are inoperative, rendering it unable to receive a signal, it can still participate in communication, since it does not need knowledge of other robots' messages. The obvious caveat to this approach is that if the synchronization step is not performed before a game, then there is almost no hope for a properly functioning shared communication channel.

For the purposes of testing, a third scheduling mode was developed which simply uses one robot to continually transmit messages separated by a short period of silence, while all other robots are only listening for messages. This allowed for more controlled experiments; the transmitting robot would remain stationary and the experimental variations (such as varying the distance, or the activity level) could be applied to just the receiving robot.

4.4 Room Acoustics Simulation

Once we had performed our first experiments, we sought to explain some of the phenomena we observed. Particularly, we observed an unexpected relationship between the distance to the receiver and the number of dropped messages. It was expected that the furthest robot would have the most difficulty receiving messages, yet the experiment indicated that the mid-range robot experienced the most error. We hypothesized that this was due to the echoing and reverberation of sounds throughout the room, but without a multitude of microphones or robots, this hypothesis would be difficult to confirm. The solution we found was to create a (rough) 2-dimensional simulation of the testing conditions.

The simulation consists of 800 grid points initialized to small random values, a sound source (a “speaker”) and three “microphones”. The geometry of the room was modeled by setting grid points that occur within obstacles and walls to zero; these are the boundary conditions. The speaker and microphones were placed in the same positions which were used for the experiments. The simulation was carried out by applying the discretized wave equation to each grid point for every time step, enforcing the boundary conditions and then repeating. The speaker works by driving one of the grid points with an oscillation pattern similar to those produced by the robots.

The simulation allows us to observe the kinds of interactions of the transmitted waves and the geometry of the room, and also the effects of these interactions on the signals that were picked up by the microphones. That said, the simulation does not attempt to model the room’s acoustics exactly; such a simulation would need to be extremely complex, modeling the space in three spatial dimensions, as well as the energy absorption and materials of the walls, floor, and ceiling. Furthermore, the acoustical properties of the physical robot are not considered, as they are too complex and variable to model with any accuracy.

Chapter 5

Experiments and Results

We have conducted our experiments to determine the effectiveness of DTMF as a communication method between robots that are different distances apart (Fig. 5.1), while also measuring the effect of internal noise while the receiving robot is walking. For each of the stationary experiments, 500 fixed length messages were sent, 7 bytes each. For the experiments involving a walking robot, only 200 messages were sent, due to limitations of battery and motor temperature. For each of the distances separating the robots and the different activities of the robots, two kinds of messages were tested. The first method uses random blocks of data that span the entire message length, and the second method uses a short header, including a checksum of the message, the transmitting robot's ID, and the ID of the robot from which the transmitter has last received, followed by random data.

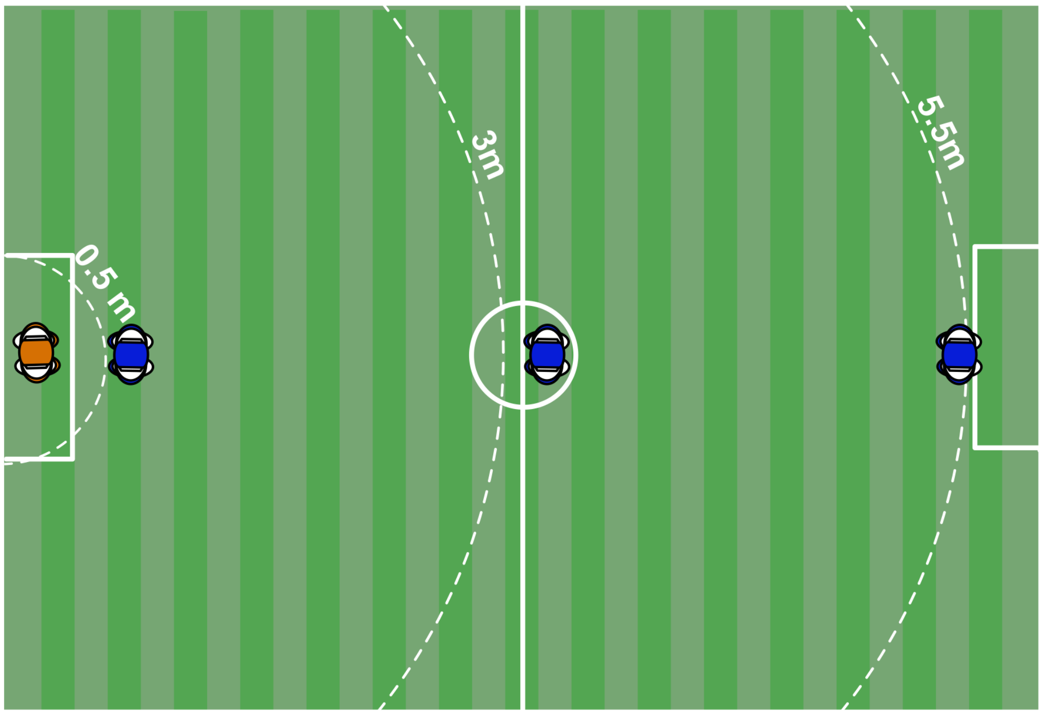


Figure 5.1: Robot placement on a 4 $\times$ 6 meter field

For both block and packet messages, random data were generated by the transmitting robot and sent via the robot's loudspeakers. The same data were then saved to a file for later comparison. The receiving robot, upon obtaining a block message, records it in a file, however, upon receiving a packet message, the packet is verified using its checksum. If the packet is valid, only the data portion of the packet is written to a file; otherwise the entire packet is discarded, with none of its contents being written. After each test, the message files are copied from the robots for analysis. This experimental set-up is shown in Figure 5.2. The analysis commonly performed on communication channels is the Hamming distance between the sent data and the received data. However, this metric is inappropriate in this case; it does not account for errors involving insertions or deletions, only substitutions. This is a concern because there is a relatively high probability of dropping messages. If the Hamming distance were to be used, the data would become misaligned after the first message drop, resulting in incorrect error rates. Besides that, the Hamming distance is not defined for data of different length. For these reasons we use the Levenshtein distance (Navarro (2001)), also known as the edit distance. This metric allows for measuring error due to the insertion, deletion, and substitution of symbols, as it counts the minimum number of such edits to transform one string into another. Here it is defined in a recursive form for two strings a and b as follows:

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where i and j are prefix lengths of a and b respectively.

Because the Levenshtein distance measures the minimum number of insertions, deletions, or substitutions needed to convert one string into another, we can also count the number of each kind of error. This information can give valuable insight as to the possible causes for different kinds of errors. For example, if the majority of errors were of the substitution variety, then one conclusion might be that there is a great deal of perplexity between symbols, meaning that they are difficult to distinguish. Similarly, a great deal of insertions might indicate that the system overly sensitive: trying to decode a signal when none is present. If the majority of errors are deletions, then it would seem likely that the system has difficulty detecting the presence of the signal, thus ignoring incoming messages.

