2024-12-20

Development of a Real-Time Convolution System to Simulate Speech in Different Rooms for the Study of Vocal Strain

Bethany E. Wu
Brigham Young University

Development of a Real-Time Convolution System to Simulate Speech in Different Rooms for the Study of Vocal Strain

Bethany E. Wu

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Brian E. Anderson, Chair
Tracianne B. Neilsen
Brian D. Jeffs

Department of Physics and Astronomy

Brigham Young University

ABSTRACT

Development of a Real-Time Convolution System to Simulate Speech in Different Rooms for the Study of Vocal Strain

Bethany E. Wu
Department of Physics and Astronomy, BYU
Master of Science

Due to prevalent vocal health issues in teachers, the acoustics of K-12 classrooms has become a topic of study in acoustics. One way to understand the effects of a classroom's physical space on speech is with vocal effort studies. This thesis aims to enable these studies, without the need to move a talker from room to room, by creating auralizations through real-time convolution of speech with oral binaural room impulse responses (OBRIRs). These auralizations can be used to test talkers inside an anechoic chamber as they experience speaking in different acoustical environments. A system that can successfully execute convolution in real time requires finely-tuned parameters and an optimized algorithm. Efforts and lessons learned during the development of this system are shared. Finally, results from preliminary testing of talkers located inside classrooms are shared (the goal was to compare these results to those obtained using real-time convolution system (RTCS) simulations using these same OBRIRs); the data from these in-classroom tests provides lessons learned that can inform future vocal strain tests, those made in classrooms and with an RTCS, to ensure less variability and clearer trends in the results.

Keywords: real-time convolution, auralization, classroom acoustics, oral binaural room impulse response, STM32F746 discovery board, ping-pong buffer, block convolution, overlap-save method, non-uniform partitioned convolution, filter partitioning, vocal strain, Lombard speech, jitter, shimmer

ACKNOWLEDGMENTS

Funding for this research comes from the National Institutes of Health (NIH). The contract for this specific project has been setup by Dr. Eric Hunter at Iowa State University and Pam Hallam here at Brigham Young University (BYU), who I'd like to thank.

First and foremost, I'd like to thank my graduate advisor, Dr. Brian Anderson. Thank you for reaching out to me and entrusting me with this difficult project. And thank you for teaching me about the ups and downs of research, and most importantly, for teaching me how to continue working through them.

Thank you to my graduate committee and all other professors who willingly gave me advice that helped me keep moving forward. I'd especially like to thank Dr. Brian Mazzeo from BYU's Department of Electrical and Computer Engineering; Dr. Mazzeo's real-time convolution code was used and optimized for the purposes of this research. Other than providing code, Dr. Mazzeo also helped with optimizing the code to achieve low latency and helped diagnose issues I came across while working with the STM32F746 boards.

Lastly, I'd like to acknowledge my husband and all my fellow physics grads who made my time as a graduate student enjoyable. Thank you for being listening ears and tea-time buddies, and for encouraging me to de-stress through fun activities outside of school.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background Summary

1.1.1 Vocal Health of Teachers

The environments in which we live and work can negatively affect our vocal health and well-being. Because teachers are recognized as one of the largest groups of professional voice users, their vocal health is at risk daily. Elementary and high school teachers especially, are three times more likely than individuals working other occupations to develop issues related to vocal health [1]. Speaking for long periods of time in noisy environments leads to overexertion of the muscles around the larynx which can cause various occupational voice disorders [2,3]; these issues can in turn effect teachers' day-to-day lives and ultimately impose significant financial costs for them when seeking treatment. In addition, if these teachers are forced to continue teaching in their unaltered classrooms despite the adverse impacts on their vocal health, their poor voice quality can inhibit the children's learning and engagement [4]. For this reason, assessing typical current conditions and studying the potential causes of teachers' vocal strain is important.

However, not all teachers are susceptible to the same risks for developing vocal health issues, and not all teachers experience the same issues; this fact has added to the difficulty in studying teacher vocal health thus far. Previous studies have suggested that many factors–such as the general health of the teacher [5], teaching responsibilities [6], recovery periods [7], physiology [8], and the general environment of the classrooms [9]–can all play a role in causing vocal health issues in teachers. In an effort to isolate potential causes involved, this research specifically aims to investigate the relationship between teacher vocal strain and the acoustical properties of the classrooms in which they teach in.

1.1.2 Classroom Acoustics

Various studies researching the correlation between teacher vocal health and poor acoustics in classrooms have relied on self-reports from the teachers' themselves [10]. These self-reports are prone to be influenced by biases related to general satisfaction of the teachers' classrooms/workplaces. The present, overall research effort (including other research teams at BYU and at the Univ. of Iowa) aims to contribute objective measurements of acoustical conditions in the classrooms. With information from both self-reported and objective studies, the acoustical conditions of classrooms and some indications of how these conditions correlate to teacher vocal health in classrooms can be determined.

The acoustics of a classroom can be quantified by metrics such as the A-weighted background noise level, reverberation time (RT60), speech transmission index (STI), the percentage of articulation loss of consonants (%ALCONS), clarity (C50), strength (G), support (ST), room gain (Grg), and voice support (STv). Because the American National Standards Institute (ANSI) for Classroom Acoustics only prescribes standards for A-weighted background noise levels and RT60, this research focuses on these two acoustical properties and their effect on teachers. Additionally, most of the other metrics listed aim to quantify the intelligibility of the speech for the listeners, which is not the focus of the current study.

The A-weighted background noise level is a measure of how noise levels are perceived in the room. The A-weighting is applied to correspond to how humans perceive the frequency dependence of that noise (nominally assuming sound levels that exist on the 40 phon curve) and denoted with the unit dBA. Common sources of background noise heard inside classrooms include HVAC systems and electrical equipment (projectors, computers, etc.), as well as external sources such as noise generated in adjacent classrooms, corridor noise, playground activity, traffic, and planes. The ANSI limit for background noise inside a classroom is 35 dBA maximum, when levels are averaged over one hour.

RT60 quantifies the slope of the decay of the sound level over time and is defined as the time it takes for sound level to degrade by 60 dB in an environment. According to the ANSI S12.60-2002 standard, the RT60 in an unoccupied, furnished classroom with a volume under 10,000 ft^3 must be <0.6 s; for classrooms with volume between 10,000 and 20,000 ft^3, the maximum RT60 requirement is relaxed slightly to <0.7 s.

1.1.3 Auralization

Auralization is the process of rendering the binaural listening experience at a given position from a sound source in a specific modeled space; the auralization allows a listener to virtually hear what it would sound like in that physical space, while that listener is actually located in a different space. Auralization not only recreates the sensation of a sound source within a space, but also the aural impression of the acoustic characteristics of a space (indoors or outdoors) [11]. When created accurately, auralizations can be used to virtually bring a person to another environment through just their hearing. This approach could be helpful for a variety of uses including noise quality assessments, virtual reality systems, training of architects and audio professionals, and studies in psychoacoustics.

Using auralizations, teacher subjects can be virtually placed in the acoustic environments of different classrooms, while they are physically located in an anechoic chamber. Tests can be conducted and changes in their vocal effort can be easily tracked, without any consistency issues that may arise while moving from one room to another. Different acoustic environments can be simulated through auralizations based on measured oral-binaural room impulse responses (OBRIRs). These room impulse responses are measured by emitting sound from a mouth (oral) of a manikin and recording the response using microphones located in both ears (binaural) of the same manikin. For this project, OBRIRs of measured classrooms in the Eyring Science Center at BYU are used to create the auralizations.

1.1.4 Vocal Strain

Vocal effort is defined as the individual's physical and/or mental exertion involved in their vocal production. Vocal effort is quantified by parameters such as fundamental frequency (F0) (associated with the pitch of one's voice), fluctuations of F0 (quantified by its standard deviation over time), first formant frequency (F1) (formant frequencies are the resonances of the vocal tract), and voice sound pressure levels (SPLs) [12]. Three environmental components have been shown to affect the vocal effort of a talker: distance to the listener, background noise, and duration of vocal use [13]. From the point of view of an individual reporting high vocal effort, some common symptoms experienced include pain or discomfort while speaking or a tight feeling in the throat [14].

In relation to vocal effort, vocal strain is defined as the physical discomfort experienced by individuals who report high vocal effort. Multiple metrics are associated with the perception of vocal strain; some main metrics include vocal intensity or loudness, jitter and shimmer levels [15], F0, cepstral peak prominence (CPP), and spectral slope [16]. In noisy environments, adaptations in a talker's speech production are known as the Lombard effect (leading to Lombard speech); this effect elicits certain changes in vocal strain parameters, such as voice SPLs, F0, and spectral slope, and lead to what studies call Lombard speech [17, 18].

The vocal strain parameters mentioned above can be calculated from recordings of talkers under acoustic conditions that lead to increased vocal effort. The vocal strain analysis for this project is done by speech scientists working with Dr. Eric Hunter at The University of Iowa, who are partners on the overall research effort that is funding the work described in this thesis. Using the same Statistical Package for the Social Sciences (SPSS) software used in past studies [19, 20], Hunter and his group can easily process speech recordings using analysis programming code to obtain various parameters quantifying vocal strain.

1.2 Previous Work

Understanding all the measures taken to make an RTCS and creating one with minimal latency is one of the main goals of this research. Although RTCS's exist from previous research studies and in proprietary commercial implementations, the literature lacks information on how computation of the convolution was achieved in real time. The previous studies mentioned below provided background information relevant to this current research.

1.2.1 Work of Yadav *et al.*

In 2012, Yadav *et al.* [21] from the University of Sydney published a paper on the development of a low-latency RTCS. The goal of their research was to create a system that could be used to study the effects of room acoustics on the sound of one's own voice. Using a visual programming language called Max/Max Signal Processing (MSP), real-time digital audio signals were manipulated in their RTCS without dedicated digital signal processing (DSP) hardware. To perform convolution in real time, a commercially available virtual studio technology (VST) plugin called SIR2 was used. Head-tracking was also implemented into the system in Max/MSP; this required the measurement of OBRIRs using a head and torso simulator (HATS) at 2° yaw intervals from -40° to 40° [22]. For the analog-to-digital and digital-to-analog (AD/DA) conversion, an 8-channel hi-end AD/DA converter from RME called the RME ADI-8 Quadspeed was used. A typical user of their RTCS would be seated on a wooden chair in an anechoic environment; they would wear a 4066 omnidirectional headset microphone by DPA Microphones to collect input speech and K1000 off-ear headphones by AKG Acoustics that output the system's response. After their RTCS was developed, Yadav *et al.* used it for variety of studies related to the acoustics of open-plan office environments [23, 24] and auditorium stages [25, 26], and the auditory room size perceived when using an RTCS [27]. These studies addressed issues related to the effect of multiple talkers in a room, and how a person's auditory experience has a great effect on how small or large they perceive a room to be. However, specific work had yet to be done on using the RTCS for testing how talkers change their voice in different acoustical environments. This became the main focus of another research project by Jenny Whiting mentioned below.

A key feature to the success of making an RTCS operate in real time is low latency of the system. The total system latency (TSL) is defined as the time delay between the onset and the response of an event. In the case of an RTCS, the TSL is described as the round-trip time taken from the instant a person's voice is recorded at the headset microphone to the time that the convolved speech is emitted from the headphones. The TSL, in Yadav *et al.*'s RTCS, was measured to be 7 ms and 11 ms with a buffer size of 128 and 256 samples respectively when sampled at 48 kHz. To make up for this, the OBRIRs used were truncated by the latency time (7 or 11 ms depending on the buffer size), so the latency was essentially zero at the output. The truncation also effectively removed the direct sound from the OBRIRs. Fortunately, the off-ear headphones allow the direct sound to be heard naturally rather than through the system. Through this process, the RTCS created by Yadav *et al.*

achieved the low latency required to accurately simulate nearby environments. In addition, they also found that their system could perform convolution with long room impulse responses (RIRs) and still hold the same low latency (although the length in time of a 'long RIR' is never quantified within their paper). This accomplishment makes the RTCS by Yadav et al. an ideal system for use in various vocal studies. However, due to the lack of details on the actual development of the RTCS algorithm, it is difficult to reproduce their results (especially when using longer RIRs). One goal of this current research is to bring to light tips and assumptions that can be made to achieve a low-latency RTCS.

1.2.2 Work of Whiting and of Rollins

In 2018, BYU Master's student Jenny Whiting [28], under the supervision of Timothy Leishman, had a goal to replicate and improve upon the RTCS created by Yadav *et al.* [21], and then use it for a study on vocal effort. The hardware used for Whiting's RTCS also included the RME ADI-8 Quadspeed AD/DA converter (similar to Yadav *et al.*), as well as an HDSPe AES PCI card combined with a personal computer and the Windows operating system. The convolution in Whiting's system was performed using the same SIR2 VST plugin used in the RTCS by Yadav *et al.*

The OBRIRs used by her system were measured using the KEMAR manikin. These OBRIRs had to be equalized to account for the frequency dependence introduced by KEMAR's mouth loudspeaker, which did not have a flat response over the range of interest (80 Hz-10kHz). Similar to the methods used by Yadav *et al.*, the OBRIRs were also truncated by 6 ms to account for the 6 ms RTCS latency she measured. Finally, Whiting improved upon their system by applying another equalization filter to the OBRIRs to compensate for any "undesirable effects caused by components of the RTCS".

After conducting objective and subjective tests on the RTCS to verify that it can realistically produce binaural auralizations in real-time, Whiting used the system to perform vocal effort tests. These tests were done by assigning talkers with three speech tasks in nine different virtual environments simulated by the RTCS. The OBRIRs used to simulate these virtual environments were measured in the reverberation chamber, a large classroom, and the de Jong Concert Hall all at BYU. By adding varying amounts of anechoic wedges to the reverberation chamber, the absorption was changed to represent seven different acoustic conditions.