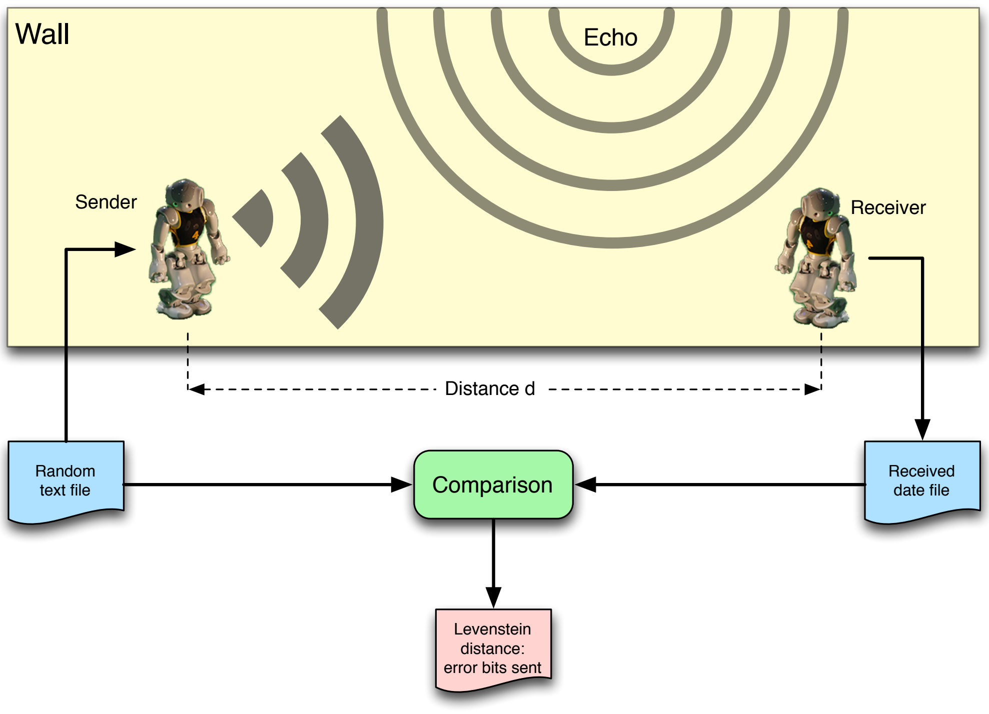


Figure 5.2: Experimental setup

For the first round of experiments, both robots were kept inactive to prevent the internal noise of motors from interfering with the signals. The transmitting robot was placed in the keeper's position on the field (between the goal posts). The receiving robot was then placed 0.5 meters (on the penalty box) in front of the transmitting robot, so that the two robots were facing each other. In this position, 500 fixed length randomized messages were sent from the transmitting robot to the receiving robot twice; once for the random block method and once for the random packet method (note that for the random packet method, the header is not randomized).

Set	Distance	Mode	Bits Sent	Error Bits	Error Rate
1	0.5	block	28,000	2,392	0.0854
2	0.5	packet	20,000	880	0.0440
3	3.0	block	28,392	5,948	0.2094
4	3.0	packet	20,000	10,200	0.5100
5	5.5	block	28,000	5,759	0.2056
6	5.5	packet	20,000	2,338	0.1169

Table 5.1: Results for Silent Robots

The above procedure was then repeated for distances of 3 meters (midfield) and 5.5 meters (opponent's penalty box) as shown in figure 5.1. The results of these tests can be seen in table 5.1.

For the second round of experiments, in order to simulate game-like conditions, the receiving robot was made to walk in such a way that it's average position remains at distance d from the transmitting robot, while all other variables were kept the same as in the above experiments. These results can be seen in table 5.2. We can see from these results, presented in Figure 5.3, that walking has the most detrimental effect on the channel of all of the experimental variables.

Set	Distance	Mode	Bits Sent	Error Bits	Error Rate
7	0.5	block	11,200	6,440	0.5750
8	0.5	packet	8,080	4,680	0.5792
9	3.0	block	11,200	7,056	0.6300
10	3.0	packet	8,000	5,080	0.6350
11	5.5	block	11,200	6,552	0.5850
12	5.5	packet	8,000	4,400	0.5500

Table 5.2: Results for Walking Robots

Since the Levenshtein distance counts three different types of errors (or edits), and it seems reasonable enough that these different kinds of errors might be caused by different acoustical phenomena, we sought to count the occurrences of each type of error, hoping that this would give additional insight. Though the number of each type of error is not directly recoverable from the data, the Levenshtein distance algorithm can be modified to output the minimal set of edits to produce the received message from the error-free original.

We can clearly see in Tables 5.3 and 5.4 that the vast majority of errors through the channel are deletions, with the occasional insertion or substitution. Table 5.4 takes this to an extreme, indicating that for all of the walking trials there was not a single symbol insertion or substitution; all errors for those trials were deletions. This strongly indicates that the difficulty for this approach is primarily in detecting the presence of a message; if a message is not detected, then all of the symbols count as deletions.

Set	**Distance**	**Mode**	**Substitutions**	**Insertions**	**Deletions**
1	0.5	block	0.0334	0.273	0.694
2	0.5	packet	0.0	0.0	1.00
3	3.0	block	0.00168	0.000168	0.998
4	3.0	packet	0.0	0.0	1.00
5	5.5	block	0.0366	0.0295	0.932
6	5.5	packet	0.00342	0.00214	0.994

Table 5.3: Errors by Type for Silent Robots

Set	Distance	Mode	Substitutions	Insertions	Deletions
7	0.5	block	0.0	0.0	1.00
8	0.5	packet	0.0	0.0	1.00
9	3.0	block	0.0	0.0	1.00
10	3.0	packet	0.0	0.0	1.00
11	5.5	block	0.0	0.0	1.00
12	5.5	packet	0.0	0.0	1.00

Table 5.4: Errors by Type for Walking Robots

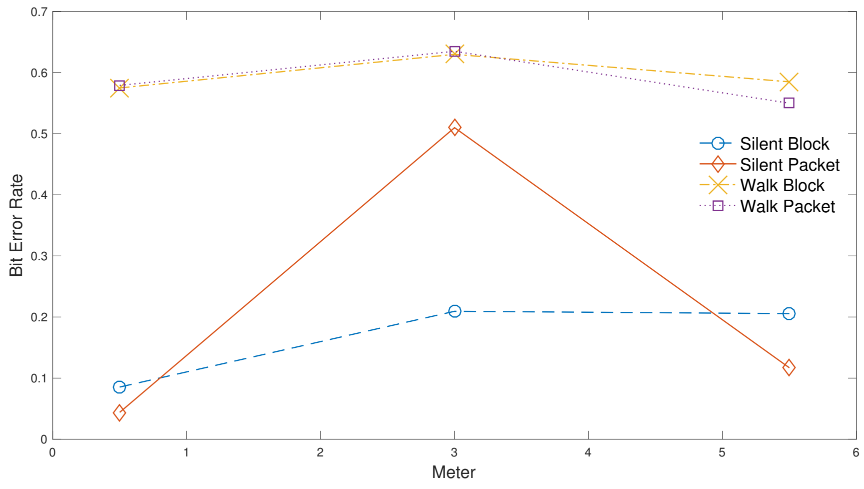


Figure 5.3: Bits of error per bits sent vs. transmission distance

Figure 5.3 visualizes the error curve as a function of distance between robots. An interesting observation is that the 3 meter trial yielded poorer results than the 5.5 meter trial. We expected that the channel's error should be positively correlated with the transmission distance. Figure 5.3 clearly shows that there must be some other phenomenon which affects the channel quality more than distance does. It seemed likely that such behavior could be explained by reverberation and echo throughout the room, but this hypothesis is difficult to test without a lot of expensive equipment.