The design and implementation of Whiting's vocal effort study was strongly influenced by the work of another previous BYU student, Michael Rollins, also under the supervision of Leishman [29]. For Rollins' undergraduate capstone project, he conducted vocal effort tests inside different physical spaces; participants were recorded as they were prompted by an interviewer (in the same physical space as the participant) through several speech tasks. The recordings were then used to calculate key vocal parameters such as F0, mean loudness of voiced speech (dB), CPP, and spectral slope to quantify vocal effort.

Whiting followed the same procedure as Rollins, but with participants using her developed RTCS inside the anechoic chamber; she simulated the same physical environments Rollins used, presented the same vocal tasks to participants (without having an interviewer in the chamber), and calculated the same vocal parameters. Comparing her results with that of Rollins', she found that overall, there was good agreement in vocal effort trends between the two studies; however, some vocal parameters such as mean pitch strength, shimmer, and AVQI showed large disagreements. The results from Whiting's research led to doubts about the accuracy of the RTCS, because it used a commercial plugin to perform the real-time convolution. Upon completion of Whiting's research, it was suggested by her thesis supervisor Leishman that ideally, future implementations of an RTCS could perform the real-time convolution with known functions and assumptions to decrease the existence of any errors in the system's computation.

1.2.3 Work of Robertson

During her undergraduate degree at BYU, under the supervision of Dr. Brian Anderson, Megan Robertson conducted research with an overall goal to quantify the acoustical parameters of various classrooms in the Utah County, Utah area. She surveyed the acoustic environments found in typical K-12 classrooms. By visiting three elementary schools and three high schools in Utah County, she obtained RIR and OBRIR measurements from a total of 26 classrooms: 17 small (<283 m^3), 4 medium (283-566 m^3), and 5 specialized classrooms (for choir, orchestra, dance, and woodshops classes). The RIR measurements were used to calculate speech parameters such as C50, STI, %ALCONS, decay time (DT), RT60, and background noise; these calculated parameters were then compared to the classroom acoustics standard metrics (ANSI S12.60-2002). Robertson found that from the classrooms sampled, C50, STI, and %ALCONS met the recommended standards for classroom acoustics. However, the majority of the classrooms had background noise levels that were above the ANSI standard, and only 16 out of the 26 classrooms met the RT60 ANSI standard. Robertson's research motivates the need to study the specific effects of RT60 and background noise on teacher vocal strain, since they are a current problem in most K-12 classrooms. Measurements have been made in an additional 13 classrooms in Utah by students who continued Megan's work after she left but these results have not yet been tabulated.

Robertson also measured RIRs and OBRIRs inside 10 different rooms in the Eyring Science Center (ESC) at BYU. The ten rooms included lecture-style classrooms of varying sizes, laboratory-style classrooms, and a reverberation chamber. She did not do further analysis of these rooms, nor did she report on them.

1.3 Thesis Outline

The main objective of this research is to develop an RTCS based on known techniques and without using shortcuts or approximations, and then use it to study the correlation between classroom acoustics and vocal strain in teachers. The low-latency RTCS created and explained in this thesis is unique, because it does not utilize the same commercial VST plugins used by previously created systems. By implementing a real-time convolution algorithm from scratch, accuracy in the convolution results is possible and, most importantly, all the assumptions and shortcuts used in the development of the algorithm may be clearly stated. After modifying the system parameters to obtain acceptable latency, the RTCS was intended to be used for a vocal strain study.

The goal of this work is to compare vocal strain measurements of talkers in actual classrooms with varying noise levels and varying reverberation conditions to vocal strain measurements of talkers using the RTCS system developed here. To find the correlation between classroom acoustics and vocal strain specifically, subjects must be tested as they experience various classroom conditions. However, physically bringing a subject to multiple locations to achieve a diverse survey of classrooms is troublesome and can introduce unwanted variables into testing. For this reason, the RTCS would be used to virtually bring people to multiple classrooms, as they physically stayed in an anechoic chamber. Using an RTCS would simplify the testing process, allowing subjects to be in a neutral setting (inside the anechoic chamber) in between the switching of virtual sound spaces. The following chapters in this thesis explain the whole process of developing a low-latency RTCS, and a preliminary vocal strain study that was designed and conducted inside classrooms.

Chapter 2 is deep dive into the developed system. It starts by describing and illustrating the methodology on how convolution is done in real time; the algorithm used was specifically created to reduce computational time and latency. Then the actual implementation of the RTCS is discussed. After much trial and error, a software platform and specific hardware were ultimately chosen to prioritize real-time accuracy. Lastly in this chapter, a discussion on latency and measures taken to reduce it is given. This subsection includes an analysis of parameters such as sampling frequency, input block size, and OBRIR duration, and how they affect system latency. The final RTCS is described in detailed including the software algorithms and multiple hardware components used to result in a zero-latency system. Findings of unknown artifacts in the RTCS output are also explained, and recommendations for future steps towards improving the RTCS are shared.

Chapter 3 consists of results from vocal strain tests done in physical classroom environments. These tests highlight the impact of classroom acoustics on teacher vocal health. Specifically, the vocal strain tests focused on the impact of background noise and reverberation time within a classroom environment. Multiple environments were tested by producing artificial noise inside a classroom and reducing reverberation inside a classroom by adding absorptive wedges. The test results are analyzed by focusing on four main vocal strain parameters that are good indicators for vocal strain: loudness (in dB), fundamental frequency, jitter, and shimmer. The results from this vocal strain study emphasize the importance of creating acoustically ideal classroom environments to reduce vocal strain of the people talking inside of them.

Lastly, Chapter 4 is a summary of the work done for this Master's thesis. It explains the limitations and setbacks that were experienced throughout the project and discusses suggestions for future work relating to this project.

Chapter 2

Development of the RTCS

A portion of the work mentioned in this chapter was presented on May 15, 2024 at the 186th ASA Meeting in Ottawa, Canada [30].

2.1 Introduction

Teachers are recognized as one of the largest groups of professional voice users. Because of this, their vocal health is at risk daily, which can cause lasting effects and impose significant financial costs for them. Elementary and high school teachers especially are three times more likely than individuals working other occupations to develop issues related to vocal health [1]. A study relating classroom acoustics to teacher vocal health found that changes in voice symptoms correlated positively with the teachers' average noise exposure during the workday [9]. In the study, noise exposure was quantified by measuring background noise levels and the reverberation time (RT60) of the classrooms that the teachers were tested in.

Previous studies with this same goal conducted tests by placing teachers inside different classrooms and repeating test procedures to obtain results. However, physically bringing teachers to multiple locations to achieve a diverse survey of classrooms is troublesome and can introduce unwanted variables into testing. For this reason, a real-time convolution system (RTCS) is being created to virtually bring teachers to multiple classrooms, as they physically stay inside an anechoic chamber. This RTCS would produce binaural auralizations of any classroom environment, which would be created through convolution of real-time speech with prerecorded oral binaural room impulse responses (OBRIRs).

To both ensure the accuracy of the system's output and have the ability to state all the assumptions and shortcuts used, the RTCS was coded from scratch. In order to create realistic auralizations with an RTCS, the latency of the real-time convolution computation must be as low as possible and at a shorter time than the earliest sound reflections might arrive, which was estimated to be about 6 ms. This chapter highlights the various efforts made to minimize latency by optimizing both the software and hardware used for the RTCS. The real-time convolution algorithm was optimized to further decrease total system latency (TSL); this process included finding optimal system parameters to ensure both an accurate system output and low latency. Overall, the final implementation and parameters chosen for this RTCS resulted in ideal latency for the purposes of this research. Problems with noise in the RTCS still exist but this chapter describes progress being made.

2.2 Convolution in Real Time

An auralization is created by computing the convolution of a room's finite impulse response (FIR) with a source signal (such as a speech recording having L samples). Convolution, which commonly implies using signals in the time domain, computes a summation of time shifted impulse responses. Specifically, the convolution of two signals is formed by taking each point in the source signal and multiplying it by each point in the FIR signal and preserving the relative time delays of the FIR. Each of these L multiplication results, which are just scaled versions of the FIR, are then time staggered so that they start at the same time as the individual source signal sample. The resulting time staggered and scaled FIR results are then summed together to form the overall convolution output.

Another important thing to note about convolution is the length of each signal involved. For an input *x[n]* with *L* samples and FIR *h[n]* with *P* samples, the output signal *y[n]* (from the linear convolution) will be of length *N*=*L*+*P*-1 samples long. Discrete time convolution is expressed mathematically in the time domain in Equation 2.1, where *x[n]* represents the input signal, *h[n]* is an FIR, and *N* is the total number of samples in the output; this equation also states the commutative property of convolution.

$$y[n] = x[n] * h[n] = \sum_{m=0}^{N-1} x[m]\, h[n-m] = h[n] * x[n] \tag{2.1}$$

Convolution can also be seen as a modification of the input *x[n]* by the FIR *h[n]*. For the purposes of this RTCS, *x[n]* is blocks of real-time speech from a talker, and *h[n]* is a FIR recorded by one ear microphone (left or right) of a classroom's OBRIR. The resulting signal, *y[n]*, is the auralization created, sounding as if the talker was speaking in the room where the OBRIR, *h[n]*, was measured.

For long input signals, the number of computations can become very large, which increases latency for the RTCS. To combat this, convolution can be done in the frequency domain. This changes the order of the complexity of convolution for an input signal of length N from $\mathscr{O}(N^2)$ in the time domain to $\mathscr{O}(N \log_2 N)$ in the frequency domain, which is only strictly true when N is a power of two [31]. Convolution in the frequency domain is computed by Equation 2.2,

$$y[n] = x[n] * h[n] = \mathscr{F}^{-1}\{\mathscr{F}\{x[n]\} \cdot \mathscr{F}\{h[n]\}\} = \mathscr{F}^{-1}\{X[k] \cdot H[k]\} \tag{2.2}$$

where $\mathscr{F}$ denotes the Discrete Fourier transform in this case; *X(ω)* is the source spectrum while *H(ω)* is the frequency response function. As mentioned before, for long input signals or a long FIR, convolution can become very computationally expensive. To further combat this, a few different methods were used to decrease the computational complexity (and therefore latency) of the RTCS, which are discuss in the sections below.

2.2.1 Block Convolution and the Overlap-Save Method

When looking at the case when the input signal *x(t)* is significantly long, convolution with an FIR can be performed in blocks through block convolution [32]. This process suggests dividing the input signal into smaller blocks $x_i[n]$ and then simultaneously performing the convolution of each block with the FIR *h[n]*. Then, by using a variety of methods, the convolution output for each block can be correctly combined to form the overall convolution result. The overlap-save method is one way to efficiently perform block convolution using less memory, direct processing, and avoiding redundant computations.

Using the overlap-save method starts by breaking the input signal into overlapping blocks; the amount of overlap depends on the length of the FIR chosen. Assume the input signal is initially divided into successive blocks of length *L* and the FIR is *P* samples long. The overlapping blocks of the input signal will be of length *L*+*P*-1, where each block overlaps the preceding block by *P*-1 samples. Next, the circular convolution result of each overlapping block is calculated using the Fast Fourier Transform (FFT) method.

As mentioned before, when performing the linear convolution of an input with *L* samples and FIR with *P* samples, the output will be *L*+*P*-1 samples long. Conversely, when performing the circular convolution of the same lengthed input signal and FIR, the length of the output will be the maximum value between the two lengths. Therefore, in the circular convolution of an overlapping block of length *L*+*P*-1 and FIR of length *P*, the output will be of length *L*+*P*-1. Due to the circular nature of the FFT-based convolution, the first *P*-1 samples are discarded leaving *L* samples from each convolution result. These *L* samples can then be concatenated directly, without overlap or addition, to form the correct convolution result of the whole input signal with the FIR. The overlap-save process is illustrated in Fig. 2.1.

Because the input signal used for convolution in this project is speech recorded in real time, it can be classified as infinitely long. Because of this, block convolution and the overlap-save method are used to compute the convolutions in the RTCS. To perform block convolution, the input signal is continuously recorded in blocks of chosen length. The input block size (in ms) is one of the RTCS parameters discussed in Section 2.4. To ensure that speech is recorded continuously (without gaps), a special structure called a ping-pong buffer is used.

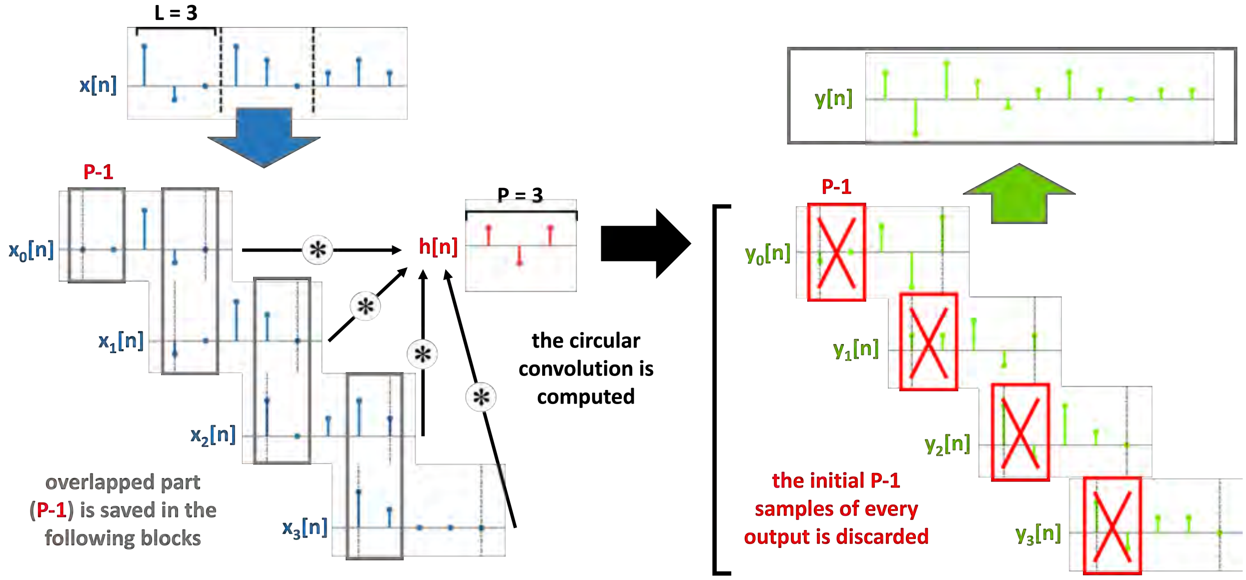
L = 3
x[n]
P-1
x0[n]
x1[n]
x2[n]
x3[n]
overlapped part (P-1) is saved in the following blocks
*
h[n]
P = 3
the circular convolution is computed
y[n]
P-1
y0[n]
y1[n]
y2[n]
y3[n]
the initial P-1 samples of every output is discarded

Figure 2.1 An illustration of using the overlap-save method for block convolution of an input signal *x[n]* with an FIR *h[n]* to obtain the overall result *y[n]*. (Note: The latency caused by the computational time of each convolution process is not shown in this figure.)

2.2.2 Ping-Pong Buffer

Ring buffers are a common data structure used in computer science and help in simplifying data management, optimizing latency, and reducing the number of possible operations per second [33]. They perform like a queue with a first in, first out (FIFO) process. Ping-pong buffers are ring buffers with just two memory locations and operate using pointers to the head and tail of the buffer. These pointers keep track of where the processor is so that data can be simultaneously read and written while the RTCS runs [34].

Using the ping-pong buffer, speech data can be recorded and written into one block while the previously recorded block of speech can be read from the other block and used for convolution. The length of the input blocks stored must be chosen carefully so that the program has enough time to do the convolution before data is overwritten. If, for example, the convolution of a certain data block size (in time) took more computational time than the time needed to record the block size itself, multiple blocks would be needed to prevent corruption of data. For this project, the input block size was minimized to both reduce latency and to avoid the need for a larger ring buffer.

2.3 Implementation

Although there are a variety of platforms (software and hardware) for implementing an RTCS, few can run the RTCS with minimal/zero latency. The sections below highlight the efforts made to minimize latency by researching and testing different software and hardware platforms in which to implement the RTCS. Through the guidance of multiple electrical engineers and other scientists, the decision was made for the implementation of the current RTCS.

2.3.1 Code Implementation

2.3.1.1 MATLAB

Since the 1990s, the software system MATLAB has been commonly used for digital signal processing (DSP) projects [35]. What makes MATLAB ideal for DSP is its interactive environment, extensive libraries, and high-speed calculations. In addition, since MathWorks (the company behind MATLAB) offers educational licenses, MATLAB is easily accessible to universities and their students. At BYU, MATLAB is the main software program used for computational work in acoustics, as well as in a few electrical engineering courses. Because of this, the first version of the RTCS was implemented in MATLAB.

To perform the convolution of two signals in MATLAB, the '*conv*' function can be used with the signals as the function's input arguments. This function uses the direct method of computing the convolution, in the time domain, and can be computationally expensive for large signals. For real-time processing, the '*filter*' function can be used to compute the convolution efficiently, in the frequency domain, using digital filtering methods. The most efficient MATLAB function for computing the convolution of two long signals is '*fftfilt*'; this function performs the frequency domain convolution with block processing to leverage the efficiency of the FFT.

The MATLAB version of the RTCS was implemented using a ping-pong buffer, and all three functions mentioned above were used and tested. The minimum latency achieved was around 270 ms; this latency was attributed to an inherent delay in MATLAB when recording and immediately playing audio. Some potential sources of the inherent delay include the audio buffering (overflows/underflows), audio driver latency, and the operating system and hardware used. To combat this inherent delay, the MATLAB RTCS was also converted to a standard executable file (EXE), a MATLAB executable file (MEX), and a MATLAB audio plugin; all of these other implementations were expected to run with lower latency than a MATLAB script but resulted in the same measured latency.

2.3.1.2 Field-Programmable Gate Array

To solve the problem of inherent delays, an RTCS implemented on a field-programmable gate array (FPGA) was considered. An FPGA is a type of configurable integrated circuit that can be reprogrammed after manufacturing to implement a large variety of systems. They consist of an array of programmable logic gates that can be configured to interconnect with other gates and perform various digital functions. These devices are known for their inherent parallel architecture, high clock speed, and, therefore, fast input/output (I/O) rates, which make them popular for implementing real-time systems [36–38]. To control the behavior of an FPGA, a user must implement a design in a hardware description language (HDL) like VHDL and Verilog.

In MATLAB and Simulink, an algorithm can be converted to HDL code and implemented in hardware by using MATLAB's HDL Coder package. As long as the algorithm is written with syntax and functions that are compatible with HDL code generation, its equivalent in an HDL can be created. Unfortunately, none of the convolution functions (i.e. '*filter*' or '*filtfilt*') used in the MATLAB version of the RTCS algorithm were compatible with the HDL code generator. Therefore, in order to implement an RTCS on an FPGA, a new algorithm would have to be written using functions compatible with the generator; these functions can be found in MATLAB's DSP Toolbox. We were advised that the learning curve for the HDL code and FPGA architecture was too steep for this RTCS application. However, FPGAs should be a viable option for future implementations of real-time systems like this.

2.3.1.3 STMicroelectronics Microcontroller

In between the efficiency and power of MATLAB and FPGAs are microcontrollers made by STMicroelectronics. The STM32F746 Discovery Board specifically is powered by the ARM Cortex-M7 core, which provides high processing power for a wide range of applications. The board includes a 4.3-inch capacitive touchscreen thin film transistor liquid-crystal display and extensive connectivity options to ease user interaction and communication. Using STM32 software development tools like STM32CubeIDE and Keil Studio, the board is most commonly programmed in C or C++. Due to the ideal efficiency and power of the STM32F746, as well as its user-friendly architecture, the current version of the RTCS is implemented on these boards. Figure 2.2 shows the front and back of an STM32F746 Discovery Board used in the RTCS.

In the senior-level, Introduction to DSP course taught by BYU's Electrical and Computer Engineering Department, these discovery boards are programmed by students to solve problems in many different contexts such as radar, sonar, and communications. Using preexisting code that employs a ping-pong buffer to perform functions in close-to real time, the RTCS was implemented on the STM32F746 Discovery Board; a portion this code is included in Appendix C. The system utilizes the board's line I/O ports to send a speech signal into the board for convolution and then send the auralization back out. To maximize the speed of the RTCS, a total of six boards (three for each channel and respective OBRIR) are used to perform the convolution of the left and right OBRIRs with input speech. The reasoning for using three boards per OBRIR convolution is explained in Section 2.4.2.

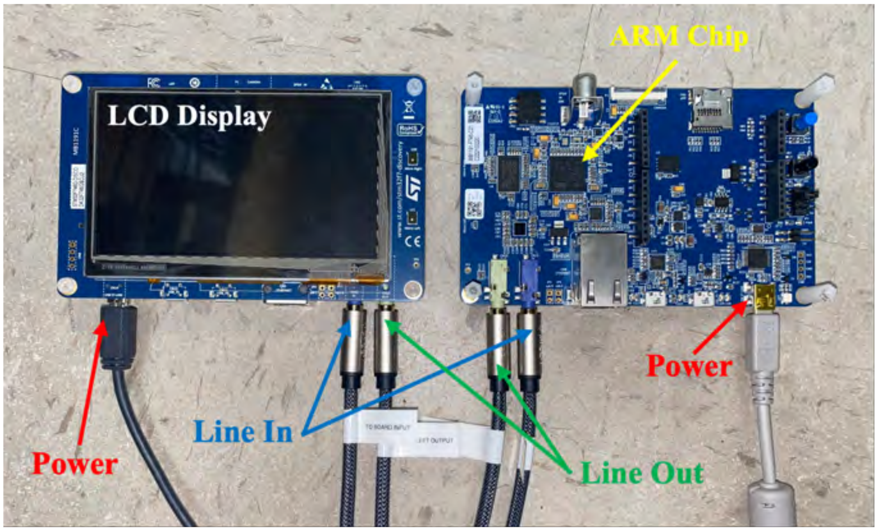

Figure 2.2 The front (left) and back (right) of an STM32F746 Discovery Board. The main components of the board are indicated with arrows.

2.3.2 System Hardware

Many pieces of hardware are used in conjunction with the RTCS implemented on the STM32F746 boards. A diagram of the whole system's process is shown in Fig. 2.3, and the real setup is pictured in Fig. 2.4. Speech from a talker is recorded using a DPA 4066 headset microphone, which is provided 48V phantom power and preamplification by a Focusrite Scarlett 18i20 (2nd Generation) before being sent into the STM32F746 boards. After the real-time convolution processing, the resulting output signals from the STM32F746 boards are correctly summed and amplified by a second Focusrite (acting as a mixer). By routing the output signals directly to the Focusrite's headphone outputs (in both instances), the resulting signals can be output with "ultra-low latency". From the second Focusrite's headphone outputs, the signals are played through off-ear AKG K1000 headphones; these headphones were chosen to allow for the uninhibited sound transmission of the direct sound from the talker's mouth to each ear. Figure 2.5 shows KEMAR, a G.R.A.S. head and torso simulator, wearing both the headset microphone and off-ear headphones used in the RTCS.

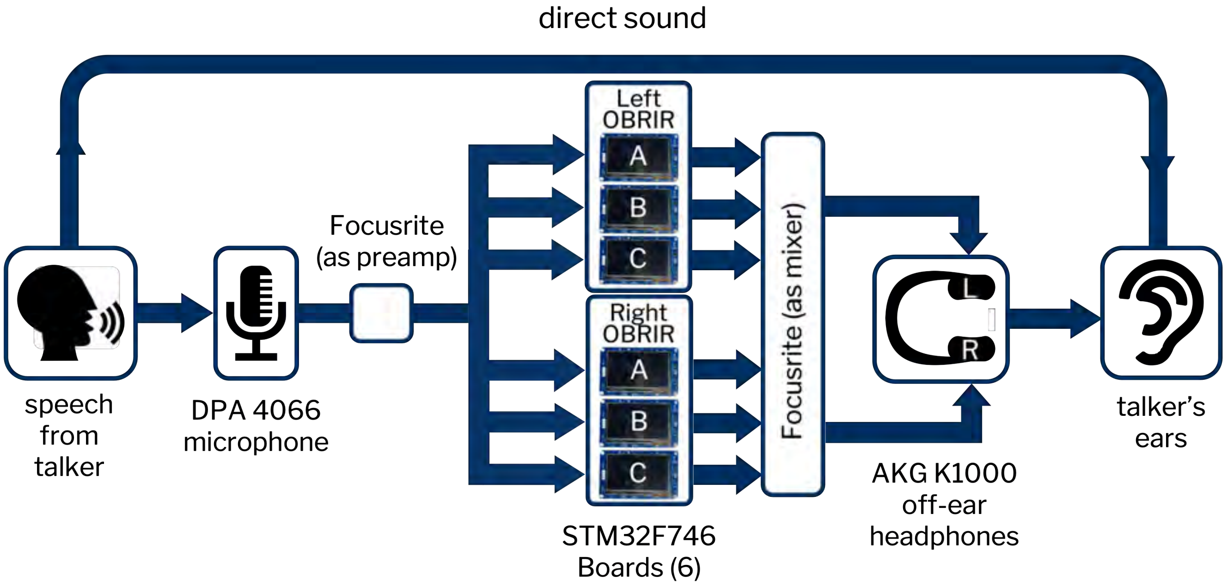

Figure 2.3 A diagram of the basic process and components of the whole RTCS. Different parts of the left and right OBRIRs are loaded onto each STM32F746 board.

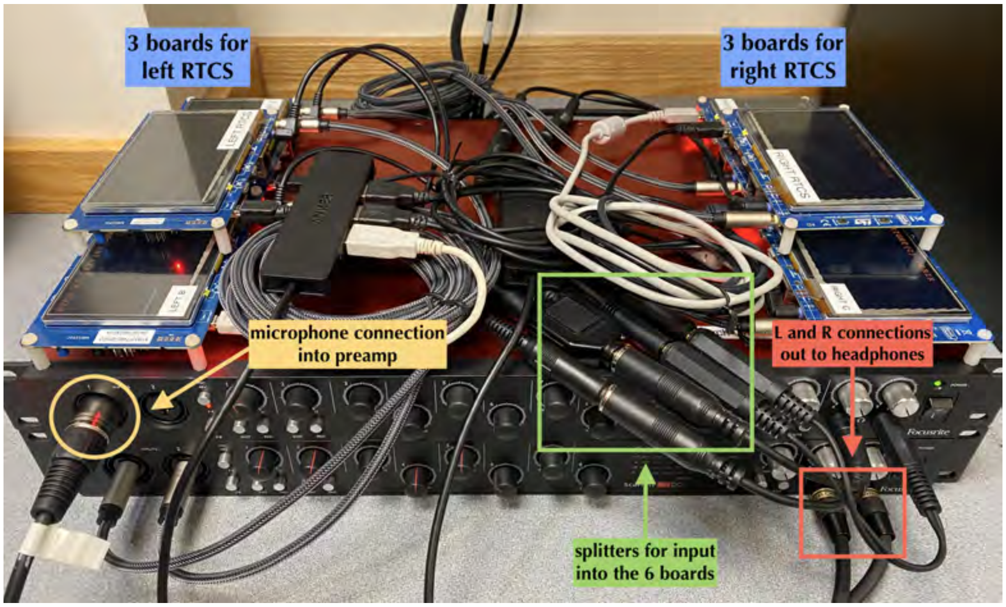
3 boards for left RTCS
3 boards for right RTCS
microphone connection into preamp
L and R connections out to headphones
splitters for input into the 6 boards

Figure 2.4 A picture of all the hardware involved in the RTCS which includes two Focusrite Scarletts (the top one acts as a preamp and the bottom one as a mixer) and six STM32F746 Discovery Boards (three for each left and right side).