We decided to try simulating the acoustics of the testing conditions, using a discretized version of the 2-dimensional wave equation:

$$u_{i,j}^{l+1} = 2u_{i,j}^{l} - u_{i,j}^{l-1} + r^2(u_{i-1,j}^{l} + u_{i+1,j}^{l} + u_{i,j-1}^{l} + \ u_{i,j+1} - 4u_{i,j}^{l}) \tag{5.1}$$

$$= 2u_{i,j}^{l} - u_{i,j}^{l-1} + [\Delta u]_{i,j}^{l} \tag{5.2}$$

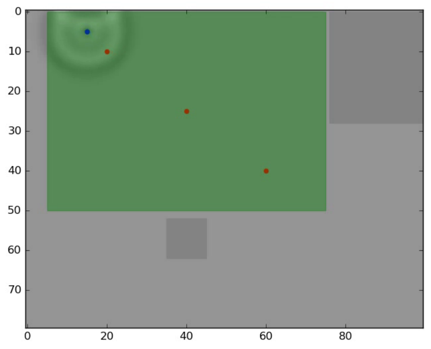
where $u_{i,j}^{l}$ denotes the scalar value at grid point i, j at time l. However this equation does not model the decay in the signal as it passes through the air; this is an important property for the simulation, otherwise the system would never lose any of its energy as more and more energy is pumped in via the oscillating speaker. To account for decay of a wave, we use a coefficient λ on the (linearly approximated) derivative:

$$u_{i,j}^{l+1} = u_{i,j}^{l} + \lambda(u_{i,j}^{l} - u_{i,j}^{l-1}) + [\Delta u]_{i,j}^{l} \tag{5.3}$$

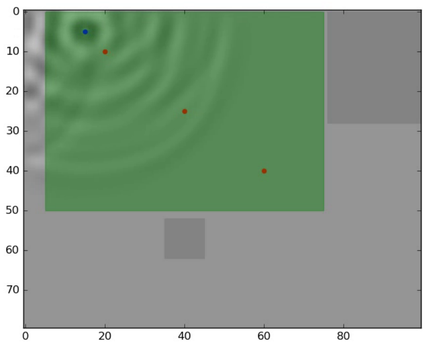
so that if we set λ to some value close to 1 such as 0.9, we observe a reasonable decay of the wave as it propagates about the room. Note that this assumes that reflections absorb none of the wave's energy, which is not expected in practice, but for the purposes of this simulation, it is close enough to the highly reflective and hard surfaced walls found in our lab. If we wanted to model the reflections more accurately, we would need to carefully choose a different set of boundary conditions (which are more difficult to enforce than simply setting the boundaries to 0), but we did not deem it to be necessary for our purposes. The simulation is conducted over a rectangular lattice with dimensions 80×100 over a period of 800 time steps. The walls of the lab are simulated by imposing that $u = 0$ as the boundary conditions, causing waves to be reflected back into the room. Similarly, internal columns and other geometries are modeled by setting $u = 0$ for all grid points that lie within these obstacles. Speakers can be simulated by oscillating one grid point of the simulation over time, and microphones are similarly modeled by simply recording the scalar values of a single grid point over time. We did not attempt to model the ambient noises in the room, such as the sounds produced by the air vents in the room, nor did we try to model the internal noise of the receiving robot. Therefore the test is most comparable to the trials involving a silent, seated robot.

We chose to simulate the interactions of two different symbols, one after the other, to observe how reverberation might affect the signal. Figures 5.4 and 5.5 shows a selection of frames from the simulation, where the blue dot represents the position of the speaker, the three red dots represent the positions of the microphones on the field, and the gray rectangles represent the shapes and positions of walls and a column in the testing environment. For a smooth video playback of the simulation, please visit the following link: video. We can thus directly observe the waves as they pass over the microphones, bounce off the walls, and interfere with other parts of the signal. We can clearly see significant reverberation throughout the period of silence between symbols and into the next symbol.

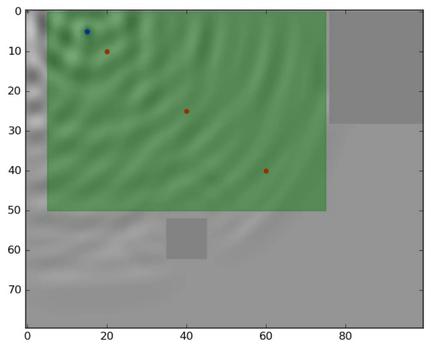
We replicated our testing conditions within this simulation, placing a speaker in the position of the transmitting robot, and microphones in the positions of the receiving robots. We can observe several useful properties from the simulated microphone readings (shown in Figure 5.6), including the propagation delay between microphones and the interference caused by the symbol bouncing off of the walls. It becomes diffi-