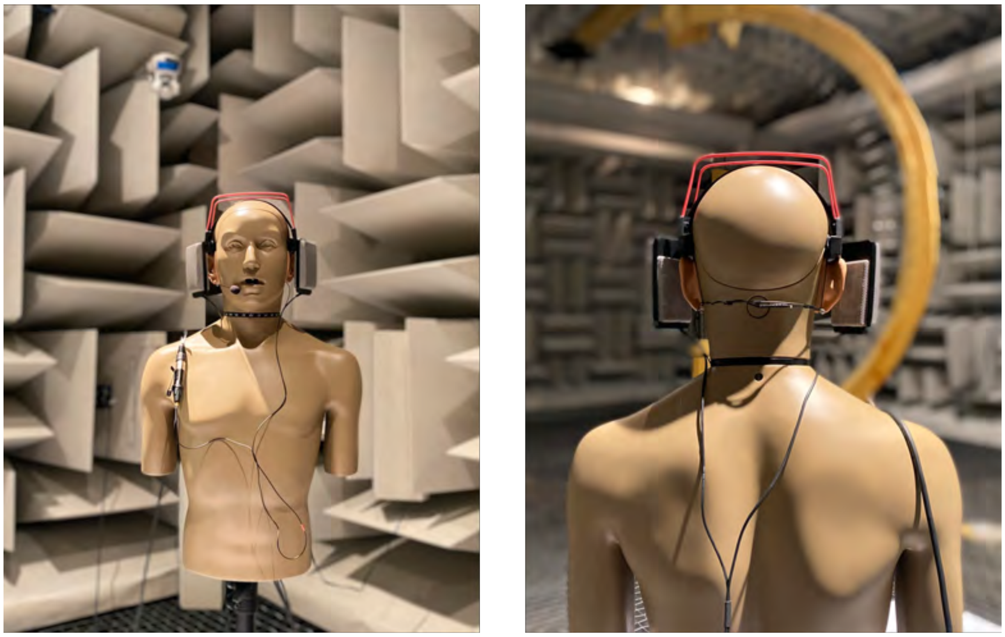

Figure 2.5 A G.R.A.S. head and torso simulator manikin (KEMAR 45BC) wearing AKG K1000 off-ear headphones and a DPA 4066 headset microphone used for the RTCS.

2.4 Latency

Total system latency (TSL) is defined as the time delay between the onset and the response of an event. In the context of a subject using the RTCS, TSL would be the time measured between the moment the microphone records a speech signal, to when the response signal is sent out to the headphones. Ideally, TSL should be minimized for any "real-time" system. However, a variety of unavoidable issues exist that increase TSL. In the case of the RTCS in Fig. 2.3, the minimum TSL is 3.6 ms. While a tiny fraction of this latency comes from passing analog signals through the Focusrites, the majority of this latency is due to the analog-to-digital and digital-to-analog conversion (ADC and DAC) on the STM32F746 boards. This portion of latency was measured by omitting the convolution computation on the boards and electrically looping audio data straight through the RTCS hardware; in other words, a chirp signal was input directly to the first Focusrite, this input was passed through the boards without any changes to the signal, output to the second Focusrite, and recorded after amplification. Custom in-house software named Easy Spectrum Time Reversal (ESTR), which was created by Kingsley *et al.* [39], was used to both generate the chirp signal used and record the output signal after amplification. Using cross correlation (a substitute for deconvolution) of the chirp with the output signal, the inherent TSL of 3.6 ms was determined.

Additional latency can be caused by the size of the buffer chosen to load in data to the ping-pong buffer. If a buffer size of x ms is chosen, it takes exactly x ms to record one block of speech data. This in turn adds an additional x ms to the inherent 3.6 ms latency resulting in a TSL of at least 3.6+x ms. As mentioned before, because convolution is a computationally expensive process it inevitably adds latency to any system. However, the processing time can be greatly reduced by choosing a shorter input signal and/or impulse response. The following subsections describe tests done to optimize the system's parameters, such as sampling frequency, input block size, and OBRIR length; these tests were done using one STM32F746 boards. After choosing the final values for these parameters, a method called partitioned convolution was used to finally achieve the right latency at output for this RTCS to function as a real-time system. To accomplish this with the limited hardware, more STM32F746 boards were added to help in the RTCS process.

2.4.1 Optimizing the RTCS Parameters

With block size kept constant, there are two system parameters that must be kept within a certain range in order to ensure a correct output signal from the system. These parameters are the OBRIR length and sampling frequency (f_s), which affect the time of the convolution computation. If these parameters are set to be outside their allowed range of values, the computational time for the convolution will exceed the block size causing increased latency and incorrect timing of the output signals. A longer OBRIR results in a more complex convolution computation and increase the computational time. Conversely, sampling rates that are too low result in slower data buffering for a fixed length OBRIR in time. Figure 2.6 is a plot of input block size versus latency of the RTCS when using a 100 ms long IR for different f_s. It shows that the higher f_s is, the lower latency is for various speech block sizes. Additionally, the shorter the block length, the lower the latency.

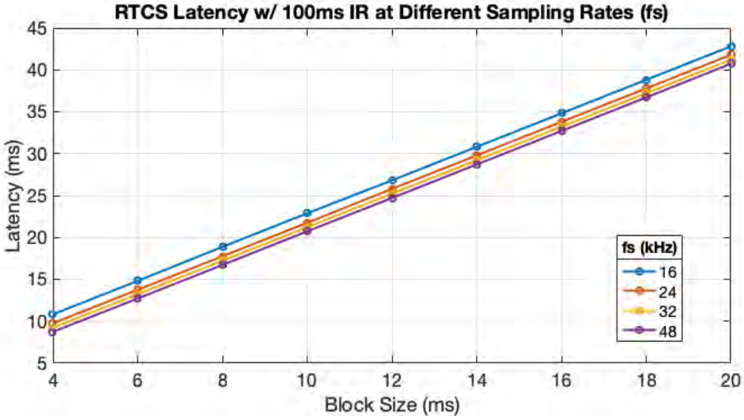

Figure 2.6 The measured RTCS latency for different input block sizes when using an IR of fixed length. The data shows that increasing the sampling rate decreases latency across all block sizes.

By setting the sampling rate to 48 kHz and block size to 4 ms on a single STM32F746 board, the minimum TSL achieved is 8.7 ms; the maximum OBRIR length allowed with these two parameter values is a little longer than 100 ms. To allow for a longer OBRIR, a lower sampling rate was considered. Because there is not much critical speech information above 10 kHz, a sampling rate of 24 kHz was tested on an STM32F746 board. At this sampling rate, frequencies up to the 12 kHz Nyquist frequency would still be correctly sampled by the RTCS making this the ideal minimum sampling rate to use. At a sampling rate of 24 kHz and same block size of 4 ms, the minimum TSL achieved is 1 ms higher, but allows for a maximum OBRIR length that is twice as long (200 ms).

According to the ANSI S12.60-2002 standard for Classroom Acoustics, the recommended RT60 inside a classroom should not exceed 0.6 s. However, in a study by Bistafa and Bradley [40], it was found that RT60s between 0.4 and 0.5 s are ideal for achieving 100% speech intelligibility in quiet classrooms. Furthermore, the Technical Committee on Architectural Acoustics of the Acoustical Society of America (ASA) stated that ideally classrooms should have RT60s in the range of 0.4-0.6 s [41]. Using these facts, Fig. 2.7 shows the decay of sound in classrooms in time for RT60s between 0.4 and 0.6 s. This plot illustrates that by using a 100 ms IR, only a maximum of 15 dB decay in initial sound levels can be achieved for a 0.4 s RT60. If a 200 ms IR is allowed, a 20 dB decay can be achieved for a 0.6 s RT60, a 25 dB decay for a 0.5 s RT60, and a 30 dB decay for a 0.4 s RT60; due to these higher decay levels achieved within the ideal RT60 range, the minimum IR/OBRIR length was chosen to be 200 ms.

Through these analyses, a sampling rate of 24 kHz and block size of 4 ms were ultimately chosen to allow for a 200 ms long OBRIR in the RTCS. The resulting TSL of the RTCS programmed on a single STM32F746 board with these parameter values is 9.75 ms.

During OBRIR measurements, KEMAR was kept at least 3 feet away from any large reflecting surface (walls, ceiling, floor, etc.). This location was to ensure that no reflections arrived at KEMAR's ear microphones earlier than 6 ms after the initial sound (using the approximation that sound travels around 1 ft/s). Because of this, the OBRIRs were truncated by 6 ms without loss of important information; as a result of this, the OBRIR sections used would be the 200 ms from 6-206 ms of the original, full-length OBRIRs. Due to this truncation, the RTCS can only allow for 6 ms of TSL when using these OBRIRs. To achieve a TSL of 6 ms, the 200 ms long IRs can be partitioned into nonuniform sections so that the convolution process can be parallelized. Correspondingly, parallelization achieved in the hardware by adding more STM32F746 boards to separately compute the convolution of each nonuniform sections.

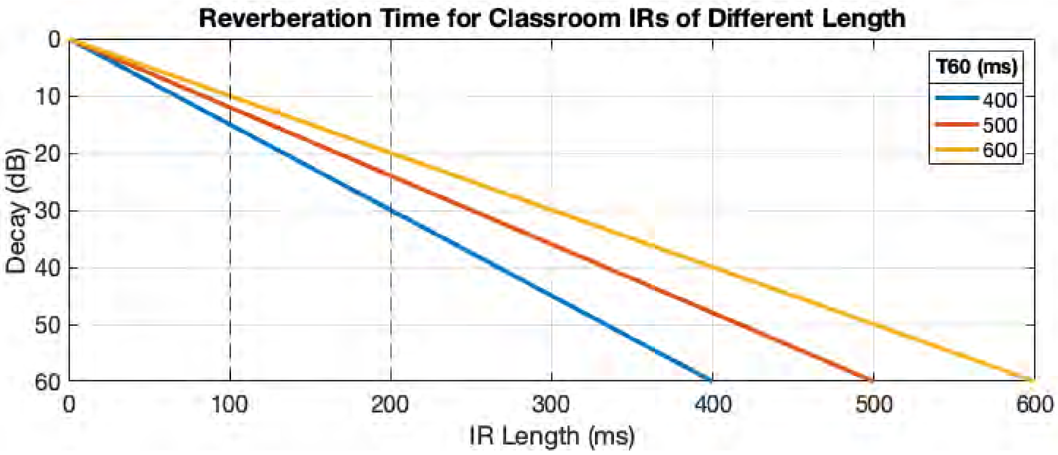

Figure 2.7 Decay of sound levels in time inside a classroom for different RT60 values. Shortening the IRs to 100 or 200 ms show that a maximum decay of 15 or 30 dB respectively can still be achieved.

2.4.2 Non-Uniform Partitioned Convolution

Non-uniform partitioned convolution is an efficient method for computing the real-time convolution of a long signal with a shorter signal by breaking the longer signal into smaller, manageable segments or partitions [42]. When the filter (in this case IR) is much longer than the input signal, the process of filter partitioning is used. Instead of performing a direct convolution that can be computationally intensive, the process involves dividing the IR into segments, convolving each segment separately with the input signal, and then combining the results. This approach reduces computational complexity and memory usage, making it particularly useful for real-time applications where quick processing is essential.

Because typical classroom IRs could have RT60s ranging from 0.3-0.7s, the length of IRs used in real-time room reverberation simulation systems like this could be long (a few seconds). Even if the input block size is chosen to be small (a few ms) and the FFT-based, overlap-save method is used to compute the convolutions, there will still be significant I/O delays due to the long IRs. In 1995, Gardner published a paper detailing the method of non-uniform partitioned convolution for low-latency filtering that is widely-known and used today [43]. The algorithm for non-uniform partitioned convolution used in this RTCS follows a similar approach to Gardner's. By first truncating the front of the IRs and then partitioning them into three separate IRs for convolution, the effects of the measured TSL (due to the convolution computations) can be eliminated at the system's output. Meaning, the RTCS can function as a real-time system to its users.

After accounting for the inherent TSL, only about 2.4 ms would be allowed for the convolution computation process; this was made possible by partitioning the OBRIR into three separate OBRIRs. As shown in Fig. 2.6, latency can be manipulated by changing the input block size. By making the first OBRIR be the first 1.7 ms of the original OBRIR and by changing the input block size to be 0.5 ms, a delay of 2.4 ms was "created"; adding the inherent delay (3.6 ms) to this results in a TSL of 6 ms to match the 6 ms truncation of the initial OBRIR. Then, since the second partitioned OBRIR would have to start 7.7 ms after the direct sound, it would need to have the appropriate length such that its convolution process would take 7.7 ms. The second OBRIR was chosen to be 3.3 ms long and the block size was kept at 0.5 ms to achieve this 7.7 ms delay. Lastly, the third OBRIR was set to be the remaining 195 ms, which started at 11 ms after the direct sound. Because the length of this third OBRIR was fixed, the block size was changed until a delay of 11 ms was achieved; this happened when the block size was set to be 3.8 ms. Figure 2.8 shows this partitioning visually on a fake OBRIR created. Since the first two OBRIRs are of shorter duration and the block size is also short, direct convolution was used in their computations. This was due to the fact that the overlap-save method is only efficient for long-duration signals. When the convolutions of these three partitioned OBRIRs are done in parallel and their results are concatenated in time, the overall result will output 6 ms after the direct sound. Therefore, by truncating and partitioning the OBRIRs in this specific manner, the inherent TSL and processing times do not delay the output of the RTCS. By using this method of partitioned convolution, this RTCS (using the specific OBRIRs measured) can function as a real-time system to its users and output responses 6 ms after the direct sound.

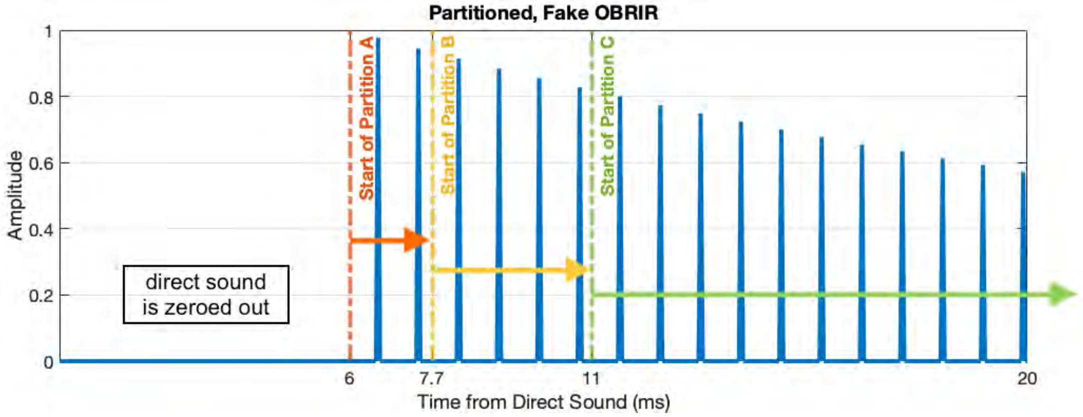

Figure 2.8 A plot of a fake oral binaural room impulse response (OBRIR) partitioned the same way as a real OBRIR that would be used in the RTCS. (Notice only the first 20 ms of an OBRIR are shown in this plot; in reality, the 6-206 ms portion of a real OBRIR is used in the RTCS.

2.5 STM32F746 Board Limitations

After optimizing the RTCS parameters and using multiple STM32F746 boards to achieve a truly real-time system, the gains on each piece of hardware used were adjusted. This step was essential to ensure that the output levels of the RTCS corresponded correctly with the input speech levels of the user. The different gains include that on the microphone preamp (to improve the signal-to-noise ratio (SNR)), the STM32F746 board's input (to avoid clipping) and output volume levels, and the headphone amplifier output level (to have correct output levels to the user's ears). To adjust all these gains to result in the correct input speech level to RTCS output level ratio, a head and torso simulator mentioned in 2.3.2, KEMAR, was used. Equipped with a precision microphone in each ear and a built-in loudspeaker at the mouth, KEMAR can measure the levels heard by a talker using the RTCS. Wearing the RTCS headphones and headset microphone, KEMAR was used to essentially remeasure the OBRIRs using feedback from the RTCS (with the real OBRIRs loaded in) rather than feedback from being placed in the physical classrooms. The OBRIRs were measured and calculated from KEMAR using a software program called EASERA (Electronic and Acoustic System Evaluation and Response Analysis) [44]; more details about how this is done is mentioned later in Section 3.2.2.2 of Chapter 3. The remeasured OBRIRs were then compared to the real OBRIRs, to ensure that the amplitude of the reflections (produced by the RTCS) had the same relative amplitudes as in the real OBRIRs with respect to the direct sound. To reach the correct output levels for both the left and right OBRIRs, the gains were adjusted and OBRIRs remeasured until they matched the amplitudes of the real OBRIRs.

After completing this step in getting the RTCS ready to use for vocal tests, background noise and perceptible buzzing that can be heard in the output speech was noticed. To try and combat these issues, a variety of tests were done. Each board was individually tested to make sure all produced the same output when given the same input and OBRIR. Although the same background noise and buzzing occurred in the output of each individual board, these issues were worsened when multiple board outputs were added/mixed together. Regular room IRs were loaded into the boards, but had the same noise output; this was to test whether the OBRIRs were the source of the noise. Artificial OBRIRs were created using bandpass-filtered Delta functions and used in the system, but although the noise levels could be decreased dramatically, a robotic buzzing in the output speech remained. These artificial OBRIRs tested were created to have a the same envelope as some real OBRIRs measured, but less reflection impulses were included to test whether multiple and densely-packed impulses were the problem. The background noise was found not to depend heavily on whether the OBRIRs were measured data or were artificially created, nor did the type of OBRIRs used change the buzzing quality of the output speech. Another possible noise source tested was whether the OBRIR peak amplitude values were too low, which was causing low SNR at output. The real OBRIRs contained reflections less than a tenth of the amplitude of the direct sound, so the OBRIRs were normalized and multiplied by larger numbers (from 1-10) to try and increase the SNR while performing convolution computations on the boards. However, doing this proportionately increased the noise levels at output. It was found that the left and right OBRIRs needed to be scaled correctly relative to each other, using the same normalization factor for both in order for their outputs to have the correct levels relative to each other (the use of independent normalization factors led to a noticeable increase in the background noise in the right ear compared to that in the left ear).

When consulting with the electrical engineers who helped in the RTCS development, they hypothesized potential issues that could be causing the buzzing sound in the output speech from the STM32F746 boards. One possible issue suggested was that processing synchronization across several boards, which is a very challenging architecture to work with in terms of dynamic range, could lead to the introduction of artifacts (like the background noise and buzzing quality of the speech) in the RTCS. Specifically, the ADC and DAC processes on the boards could be running on different clocks leading to slightly different latency in the output; without perfect alignment (in time) of the output signals from each board, phase discontinuity could be the cause of buzzing at the overall output.

To solve this issue in the current RTCS, it was suggested that the partitioned OBRIRs can be windowed or methods of digital synchronization/multiplexing can be introduced into the system. For future implementations of an RTCS, better hardware with synchronously clocked/started processes could be chosen. With more expertise and background, future researchers can implement an RTCS on specialized DSP hardware or FGPAs, that are more well-equipped to handle synchronization within real-time systems processes.

2.6 Conclusion

The overall goal of this research is to measure the effects of classroom acoustics on the vocal strain of teachers. To reduce variables introduced by moving subjects from room to room, a real-time convolution system (RTCS) was developed to create binaural auralizations from any measured acoustic environment in real time. Unlike other existing systems, this RTCS was developed without the use of commercial plugins to perform convolution in real time, and thus the whole process can be clearly reported on and controlled. Due to the fact that the system must function in real time, system latency was prioritized and ultimately minimized by taking the measures explained in this chapter. The algorithm performing convolution in real time was optimized by using an Fast Fourier Transform (FFT) based, overlap-save method for convolution along with filter partitioning. By using these methods, maximum speed and minimal error could be ensured during the data buffering process and convolution computation. System parameters that affected total system latency (TSL) the most were identified and optimized to make this RTCS function as a real-time system.

The RTCS was implemented on a total of six high-speed and high-efficiency microcontrollers developed by STMicroelectronics. Due to the limitations of these boards and the primarily the buzzing quality of the speech that was produced by them, the complete RTCS was not ready to be used for subject vocal strain testing. Perhaps with a deeper dive into the architecture of these STM32F746 boards, the sources of these issues could potentially be found and fixed to make this RTCS still a viable tool for vocal strain testing in the future.

Chapter 3

Vocal Strain Study

3.1 Introduction

The acoustics of a classroom can greatly impact the people talking inside: students and teachers. Teachers especially make up the largest group of professional voice users, which puts their vocal health at risk daily. When teaching in poor acoustic environments, teachers often strain their voice when trying to communicate clearly with their students. This prolonged and intense use of their voices has contributed to the high prevalence rates for voice disorders amongst teachers [45]. The American National Standards Institute (ANSI) for Classroom Acoustics only prescribes standards for background noise and reverberation time (RT60). Specifically, classrooms must have background noise no louder than 35 dBA and an RT60 no longer than 0.6 s (assuming the classroom has a volume less than 10,000 ft^3). To assess the direct impact of background noise and RT60, this study aims to measure the vocal strain of talkers as they experience talking in classroom environments with varying conditions. To quantify that relationship, artificial background noise or RT60 conditions can be generated within a classroom and corresponding changes in a talker's voice can be quantified by measuring changes in specific vocal strain parameters. The vocal parameters that are assessed for this study include voice sound pressure levels (SPL) (or loudness) in dB, fundamental frequency (F0), jitter (pitch stability), and shimmer (intensity stability).

This study includes conducting of preliminary vocal strain tests inside physical classrooms. The goal was to also conduct these inside the anechoic chamber using a completed real-time convolution system (RTCS). The purpose of doing tests in both conditions is to simulate the same conditions found inside the physical classrooms and perform the same vocal strain study on participants using the RTCS, and then to compare the results to those obtained from tests done inside physical classrooms. This comparison would help validate the accuracy of the RTCS in simulating real classroom environments. Therefore, the vocal strain study outlined below describes tests done inside two physical classrooms and describes the plan for their virtual counterparts through using an RTCS. As described in Chapter 2, ultimately the RTCS tests were not completed due to the very noticeable buzzing (machine-like) quality of the output speech.

3.2 Classroom Measurements

3.2.1 Measuring RT60 and Background Noise

In order to obtain the RT60 of a classroom, the impulse response (IR) must be measured first. In compliance with international standard ISO 3382, IRs were measured inside classrooms using a Bruel and Kjaer OmniPower 4292L dodecahedron loudspeaker as the omnidirectional source. The receiver used was a GRAS 46AQ 1/2" random-incidence microphone; the random-incidence nature of this microphone allowed for the overall response to be evenly-distributed (particularly above 10 kHz), i.e., sound was equally likely to arrive from all directions, making both the source and receiver omnidirectional. The loudspeaker was placed at the front of the classroom, where a teacher would typically stand, and the microphone was placed over 10 feet away where a student would be sitting.

A pink-weighted frequency sweep signal from 10 Hz to 20 kHz was played from the loudspeaker and the response recorded by the microphone was used to calculate the classroom's IR through inverse filtering and deconvolution. A software package called EASERA (Electronic and Acoustic System Evaluation and Response Analysis) [44], was used to generate the 5.5 second long chirp signal; this chirp was amplified by a Crown XLS1000 amplifier before being played through the dodecahedron loudspeaker. The microphone's response to this chirp was recorded through an RME Fireface UFX data acquisition system (DAQ) and then analyzed by EASERA. Using a sampling rate of 48 kHz and averaging over five runs, EASERA saved the final response signal and calculated the IR, while properly accounting for the use of the pink weighting of the sweep signal. The RT60 measurements of each classroom, that are mentioned in a later subsection, were calculated using EASERA.

The same GRAS 46AQ 1/2" random-incidence microphone, in conjunction with the RME Fireface UFX DAQ, was used to measure the background noise of each classroom. The noise recordings from this setup were processed through a MATLAB code to calculate the background noise levels in dBA. This code first A-weighted the recorded signal using the MATLAB Audio Toolbox, and then calculated the root mean square (RMS) value to obtain the overall background noise level. The background noise levels stated for each classroom, and its different classroom conditions, were all measured and calculated using this method.

3.2.2 Measuring Oral Binaural Room Impulse Responses (OBRIRs

For use in future studies involving a real-time convolution system (RTCS), OBRIRs were also measured in the classrooms involved in this study. An OBRIR is a measurement of how sound travels from a person's mouth to each of their ears within a specific room, capturing the acoustic characteristics of the space by also including the interactions with the room's surfaces and the listener's head and ears. Using these OBRIRs, an RTCS can simulate the experience of a person and how their voice would sound in a room, without the person physically being in the room. Conducting tests with this RTCS could allow for more classroom environments to be easily and quickly tested, including simulating artificial classroom conditions. For this purpose, OBRIRs were measured inside the two classrooms used in this study for use in future, conjunctive studies using an RTCS.

3.2.2.1 KEMAR Manikin

In order to measure OBRIRs accurately, a head and torso simulator (HATS), KEMAR (Knowles Electronics Manikin for Acoustic Research) [46] was used. The 45BC KEMAR created by GRAS is equipped with a precision microphone in each ear and a built-in loudspeaker at the mouth. The microphones used for this KEMAR are GRAS 40AO 1/2" prepolarized, pressure microphones, that are surrounded by pinnae simulators placed on the sides of the KEMAR head. The loudspeaker used as the mouth simulator, which had a noticeably uneven frequency response, was equalized to produce a signal from 100 Hz to 10kHz up to a level of minimum 100 dB re. 20 μPa. A photo of KEMAR can be seen in Fig. 3.1. Because KEMAR is shaped to have the same acoustical properties as an average human, it provides acoustic diffraction similar to that encountered around the human head and torso. For this reason, KEMAR was used to accurately measure the OBRIR of classrooms from a teacher's point of view at the front of the classroom.

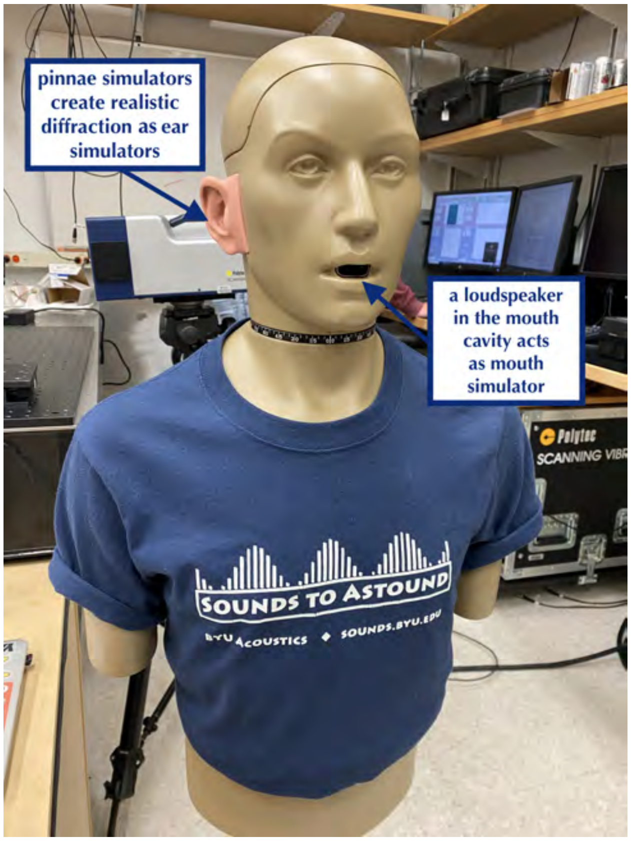
pinnae simulators create realistic diffraction as ear simulators
a loudspeaker in the mouth cavity acts as mouth simulator
SOUNDS TO ASTOUND
BYU ACOUSTICS ◆ SOUNDS.BYU.EDU
Polytec
SCANNING VIBR

Figure 3.1 A picture of the 45BC KEMAR manikin created by GRAS with arrows identifying the mouth and ear simulators.