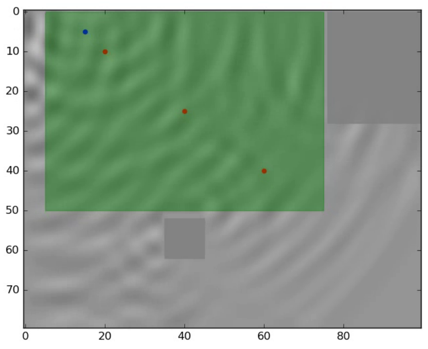
(a) Frame 60



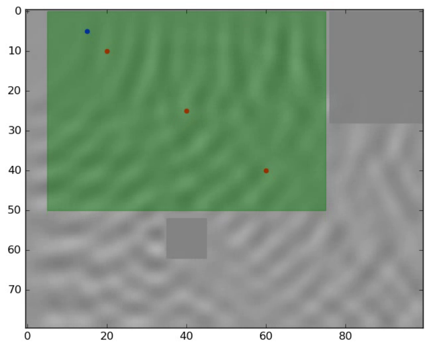
(b) Frame 120



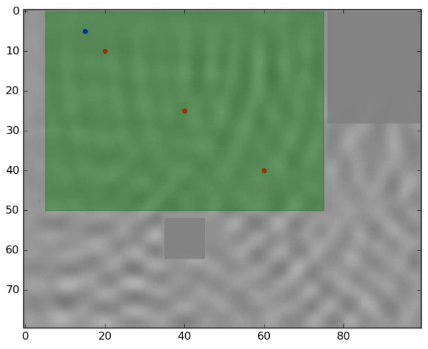
(c) Frame 180



(d) Frame 240

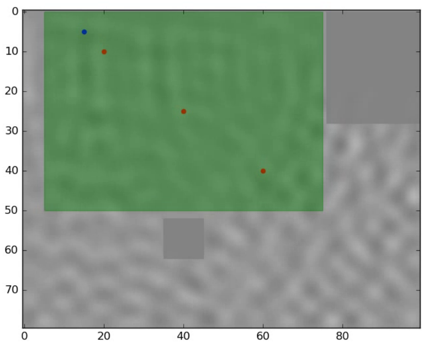


(e) Frame 300

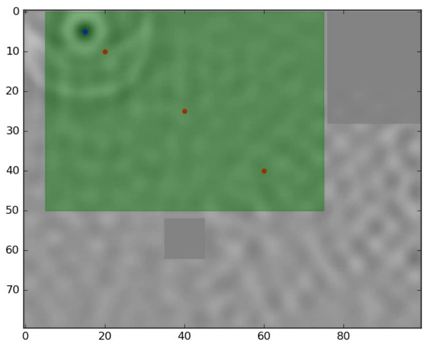


(f) Frame 360

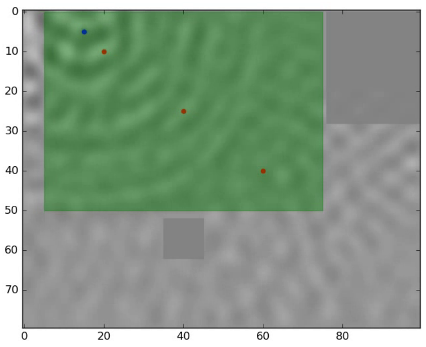
Figure 5.4: Visualization of Room Acoustics Simulation



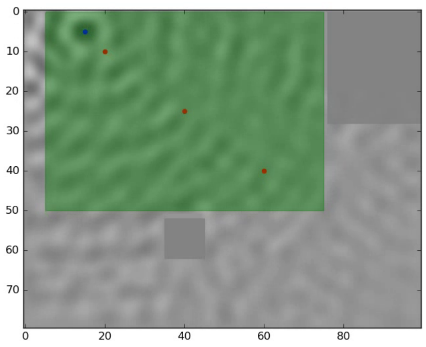
(a) Frame 420



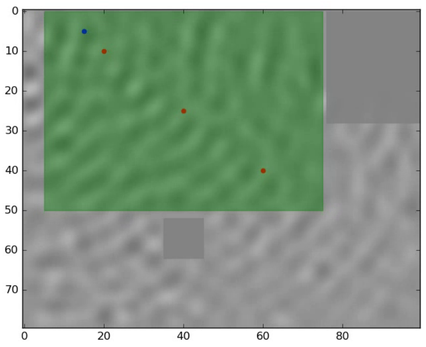
(b) Frame 480



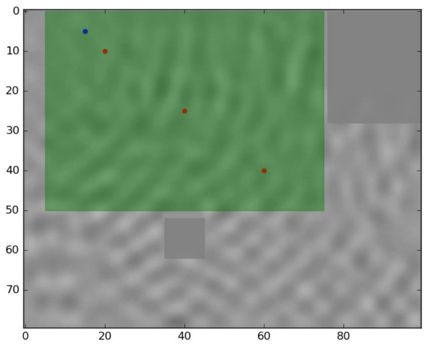
(c) Frame 540



(d) Frame 600



(e) Frame 660



(f) Frame 720

Figure 5.5: Visualization of Room Acoustics Simulation (continued)

cult to distinguish between the reverberation and the second symbol, which at least partly confirms the notion that reverberation is the main source of error apart from the internal noise of walking.

Figure 5.4, plot (a) shows the environment just as the first symbol is passing through the nearest microphone. We can see that even as early as plot (b), there is significant interference as the wave bounces off of the nearby walls, though the part of the wave now passing over the middle microphone appears undisturbed. By the time the signal has reached the third and most distant microphone in plot (c), some of the distortion arising from early collisions with the north wall seem to be affecting the signal near that microphone's location. In Figure 5.4, plot (d), we can see waves propagating radially from the column just south of the field; while in the same plot, the speaker has finished transmitting the first symbol. The remaining plots in Figure 5.4 and plot (a) of Figure 5.5 show the reflections of the waves without additional input from the speaker.

With plot (b) of Figure 5.5 we see that the speaker has already begun transmitting the second symbol, but this time, there are significant echoes of the previous symbol still prevalent throughout the room. These waves must necessarily interfere with the new symbol since they are made up of the same set of frequencies. The frequencies used by the two symbols need not be the same for interference to take place, since it is the relative magnitudes of all of the frequencies that affects the recovery of the signal. As the waves of the new symbol continue to propagate, the pattern of waves is far more obscure and less predictable than the first symbol which started with a quiet room. The results of this simulation seems to indicate that the effects of reverberation are highly nonlinear (especially in an enclosed room), and that it seems plausible that this could be a factor in the results we observed from our physical experiments.

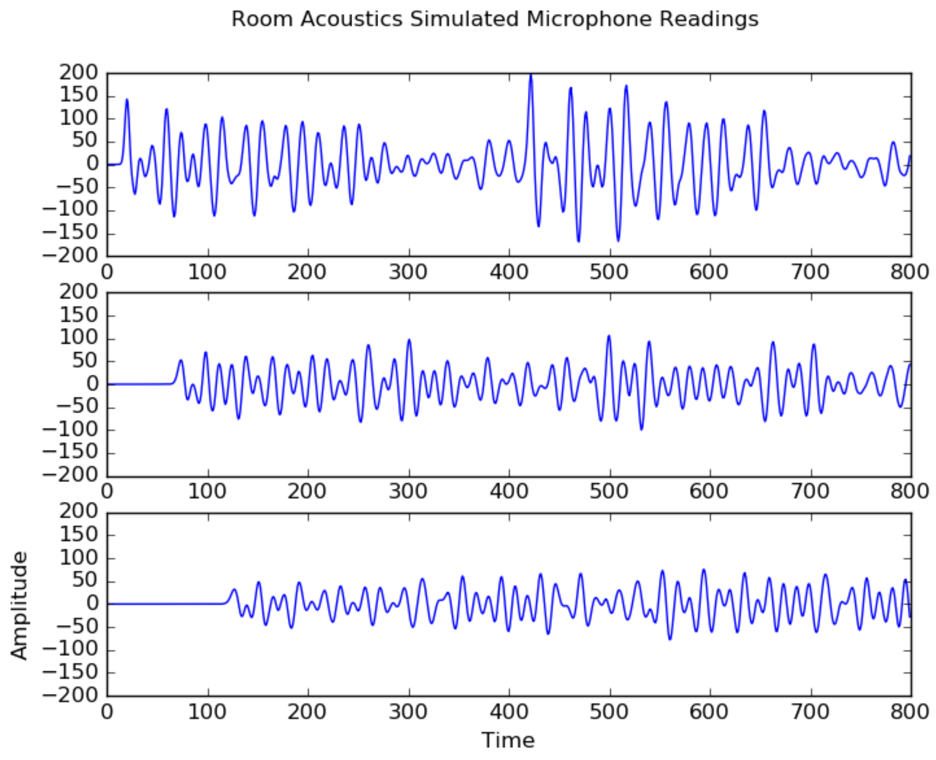


Figure 5.6: These are the simulated microphone readings for the distances of 0.5, 3.0 and 5.5 meters respectively.

One method of measuring the usefulness of the channel is to measure the number of corrupted messages (messages which contained at least one error) compared to the number of messages recovered in total. Figure 5.7 shows a plot of $M_{error}/M_{received}$ for each of the experimental trials. We can see from this data that very few successfully transmitted messages contain any errors at all, and when the packet header is used, the remaining corrupted messages can be filtered out almost completely. This shows that especially with the use of the packet header, a received message is actually quite reliable, confirming that the majority of error through the channel is not due to

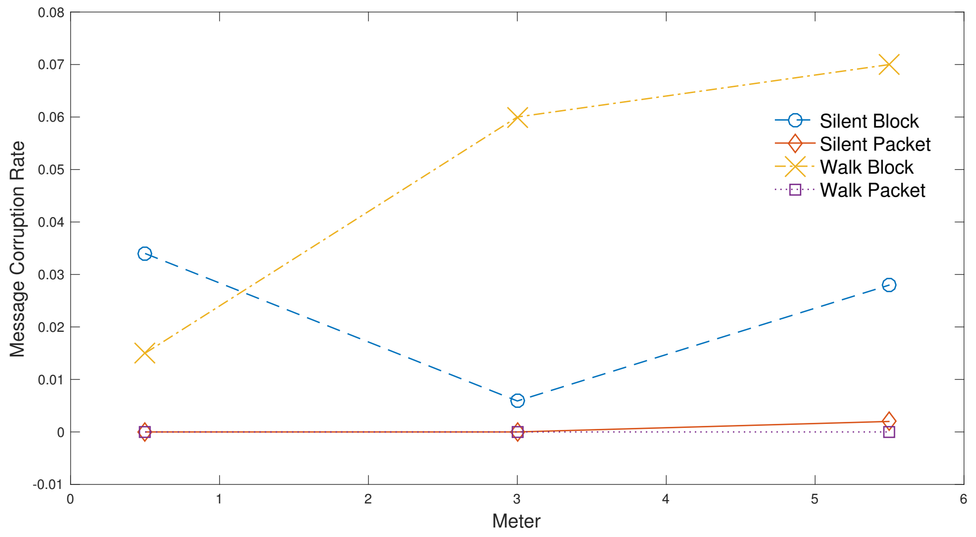


Figure 5.7: Corrupted messages per message sent vs. transmission distance

corrupted symbols, but rather a difficulty in separating a message from the noise.

Comparing Figure 5.7 to Figure 5.3, we see that the high transmission error rates observed at the three meter mark for silent robots (sending packets) is not present for the same trial when considering only messages received. This indicates that the source of error is not from incorrectly demodulating the signal, but rather from not detecting the presence of the signal. Since these messages are not detected, they each contribute significantly to the measured channel error. It seems that the message drops are related to the geometry of the room; some positions experience stronger echos than others.

We had multiple opportunities to test our communication method in a live game in a competition environment. Our first live game played with audio communication was at the RoboCup U.S. Open in Brunswick, Maine in May 2016. Table 5.5 shows the successful transmission rates between each pair of communicating robots for that

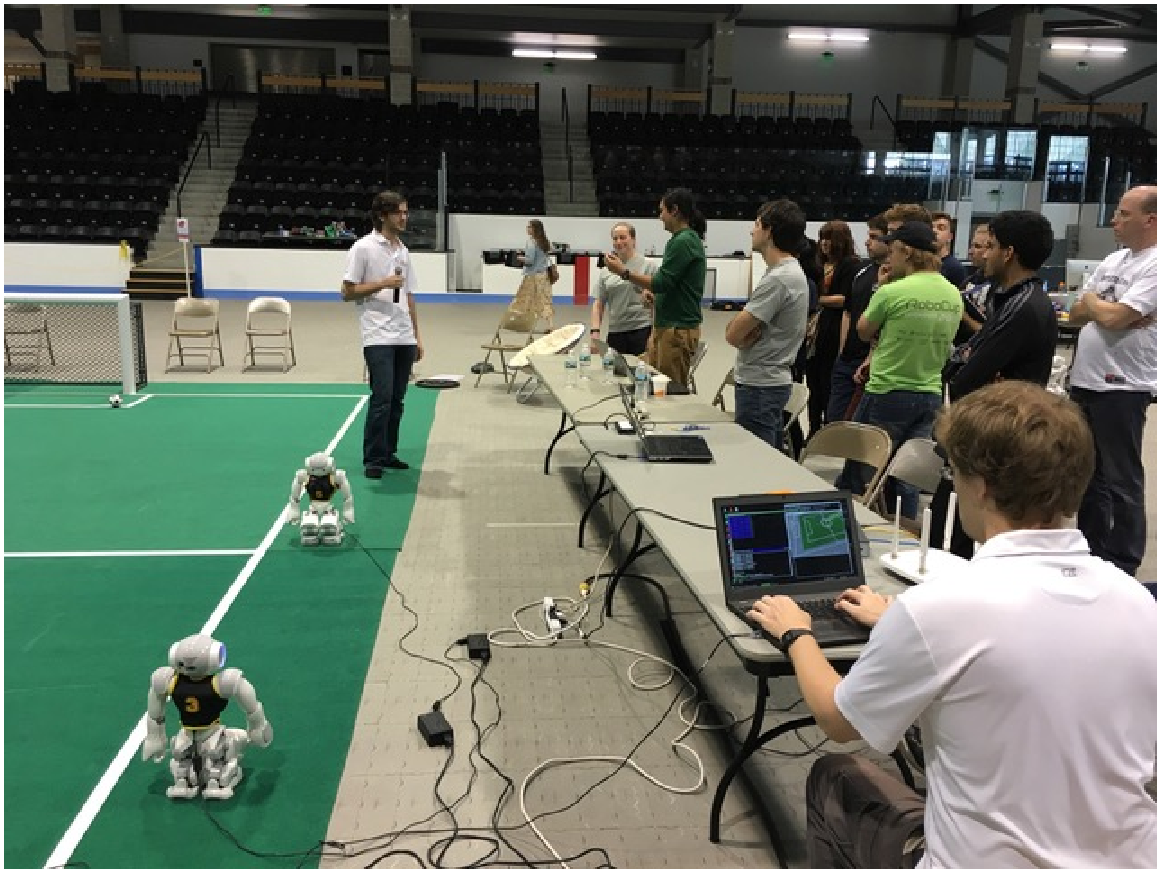


Figure 5.8: Kyle Poore talking about audio communication at the RoboCup U.S. Open in Brunswick, Maine