3.2.2.2 Measurement Process

Measurements of OBRIRs in various classrooms followed a consistent procedure. KEMAR was placed in a standard teaching position at the front of each classroom (facing the student seat locations). KEMAR was also always placed 3 feet away from the closest wall (or any other large reflecting surface including the ceiling); this ensured that no major reflections arrived at KEMAR's ears prior to 6 ms after the direct sound arrival (assuming that sound travels at around 1 ft/ms). It was assumed that a teacher may normally be standing an arm's distance away from a wall, which is about 3 ft away. The same procedure and program used to measure regular IRs was also used to measure these OBRIRs. A 5.5-s, pink-weighted frequency sweep signal from 10 Hz to 20 kHz was output from the loudspeaker at KEMAR's mouth five times and the average response recorded was computed. An RME Fireface UFX preamp was used in conjunction with KEMAR's ear microphones to obtain the recordings. The OBRIRs were then calculated by EASERA.

3.2.2.3 Equalization of OBRIRs

After examining the resulting OBRIRs measured by KEMAR, a dominant frequency between 800-900 Hz was found to be visibly present in all the OBRIR time signals. In the published frequency response plot for KEMAR's mouth simulator, there is a clear spike in SPL at 800 Hz. Because of this, GRAS rcommends that the mouth simulator is equalized by first calculating your specific KEMAR's mouth simulator frequency response. Instructions are included in the manual for GRAS's KEMAR 45BC manikin. Following the recommended procedure, the frequency response of the mouth simulator on BYU's KEMAR was calculated and is shown in Fig. 3.2; this KEMAR's frequency response has a spike at 820 Hz, as expected from looking at the published curve. An inverse filter was made by taking the inverse of KEMAR's frequency response, and applying a bandpass filter from 80 Hz to 10 kHz while all other frequencies outside that range were zeroed out. It was then applied to all the OBRIRs in the frequency domain, preserving original phase information as well. Figure 3.2 also shows this inverse filter plotted against KEMAR's mouth frequency response. The code created to equalize all the OBRIRs is included in Appendix A. By equalizing all the OBRIRs, the classrooms' acoustical properties could be correctly represented.

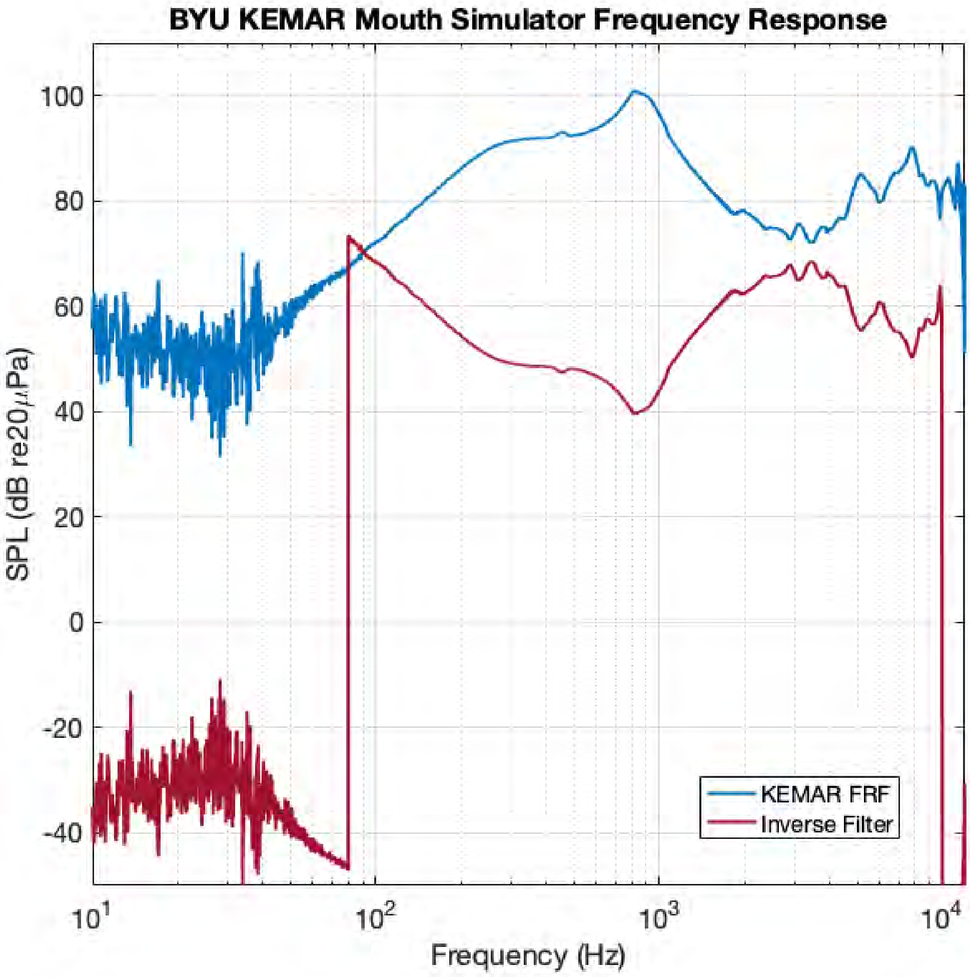

Figure 3.2 The frequency response funtion (FRF) from 10 Hz - 12 kHz of the mouth simulator (loudspeaker) on BYU's KEMAR manikin (in blue). This response was used to make an inverse filter (in red), to the remove the effects of KEMAR's mouth simulator on signals.

3.2.3 The Classrooms

3.2.3.1 Adding Background Noise to a Classroom

To test the relationship between vocal strain and background noise inside classrooms, a physical classroom that meets the ANSI S12.60-2002 standard for background noise was chosen. A photo of this classroom, room C247 inside the Eyring Science Center at BYU, is shown in Fig. 3.3; this 10.45 x 6.34 x 3.3 m room has a volume of about 218 m^3 (or 7,720 ft^3), RT60 of 0.65 s, and an ambient background noise level of 32 dBA. Two loudspeakers were placed 3 m apart and 5.5 m from a talker standing at the front of the classroom. During testing, generated white noise from Audacity was output from the loudspeakers at eight different levels between 38-56 dBA (measured near the talker's position). To simulate various background noise levels in C247 using an RTCS, new OBRIRs could be created by simply adding different white noise levels to the original OBRIR of C247; these new OBRIRs could then be loaded into the RTCS, and used in the convolution process.

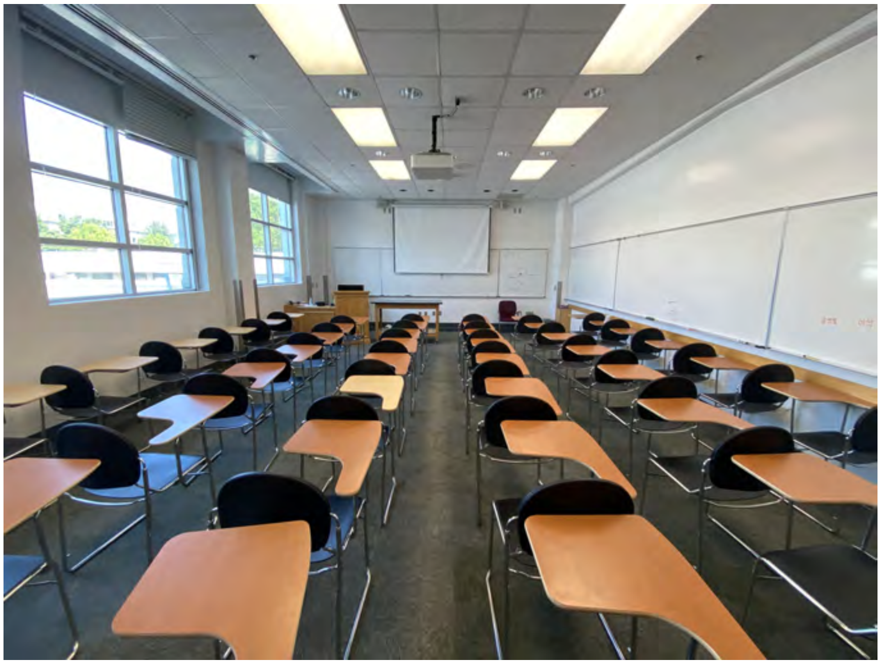

Figure 3.3 A photo of room C247 (taken from the back of the room) inside the Eyring Science Center at BYU.

3.2.3.2 Adding Absorption to a Reverberant Classroom

To test the relationship between vocal strain and RT60, a physical classroom that does not meet the ANSI S12.60-2002 standard for RT60 was chosen. A photo of this classroom, room N106 inside the Eyring Science Center at BYU, is shown in Fig. 3.4; this 5.8 x 8.56 x 3.15 m room has a volume of 156 m^3 (or 5,520 ft^3), RT60 of 0.78 s, and an ambient background noise level of 40 dBA. To dampen reflections in this classroom and thus decrease the RT60, a number of absorbing foam wedges were placed into the classroom. Each wedge was cut from 32 kg/m^3 of open cell polyether foam rubber with a 94.5 cm overall depth, a 30.5 by 30.5 cm base, and a profile similar to those suggested by Beranek and Sleeper [47]. Four different RT60s (including the original RT60 in the unmodified classroom) were measured in N106. With the addition of wedges the RT60s changed to: 0.60 s using 5 wedges, 0.54 s using 10 wedges, and 0.45 s using 20 wedges. The wedges were first placed on top of the tables in N106, then around the edges of the classroom on the floor. For each configuration of wedges that resulted in a different RT60, the OBRIR was measured (using KEMAR) to be used for vocal tests done with an RTCS.

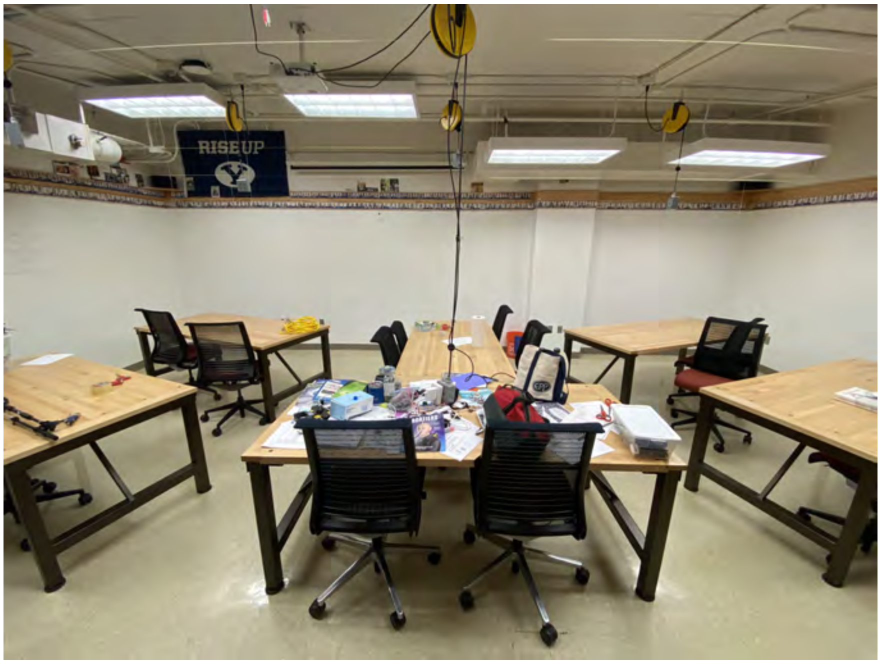

Figure 3.4 A photo of room N106 (taken from the back of the room) inside the Eyring Science Center at BYU.

3.3 Vocal Test Procedure

The vocal test procedure outlined below was used for tests done in both of the physical classroom environments and was planned to be used in the identical virtual classroom environments simulated through the RTCS. A total of five male undergraduate students (from Dr. Anderson's research group) between the ages of 22 and 29 years participated in this study as part of their research expectations to provide preliminary data.

3.3.1 Speech Tasks

For each generated classroom condition, a participant was asked to read aloud a text to the test administrator. These texts included the Rainbow Passage [48] and the first four lists from the

Harvard Sentences [49]. They are standardized texts most commonly used for evaluating speech, assessing articulation, and studying various aspects of spoken language. The Rainbow Passage contains all the sounds of American English and is specifically designed to cover a wide range of phonetic elements, including vowels, consonants, and various speech sounds. The Harvard Sentences, also known as the IEEE Sentences, were developed at Harvard University and MIT, and also designed to cover a wide range of phonetic contexts. Due to how short these reading passages/sentences are, participants only spoke for a few minutes at a time in each generated classroom condition. Appendix B.1 contains a copy of the full test procedure document used to guide tests done inside C247 and N106, and Appendices B.2 and B.3 include the whole Rainbow Passage and a sample of the Harvard Sentences, respectively.

3.3.2 Setup

Speech from the participants was recorded using a GRAS 46AQ 1/2" random-incidence microphone; the microphone was placed at least 0.5 m away from the participant's mouth and thus out of their air stream. The signal from the microphone was amplified through an RME Fireface UFX and saved using the recording application software Audacity. Each recording was sampled at 44.1 kHz with a 16 bit rate and saved as a .wav file to meet the correct formatting for vocal strain analysis.

3.3.3 Vocal Strain Analysis

In this study, vocal strain is quantified by tracking changes in the following parameters: overall voice SPL (measured in dB), fundamental frequency (F0), jitter, and shimmer. Vocal effort is defined as an individual's physical and/or mental exertion involved in their vocal production; vocal strain is defined as the physical discomfort experienced by individuals who report high vocal effort for sustained durations of time. Increases in these chosen parameters have often been associated with increased vocal effort due to background noise levels [13] and RT60 within a room [50].