game. Figure 5.8 shows us talking about our method just prior to the game. This test was most comparable to the experiment conducted on walking robots while using packets, though there were significant additional challenges posed by the acoustic conditions present at the competition. The test was held in a (mostly) quiet and empty hockey arena with very high ceilings and concrete walls. This posed unique challenges for the communication system that had not been encountered during testing. For example, the hard surfaces cause sounds to reflect well, causing significant echo and reverberation. The large distance between walls in the arena cause reflections to be delayed, meaning that one symbol might continue echoing for several seconds,

	R_1	R_2	R_3	R_4
T_1	-	0.216	0.225	0.153
T_2	0.145	-	0.382	0.145
T_3	0.056	0.157	-	0.037
T_4	0.314	0.362	0.190	-

Table 5.5: Transmission Rates from a Live Game at the U.S. Open in Brunswick, Maine in 2016

affecting the measurements of future symbols.

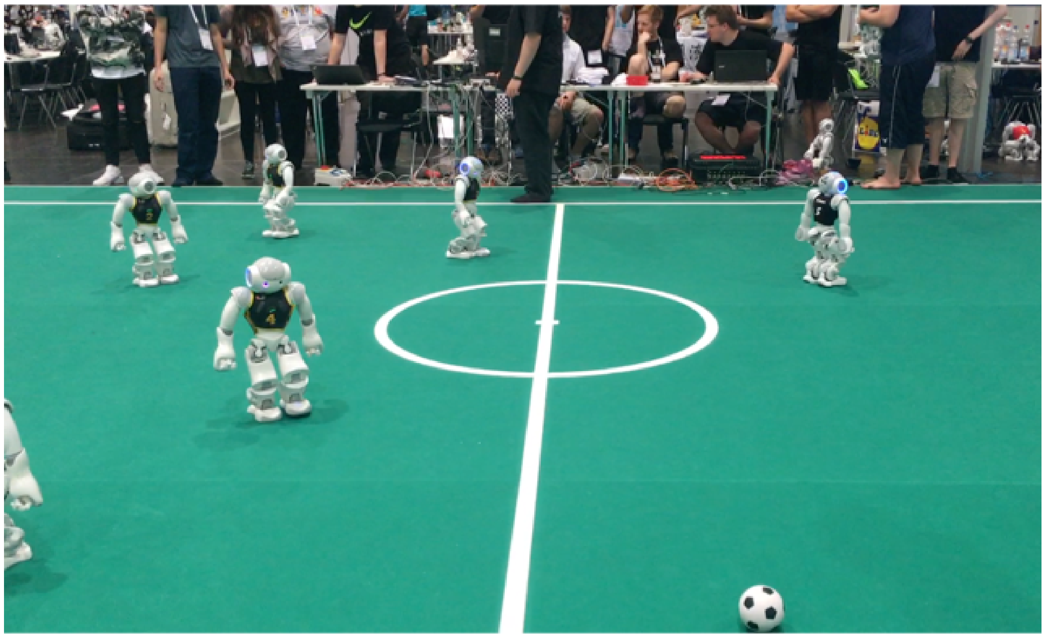


Figure 5.9: Testing audio communication in a live game at RoboCup in Leipzig, Germany

Our next opportunity to make a live test was in July at RoboCup 2016 in Leipzig, Germany, whose results are shown in Table 5.6. Figure 5.9 shows the game in progress; for a video of the game, please visit the following link: video. This test took place in a large conference center, again with hard, distant walls, though the acoustic environment behaved very differently. The room also held numerous competitions for the other robotic soccer leagues, and many spectators and participants. The ambient sounds were particularly different from the tests made in Brunswick in that there was a significant amount of noise due to the activities in the conference center.

	R_1	R_2	R_3
T_1	-	0.425	0.297
T_2	0.145	-	0.473
T_3	0.370	0.352	-

Table 5.6: Transmission Rates from a Live Game at the 2016 RoboCup Competition in Leipzig, Germany

Additionally the large number of people in the room absorbed many frequencies, diminishing the effect of reverberation.

These games showed that our method can indeed handle some of the challenges that a noisy competition environment poses for audio based messaging. For both of the live tests, we collected the transmitted and received messages from all five playing robots, and calculated the transmission accuracy for each pair by comparing the number of correctly received messages to the number of messages which were sent by each robot. The data we collected shows that several of the messages were correctly transmitted, though most others were dropped. These message drops were most likely due to the internal noise from the robot's motors, and since the robots are scattered throughout the field, the success rate between some pairs of robots are much higher than for other, more distant pairs. The performance of the communication channel performed noticeably better in the noisy conference center than it did in the quiet hockey arena. This can be explained by the excessive reverberation in the empty hockey arena, as opposed to the full conference center's dampening effects. Despite the conference center being much noisier than the quiet arena, the communication system performed better in the noisier environment. This seems to indicate that the reverberation of symbols has a greater impact on channel quality than does even high levels of ambient noise. This explanation makes sense, because the Goertzel filters have a very narrow receptive range around the desired frequency, thus most ambient noise is easily rejected. However, when the same frequencies emitted for communication echo back to the robots, delayed due to the further distance traveled, they frequently interfere with subsequent symbols, since they are made up of the same set of component frequencies.

Chapter 6

Summary and Future Work

6.1 Summary

We set out to create a backup communication system for the RoboCanes agent so that in the event of a Wi-Fi failure the agent is still able to share its beliefs of the state of the game with its team members. After reviewing the NAO robot's capabilities and hardware, we observed that transmitting and receiving audio signals via the speakers and microphones was the most promising method available to use for achieving this goal.

We consulted literature to develop a method of open-air audio signal transmission. We concluded that Phase Shift Keying (PSK) would be too sensitive to Doppler shifts and too difficult to recover with the available hardware, and that Frequency Shift Keying (FSK) would be too susceptible to external noise. The Dual-Tone Multi-Frequency approach did not seem to exhibit either of these issues. We then implemented a standalone program as a proof of concept and tested the method and refined it before integrating it into the RoboCanes framework.

Our method was extensively tested on the robots in lab conditions, taking message type, distance, and robot activity into account, and measured the error rates for each scenario. We also tested this method in multiple live games to verify that the method would work in the competition environment, and measured the message success rate between each pair of communicating robots.