Specifically, when speaking in the presence of noise, a person's voice SPL increases as background noise levels increase [51]. This phenomenon, known as the Lombard effect, has been found to start at a background noise level of 43.3 dBA [52]. Other studies have shown that vocal fatigue, which can be caused by speaking in noisy or reverberant conditions over a long period of time, leads to increased F0, jitter, and shimmer values [53]. Vocal fatigue is often characterized by stiffened vocal folds, which can increase the rate of vocal fold vibrations (increasing F0) and decrease control or stability of the fold vibrations (leading to increased jitter and shimmer) [54].

The vocal strain parameters mentioned above are calculated from recordings of the participants in this vocal strain study. The vocal strain analysis for this project is done by speech scientists working with Dr. Eric Hunter at The University of Iowa who are partners on the overall research effort that is funding the work described in this study. Using the same Statistical Package for the Social Sciences (SPSS) software used in past studies [19, 20], Hunter and his group can easily process speech recordings using analysis programming code to obtain various parameters quantifying vocal strain. The parameter values shown in later sections were calculated by Hunter's group.

To quantify the overall voice SPLs, the mean SPL of only the voiced segments (determined by the analysis code) is calculated (in dB). The mean F0 is calculated in semitones referenced to 120 Hz (the F0 of the average male voice) for only the voiced segments. The frequency corresponding to n semitones from a reference frequency (f_{ref}) can be calculated using Equation 3.1.

$$f = f_{ref} \times 2^{n/12} \tag{3.1}$$

Jitter is calculated by taking the average absolute difference between consecutive pitch periods and dividing it by the average pitch period, then multiplying by 100 to express the result as a percentage. A small jitter percentage indicates a relatively stable pitch, while a larger percentage suggests greater pitch variation, which usually signifies vocal fatigue. Shimmer is calculated similarly, but by using the 11-point Amplitude Perturbation Quotient (APQ11). The APQ11 shimmer value is calculated by taking the average absolute difference between the amplitude of a single vocal period and the average amplitude of its ten surrounding periods, then dividing that difference by the average amplitude of the entire voice sample. Higher shimmer percentages indicate greater amplitude fluctuations, which is usually caused by vocal fatigue. Jitter and shimmer are meant to indicate vocal strain or fatigue, which occurs after the talker has been exerting high vocal effort for an extended period of time. Since the talkers in this preliminary study only spoke for a couple of minutes, noticeable increases in jitter and shimmer were not expected to be observed.

3.4 In-Classroom Test Results

The following subsections discuss the results of the vocal strain tests done inside the physical classrooms chosen for this study. Measurements of each vocal strain parameter is shown in plots, and their trends are explained for each test condition type.

3.4.1 Background Noise Tests

Room C247 was used to test the effects of different background noise levels on the four vocal strain parameters chosen. Figure 3.5 shows an image of a participant and test administrator inside C247 during a background noise test. Figure 3.6 includes four plots showing the results from tests done inside C247 as background noise was increased. As expected, the mean SPL loudness and mean F0 increased as the background noise became louder; this is shown in the top two plots of Fig. 3.6. In accordance with the findings of Bottalico *et al.*, vocal effort (quantified in this study by mean loudness and mean F0) increased at a high rate after the background noise was increased to be louder than the Lombard effect change-point at 43.3 dBA. Below this level, vocal effort appears to be unaffected.

Regarding jitter and shimmer, existing studies mentioned previously show that both vocal parameters have a tendency to increase in people experiencing vocal strain or fatigue. However, the results shown in the bottom two plots of Fig. 3.6 indicate not much change and perhaps even a minor opposite effect. To avoid overexertion of the vocal folds in preliminary testing, the participants in this study did not speak in noisy conditions for more than a few minutes at a time. Because of this, they may not have experienced vocal fatigue serious enough to cause a clear decrease in jitter and shimmer. What can be deduced from these results, however, is that jitter and shimmer decreased because the participants were trying to produce clearer speech. As the background noise was increased, the participants responded by enunciating so that they could be heard more clearly. More vocal control leads to a decrease in pitch variation and amplitude fluctuation, which caused jitter and shimmer to decrease momentarily with increasing background noise. If the participants were asked to continue speaking in these noisy conditions for a longer period of time, an increase in jitter and shimmer would be expected.

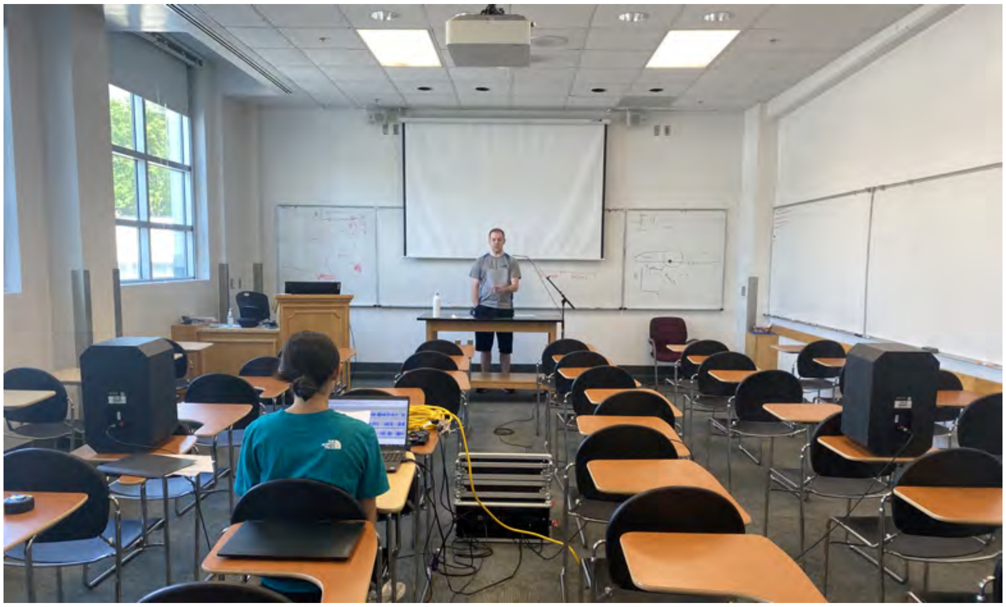

Figure 3.5 An image of vocal strain tests being administered in room C247. The participant is standing at the front of the classroom and spoke near an elevated microphone. Loudspeakers on both sides of the room output background noise at successively increased levels during testing.

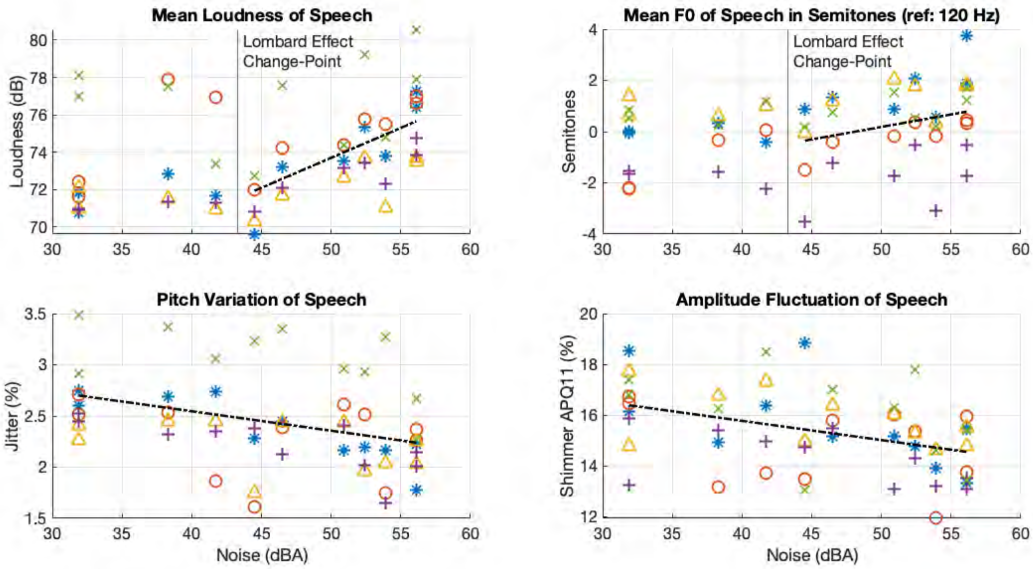

Figure 3.6 Four plots showing the trends of the four vocal parameters as background noise levels were changed. Each marker type represents a different test participant, and the dashed black lines are linear fits to the mean of the vocal parameters analyzed with each background noise condition. Notice in the top two plots, a vertical line signifying the Lombard effect change-point dBA level is added to show when the mean loudness and fundamental frequency (F0) increase.

3.4.2 RT60 Tests

Room N106 was used to test the effects of different RT60s on the four vocal strain parameters chosen. Figure 3.7 shows an image of a participant, test administrator, and a number of yellow foam wedges inside N106 during an RT60 test. Figure 3.8 includes four plots showing the vocal analysis results from tests done inside N106 as the RT60 was changed by varying the number of wedges place inside. Looking at the plots, the measured vocal parameters do not correlate linearly with RT60 as they did with background noise; for this reason, the data on each plot was fitted to a third-degree polynomial.

Starting with the top left plot in Fig. 3.8 of mean loudness, the natural response of increasing one's volume in more reverberant environments was followed. When a room's RT60 is high, sound in that room persists for a longer period of time and compete with the direct sound of subsequently spoken speech. People trying to communicate in a reverberant environment tend to increase their speech volume to be heard over the reflected sound; this has been determined as response to the Lombard effect, since higher reverberance can make an environment sound more noisy [55]. The top right plot shows a slight decrease in mean F0 when the RT60 is higher. In reverberant environments, a talker might sense difficulty in being understood. Lowering their pitch can be associated with an increase in vocal intensity and projection, making their voice more prominent against background noise [56].

Referring to the bottom two plots of Fig. 3.8, there is again a lot of variation in the results. One possibility for this variability is that in more reverberant environments, the reflected sound waves mix with the direct sound. This mixing can result in inaccurate jitter or shimmer values because the software may misinterpret reverberation artifacts and amplitude variations as irregularities in vocal production. Another realistic possibility for the variation seen in this data is an insufficient number of RT60 conditions tested.

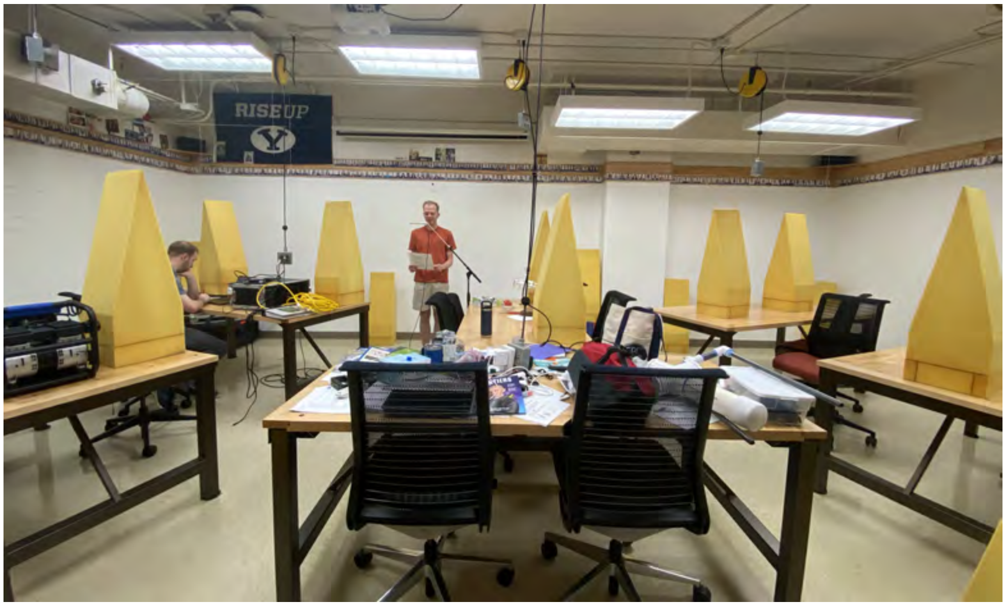

Figure 3.7 An image of vocal strain tests being administered in room N106. The participant is standing at the front of the classroom and spoke near an elevated microphone. Yellow foam wedges were placed around the room to decrease reverberation.

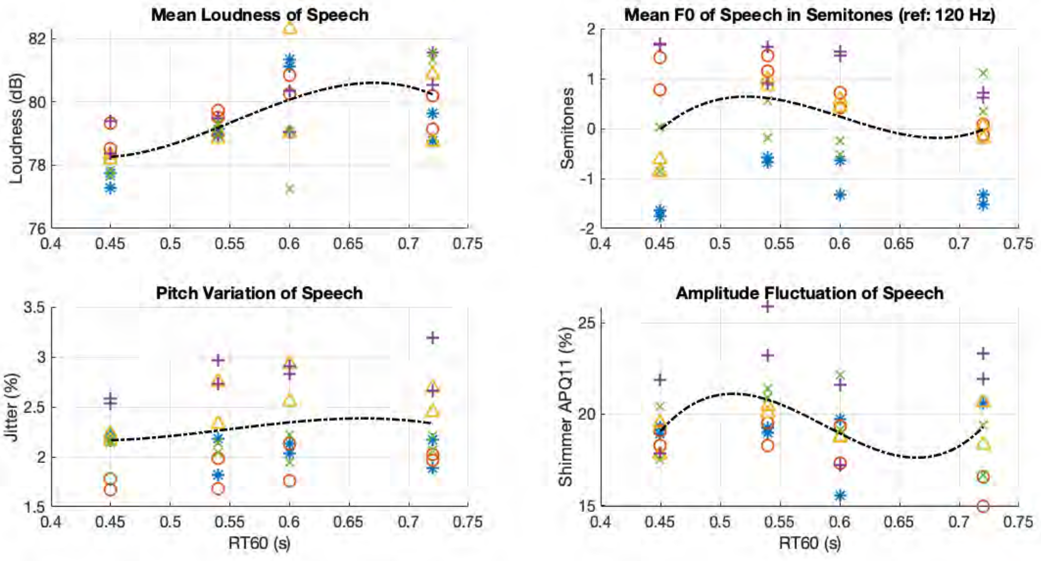

Figure 3.8 Four plots showing the trends of the four vocal parameters as reverberation time (RT60) was changed. Each marker type represents a different test participant, and the dashed black lines are third-degree polynomial fits to the mean of the vocal parameters analyzed with each RT60 condition.