6.2 Discussion

While the DTMF communication method seems to work relatively well between short range, quiet robots, the performance deteriorates drastically as the distance between the robots is increased. That said, we should expect that the error rates would be proportional to the inverse-square of the distance between the robots as the attenuation due to the lost intensity reduces the signal to noise ratio, but this is not observed in our experiments. Our best explanation for this is room reverberation. The experiments were performed in a room with hard, parallel walls, resulting in a significant echo. As these reverberations propagate and reflect about the room, the waves undergo constructive and destructive interference, contributing to the non-uniformity of the error curve in Fig 5.3.

The factor which seemed to adversely affect the communication performance the most was the robots' walking. The actuation of motors within the body of the robot contribute greatly to the noise level internal to the robot, most of which is inaudible to an observer. Other physical sources of noise on the robot stem from the creaking of the plastic covers as they deform as a result of movement and stress. These sources of noise are far from random, much less uniformly random. It is certainly possible that these internal noises interfere with one or more of the chosen DTMF frequencies. For this reason, we suggest that future methods be developed to automatically select frequencies which are not subject to as much interference from internal noise.

In Fig. 5.3, the bit error rate is relatively high for all but the short range, silent transmission experiments, but this can be misleading. Particularly in the domain of robotic soccer, message accuracy takes priority over reliable transmission. In fact it is for this reason that we do not bother to retransmit missed messages; it is simply better to wait for the next message from that robot. Similarly, we wanted to keep messages short to minimize the time between transmissions of a particular robot. This is why we did not use error correction codes; they most certainly would have improved message accuracy, but at too much a cost in the time taken to send the longer messages. Most of the error bits are bits that have been dropped due to either failure to recognize a message, or due to packet rejection because the checksum failed. As shown in Fig. 5.7, all of the received messages contained no bit corruption errors. We consider a message corrupted if as little as one bit of that message has been flipped. Such messages are completely rejected, since they are unusable by the agent. The effect is that error rates tend to be rather high. If we had counted only the erroneous bits, the error rate would certainly be significantly lower, but would not capture the effective performance of the channel.

Some of our previous work has shown excellent recognition of whistle signals, which used a logistic regression classifier on the result of an FFT (Poore et al. (2014)), however, this method is not suitable for this application because it is too computationally expensive. For this system to work properly, the receiver must be able to demodulate the signal just as fast as the transmitter can generate it. Furthermore, the mentioned approach is less appropriate for detecting/recognizing DTMF tones because it takes advantage of the presence of the many harmonic components of whistle-like signals, whereas for DTMF we specifically choose frequencies to mitigate harmonics for the reduction of interference between frequencies. The whistle detector also benefits from requiring low temporal resolution, allowing for longer analysis windows, while DTMF must reach a balance between temporal and frequency resolution.

One improvement that seems promising is to employ a frequency hopping technique that was developed to prevent torpedo control signals from being jammed during WWII by George and Kiesler (1942). The main idea behind this technique is that if an enemy detected the control signals for the torpedo and attempted to jam that set of frequencies, the control signals would already have switched to using a new set of frequencies, rendering the jamming attempt unsuccessful. A similar idea can be used for our purposes, since the effects of reverberation essentially constitute a self-jamming signal; usually the first symbol sent does not encounter much interference, but the reflections of that symbol negatively impacts the next few symbols until the wave has had enough time to decay sufficiently. We could use the frequency hopping technique to prevent the self-interference of our signal. It would require that we choose a larger set of frequencies; specifically, it should be an integer multiple of the number of distinct tones needed for DTMF. Our same frequency selection method could be used, choosing $8k$ primes instead of only 8, where k is the number of different frequency sets to be used. Additionally, k should be chosen so that the period of frequency hopping outlasts the period of decay for a single symbol. After sending our first symbol, we would continue to cycle through each different set of frequencies with each subsequent symbol. This should ensure that each symbol has had enough time to decay, so that when the same set of frequencies are used again (several symbols later), the reverberation from the first symbol has subsided.

Perhaps it is possible to combine the efficiency of only using a small number of frequencies with a machine learning technique – perhaps Reinforcement Learning (RL) or Deep Learning (DL) – which could possibly be trained to reduce the error rate. van den Oord et al. (2016) show very promising results with WaveNet in the application of DL to the generation of raw audio signals, and are able to reproduce sounds such as human voices or classical music, though their method does not yet work for real-time applications such as ours. This new approach would have to learn its own protocol instead of being given an inflexible coding scheme. For example it might learn to make symbols easily distinguishable from common internal noise patterns, or it might be able to learn the difference between a direct signal and its echo. Such a technique might also be used to improve other aspects that are important to communication, such as its bit rate, or even the scheduling of messages.

Both of the message scheduling methods we used are ignorant to the state of the soccer game. That is, whereas humans tend to communicate through context based messages, the robots are restricted to updating a fixed set of beliefs and states. Ideally, communication would be focused between robots which are performing a coordinated task, because there would be less time spent waiting for irrelevant or non-local players to perform a broadcast of their beliefs. The less latency there is between cooperating robots, the more effectively they may be able execute strategies. If the message transmission accuracy is improved enough, then it may become possible to design more adaptive message scheduling methods to allow these kinds of optimizations.

6.3 Conclusion

We have presented an approach for audio based broadcast communication between robots, using different states of activity and different message styles over multiple distances. The approach is based on fixed length DTMF messages. The results show that for short ranges, and robots with low activity levels, the method works well; however, with increased separation or activity levels, the message reliability rapidly deteriorates. The corruption of successful messages seems to stay relatively low, especially for the packet mode of operation, where the corruption rate is virtually zero, thanks to the use of a checksum. It is suspected that in the case of increased distance, the performance decline is more closely related to room reverberation; in a larger room, or outside, the method is expected to perform better.

It seems that automatic frequency calibration is a must, given the varying and unpredictable nature of ambient and internal frequencies. This could be achieved by the robot taking a sample of audio for several seconds both while motionless and while walking prior to being used in a new environment. The frequency data would then be collected from each robot to determine the least interfering frequencies, which themselves do not have any harmonic frequencies in common.

To increase the bandwidth, it might be possible to extend the number of high and low frequencies from four to perhaps eight each. This modified scheme could transmit $8^2 = 64$ different symbols, representing six bits of information each. Alternatively, one could double the number of frequency groups, yielding $4^4 = 256$ unique symbols; able to represent an entire byte of data. The problem with increasing the number of used frequencies is that it becomes difficult to prevent interference between them. The required frequency separation to achieve this would lead to the signal taking up much of the audible range of humans, which would be rather unsatisfying.

Beyond the limitations of the NAO robot's audio hardware, it is possible to extend the frequency ranges above 20kHz, enabling truly inaudible communication. This would be a highly desirable property whenever humans are present or working alongside robots, as the loud beeping can be irritating to people when listening for long periods. It is likely that microphones with better frequency response would allow us to use even higher frequencies closer to the 24kHz Nyquist limit admitted by the 48kHz sampling rate.

Other possibilities for future work might take inspiration from the ways that birds communicate. These kinds of communication have already passed the tests of millions of years of natural selection, and exhibit many properties that are desirable in the field of robotics. For instance, birds usually emit vocalizations that are easily recognizable or distinguishable from the noises in their environment. They learn to recognize these sound patterns despite them occurring at different frequencies, speeds, volumes, or distances. Birds also seem to avoid transmitting raw data through their songs, instead opting for event based communication, such as alerting the presence of a predator, or searching for a mate.

The challenges of communicating via open air sound waves are as plentiful as its potential rewards, and as robots and humans continue on a converging path into the future, the use of audio as a natural method of interaction will only prove more and more beneficial.

References

Aswath, S, Chinmaya Krishna Tilak, Abhay Sengar, and Ganesha Udupa (2013), “Design and development of mobile operated control system for humanoid robot.” *Advances in Computing*, 3, 50–56.

Athanasopoulos, Georgios, Henk Brouckxon, and Werner Verhelst (2012), “Sound source localization for real-world humanoid robots.” In *Proc. SIP*, volume 12, 131–136.

Carrara, Brent and Carlisle Adams (2014), “On acoustic covert channels between air-gapped systems.” In *International Symposium on Foundations and Practice of Security*, 3–16, Springer.

Dodd, Annabel Z (2002), *The essential guide to telecommunications*. Prentice Hall Professional.

George, Antheil and Markey Hedy Kiesler (1942), “Secret communication system.” US Patent 2,292,387.

Goertzel, Gerald (1958), “An algorithm for the evaluation of finite trigonometric series.” *The American Mathematical Monthly*, 65, 34–35, URL `http://www.jstor.org/stable/2310304`. Last visited on 6/6/2017.

Guri, Mordechai, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici (2016), “Diskfiltration: Data exfiltration from speakerless air-gapped computers via covert hard drive noise.” *arXiv preprint arXiv:1608.03431*.

Harris, Fredric J (1978), “On the use of windows for harmonic analysis with the discrete fourier transform.” *Proceedings of the IEEE*, 66, 51–83.

Hu, Jwu-Sheng, Chen-Yu Chan, Cheng-Kang Wang, Ming-Tang Lee, and Ching-Yi Kuo (2011), “Simultaneous localization of a mobile robot and multiple sound sources using a microphone array.” *Advanced Robotics*, 25, 135–152.

Kirovski, Darko and Henrique Malvar (2001), “Robust covert communication over a public audio channel using spread spectrum.” In *International Workshop on Information Hiding*, 354–368, Springer.

Kruijff-Korbayová, Ivana, Georgios Athanasopoulos, Aryel Beck, Piero Cosi, Heriberto Cuayáhuitl, Tomas Dekens, Valentin Enescu, Antoine Hiolle, Bernd Kiefer, Hichem Sahli, et al. (2011), "An event-based conversational system for the NAO robot." In *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, 125–132, Springer.

Mullins, Jonathan, Bernd Meyer, and Aiguo Patrick Hu (2012), "Collective robot navigation using diffusion limited aggregation." In *International Conference on Parallel Problem Solving from Nature*, 266–276, Springer.

Nakadai, Kazuhiro, Toru Takahashi, Hiroshi G Okuno, Hirofumi Nakajima, Yuji Hasegawa, and Hiroshi Tsujino (2010), "Design and implementation of robot audition system'hark'—open source software for listening to three simultaneous speakers." *Advanced Robotics*, 24, 739–761.

Nakamura, Keisuke, Kazuhiro Nakadai, and Gökhan Ince (2012), "Real-time super-resolution sound source localization for robots." In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 694–699, IEEE.

Navarro, Gonzalo (2001), "A guided tour to approximate string matching." *ACM computing surveys (CSUR)*, 33, 31–88.

Nguyen, Tho and Linda G Bushnell (2004), "Feasibility study of dtmf communications for robots." *Dept of EE, University of Washington Seattle WA*, 98195–2500.

Nyquist, Harry (1928), "Certain topics in telegraph transmission theory." *Transactions of the American Institute of Electrical Engineers*, 47, 617–644.

Okuno, Hiroshi G, Kazuhiro Nakadai, and Hyun-Don Kim (2011), "Robot audition: Missing feature theory approach and active audition." In *Robotics Research*, 227–244, Springer.

Panfir, Alina Ninett, Răzvan Gabriel Boboc, and Gheorghe Leonte Mogan (2013), "Nao robots collaboration for object manipulation." In *Applied Mechanics and Materials*, volume 332, 218–223, Trans Tech Publ.

Poore, Kyle, Saminda Abeyruwan, Andreas Seekircher, and Ubbo Visser (2014), "Single-and multi-channel whistle recognition with nao robots." In *Robot Soccer World Cup*, 245–257, Springer.

Sakagami, Yoshiaki, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura (2002), "The intelligent asimo: System overview and integration." In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, 2478–2483, IEEE.

Sauer, Gunter, Thomas Dickel, and Thomas Lotter (2014), "Acoustic wireless control–connecting smart phones to hearing instruments."

Saxena, Ashutosh and Andrew Y Ng (2009), "Learning sound location from a single microphone." In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 1737–1742, IEEE.

Smith, Julius O. (2010), *Physical Audio Signal Processing.* http://ccrma.stanford.edu/ jos/pasp/. Visited on 6/6/2017.

Srivastava, Anurag, Shubham Vijay, Atul Negi, Prasun Shrivastava, and Ashutosh Singh (2014), "Dtmf based intelligent farming robotic vehicle: An ease to farmers." In *Embedded Systems (ICES), 2014 International Conference on*, 206–210, IEEE.

Sun, Hao, Peng Yang, Zuojun Liu, Linan Zu, and Qinqi Xu (2011), "Microphone array based auditory localization for rescue robot." In *Control and Decision Conference (CCDC), 2011 Chinese*, 606–609, IEEE.

van de Pol, Takashi Iwai Frank, Jaroslav Kysela, and Abramo Bagnara (n.d.), "Alsa library api reference."

van den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016), "Wavenet: A generative model for raw audio." *CoRR abs/1609.03499.*

Wills, Jack, Wei Ye, and John Heidemann (2006), "Low-power acoustic modem for dense underwater sensor networks." In *Proceedings of the 1st ACM international workshop on Underwater networks*, 79–85, ACM.

Wrede, Sebastian, David Klotz, Samira Sheikhi, Dinesh Babu Jayagopi, Vasil Khalidov, Britta Wrede, Jean-Marc Odobez, Johannes Wienke, Laurent Son Nguyen, and Daniel Gatica-Perez (2013), "The vernissage corpus: a conversational human-robot-interaction dataset." In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, EPFL-CONF-192462.