3.5 Vocal Strain Tests Using An RTCS

Vocal strain tests, like the ones done in this study, repeated using an RTCS would follow a similar procedure, including recording the speech from the talker at a microphone 0.5 m away (this recording would be analyzed for vocal strain). The main difference would be the loss of a participant's visual perception of a classroom, because these vocal strain tests would be administered inside an anechoic chamber. Specifically, a participant would be asked to sit inside an anechoic chamber and wear the headphones and headset microphone required to use the RTCS. Also inside with them would be a test administrator, who would provide cues and guidance for each speaking exercise done in each virtual environment simulated through the RTCS. The test administrator inside would also act as a target for the participant to communicate clearly to through each acoustic condition they experience.

Meanwhile, another person outside the chamber (in a control room) would change the OBRIRs loaded into the RTCS to change the virtual classroom environment; this person would have constant communication with the test administrator inside the chamber, to ensure that the system is changed at the appropriate times. As mentioned before, since participants being tested with the RTCS lose their visual perception of the classrooms, projectors can be used inside the anechoic chamber to show images of the simulated classrooms to them. Accounting for these differences, the same vocal tests administered in this study can be easily done using an RTCS.

3.6 Conclusion

Because teachers are required to speak inside acoustically varying classroom environments daily, it is important to study the physical factors of the classrooms that effect them most. The objective of this simple vocal strain study was to show the specific effects of a classroom's background noise and RT60 on a person speaking inside. The study results indicated that participants' primary responses generally aligned with the expected outcomes in each generated acoustic environment. However, due to the small pool of participants and range of conditions tested, a large spread in the results was seen. To detect trends by minimizing the spread frequently found in repeatable speech measurements, a larger number of test participants and a more specific demographic amongst the participant group is encouraged for future tests. In addition, a larger (but denser) range of acoustic conditions can be tested to provide more data to see clearer trends in the results.

Chapter 4

Conclusion

The overall goal of this research effort was to create a real-time convolution system (RTCS) without the use of commercial audio plugins or algorithms to compute the convolution in real time. An accurate RTCS could then be used for quicker and more convenient vocal effort and/or strain testing, as participants would not be required to be physically moved to different acoustic environments to be tested. Using an RTCS for these tests would allow testing to be done inside an anechoic chamber while differing acoustic environments could be presented to a talker using the system.

In order to develop a truly real-time functioning system like this, many parameters had to be studied and controlled. Additionally, because the oral binaural room impulse responses (OBRIRs) used and the real-time speech contained many samples, the system's algorithm had to be capable of the computational load from the convolution. First, a fast method for real-time convolution computation was chosen; this included the use of block convolution and the overlap-save method, which are common and efficient processes used for the convolution of long-duration signals. Then, an ideal software platform and its programmed hardware counterparts were chosen, which led to the use of multiple STM32F746 Discovery boards by STMicroelectronics and two Focusrite Scarlett 18i20 audio interfaces. After assessing the latency of the RTCS (with measured OBRIRs) with this hardware, it was determined that in order to achieve real-time processing, the convolution computation would need to be parallelized across a total of six boards (three for each OBRIR used, left and right). To correctly implement this across multiple boards, a method called partitioned convolution was used. After adjusting system parameters like sampling frequency, input block size, and partition lengths, this method proved to accurately compute the convolution of a partitioned OBRIR with real-time speech. All of this work culminated to the successful development of an RTCS that truly functions in real time.

Unfortunately, as the RTCS was being tested to be used in vocal strain tests, high levels of background noise and a significant buzzing-speech sound were found in its output. Multiple tests were tried to reduce these issues, including making artificial OBRIRs that mimicked the shape and decay of real OBRIRs used. Although this led to a decrease in the background noise, issues remain with the convolution computation on the STM32F746 boards that continue to introduce artifacts into the RTCS output. Because of limited knowledge of the full architecture of these boards, a stopping point in the RTCS improvement was reached. Perhaps higher quality boards with greater dynamic range need to be used in future implementations of the RTCS, or a deeper dive into the real-time convolution algorithm can be done to ensure no errors in quantization occur at output.

In conjunction with the RTCS development and improvement, in-classroom vocal strain tests were done. The hope was to administer vocal strain tests in physical classrooms, as well as in their virtual counterparts through using the RTCS; this would act as a way to test the validity and accuracy of the RTCS by comparing the results from the physical tests to the virtual ones. However, because the RTCS was not in the right condition to be used for vocal strain tests, only the results from the in-classroom tests were reported. Although the results proved to follow the expected outcomes within each generated acoustic environment, they also showed the limitations of the tests conducted. Specifically, the large spread seen in the results was thought to be due to an insufficient number of test participants and conditions tested. With this knowledge, future vocal strain tests can be redone to include more participants (with Institutional Review Board (IRB) approval) and a larger range of acoustic conditions tested. When the current RTCS is diagnosed and improved, the same tests can be repeated using the RTCS.

Appendix A

OBRIR Equalization Function

The following is a MATLAB function written to equalize OBRIRs measured using the KEMAR manikin. Equalization is done to remove the effects (in the frequency domain) cause by KEMAR's mouth simulator. Input variables to this function include:

- *K_resp* : the time domain response of a chirp played through KEMAR's mouth and recorded by its ear microphones
- *K_fs* : the sampling frequency of *K_resp*
- *h0* : the impulse response that needs to be equalized
- *fs0* : the sampling frequency of *h0*

```matlab
function h_equ = Equalize(K_resp, K_fs, h0, fs0)

fft_K = fft(K_resp); fft_h0 = transpose(fft(h0));
K_inv = 1./fft_K; % Find inversion filter

N = length(K_inv);
f = (0:N-1)*(K_fs/N);
% Design the anti-aliasing filter in the frequency domain
f_low = 80; % Lower cutoff frequency
f_high = 10000; % Upper cutoff frequency
filter = zeros(1,N);
filter(f >= f_low & f <= f_high) = 1;
filter(f >= K_fs-f_high & f <= K_fs-f_low) = 1; % Mirror for negative frequencies

% Apply the anti-aliasing filter
K_aa = K_inv.*filter;

%%%%%%%%%%%% Resample in the frequency domain %%%%%%%%%%%%
% Create a new frequency axis for the resampled signal
N_new = round(N * fs0 / K_fs); % Number of points in resampled signal

% Initialize the new frequency domain representation
X_new = zeros(1, N_new);

% Copy the appropriate frequency components to the new frequency representation
if N_new > N
    % Upsampling: copy all frequencies and pad with zeros
    X_new(1:N) = K_aa;
else
    % Downsampling: truncate the frequencies
    X_new = K_aa(1:N_new);
end

% Make FFT of KEMAR response the same length as FFT of h0
invK = interp1(1:length(X_new), abs(X_new), linspace(1,length(X_new),length(fft_h0)));

H_equ_mag = abs(fft_h0).*abs(conj(invK)/norm(invK)^2); % Create equalized h0
IRphase = angle(fft_h0);
H_equ = H_equ_mag.*exp(1i*IRphase);
h_equ = ifft(H_equ,’symmetric’);
```

Appendix B

Vocal Strain Study Components

B.1 Vocal Strain Study Test Procedure

1. One participant is chosen to perform a series of speaking exercises under varying classroom conditions.
2. Choose a classroom
   (a) **C247**: white noise is generated in Audacity (amplitude = 0.8) and output into the room through two loudspeakers, levels vary between 38-56 dBA (defined by certain volume levels on the laptop used)
   (b) **N106**: anechoic wedges are added to the room to decrease the reverberation time from 0.7 to 0.4 s

(The following steps pertain to tests done for increasing levels of background noise)

1. RECORD regular conversation with the assigned talker (no background noise) as a warm-up! Ask them to:
   (a) Introduce themselves (name only)

   (b) Practice reading the first paragraph of the Rainbow Passage out loud

   (c) Practice reading List 3 of the Harvard Sentences

2. The assigned talker will start by reading the Rainbow Passage.

   (a) Increase volume to 6 before they begin reading

   (b) Increase volume to 12 at “There is, according to legend...”

   (c) Increase volume to 18 at “The Norsemen considered...”

   (d) Increase volume to 24 at “The actual primary rainbow observed...”

   (e) PAUSE at the end (no talking) then STOP recording

3. The assigned talker will then read the Harvard Sentences (in sets of TEN)

   (a) Tell them to PAUSE briefly in between sets

   (b) After List 1, increase volume to 8

   (c) After List 2, increase volume to 16

   (d) After List 3, increase volume to 24

   (e) Read List 4

   (f) PAUSE at the end (no talking) then STOP recording (mute noise)

4. (Bring volume back down to ZERO.) The assigned talker will now do some automatic speech exercises. Do a separate recording for each task below:

   (a) Recite days of the week.

   (b) (Increase volume to 10) Count from 1 to 20

   (c) (Increase volume to 20) Recite months of the year

After tests in C247, recordings should be split up in time by the different noise conditions. For ease, the test procedure can be rewritten to allow time for starting new recordings in new condition (which is done in the N106 tests).

(The following steps pertain to tests done while decreasing RT60 using anechoic wedges)

1. The assigned talker will start by reading the Rainbow Passage.
   - (a) Make each reading a SEPARATE recording
   - (b) Begin reading w/ NO wedges inside the room
   - (c) Pause right before "People look, but no one..." and add 5 wedges (one on each table)
   - (d) Pause right before "The Norsemen considered..." and add 5 more wedges (one on each table again, so 2 total/table)
   - (e) Pause right before "The difference in the rainbow..." and add 10 wedges (5 along the front and 5 along the back, in front of doors and in corners)
2. The assigned talker will then read the Harvard Sentences (in sets of TEN), starting in ambient conditions.
   - (a) Read List 1 then add 5 wedges (one on each table)
   - (b) Read List 2 then add 5 more wedges (one on each table again, so 2 total/table)
   - (c) Read List 3 then add 10 wedges (5 along the front and 5 along the back, in front of doors and in corners)
   - (d) Read List 4

B.2 Rainbow Passage

When the sunlight strikes raindrops in the air, they act as a prism and form a rainbow. The rainbow is a division of white light into many beautiful colors. These take the shape of a long round arch, with its path high above, and its two ends apparently beyond the horizon. There is , according to legend, a boiling pot of gold at one end. People look, but no one ever finds it. When a man looks for something beyond his reach, his friends say he is looking for the pot of gold at the end of the rainbow.

Throughout the centuries people have explained the rainbow in various ways. Some have accepted it as a miracle without physical explanation. To the Hebrews it was a token that there would be no more universal floods. The Greeks used to imagine that it was a sign from the gods to foretell war or heavy rain. The Norsemen considered the rainbow as a bridge over which the gods passed from earth to their home in the sky. Others have tried to explain the phenomenon physically. Aristotle thought that the rainbow was caused by reflection of the sun's rays by the rain.

Since then physicists have found that it is not reflection, but refraction by the raindrops which causes the rainbows. Many complicated ideas about the rainbow have been formed. The difference in the rainbow depends considerably upon the size of the drops, and the width of the colored band increases as the size of the drops increases. The actual primary rainbow observed is said to be the effect of super-imposition of a number of bows. If the red of the second bow falls upon the green of the first, the result is to give a bow with an abnormally wide yellow band, since red and green light when mixed form yellow. This is a very common type of bow, one showing mainly red and yellow, with little or no green or blue.

B.3 Harvard Sentences

The following is List 1 out of 72 lists that make up all the Harvard Sentences. For vocal strain tests, Lists 1-4 of the Harvard Sentences were used.

List 1

1. The birch canoe slid on the smooth planks.
2. Glue the sheet to the dark blue background.
3. It's easy to tell the depth of a well.
4. These days a chicken leg is a rare dish.
5. Rice is often served in round bowls.
6. The juice of lemons makes fine punch.
7. The box was thrown beside the parked truck.
8. The hogs were fed chopped corn and garbage.
9. Four hours of steady work faced us.
10. A large size in stockings is hard to sell.

Appendix C

Real-Time Convolution Algorithm

The following is a C source file of the real-time convolution algorithm used in the RTCS. As mentioned at the top of the code, the code was originally written by Dr. Brian Mazzeo from BYU's Electrical and Computer Engineering Department. The following notes are included to provide a guide to this code:

- **Line 29** is where the input block size is defined (in samples) and was changed for each partitioned OBRIR used.
- **Line 35** is where the type of convolution computation is specified. As mentioned in Ch. 2 Section 2.4.2, direct convolution was used for the first two shorter OBRIR convolution computations (Case 1), while overlap-save was used for the convolution computation of the last OBRIR computation (Case 4).
- **Lines 165-169** are code used to compute the direct convolution using a finite impulse response (FIR) filter.
- **Lines 277-315** are code used to compute the convolution using the overlap-save method.

```c
/**
  ******************************************************************************
  * @file    signal_processing.c
  * @author  Brian Mazzeo
  * @date    2023
  * @brief   This file provides a set of code for signal processing in 487.
  *          Parts are taken from example code from STMicroelectronics
  ******************************************************************************
  * @attention
  *          This code was specifically developed for BYU ECEn 487 course
  *          Introduction to Digital Signal Processing.
  *
  ******************************************************************************
  */

#include <stdio.h>
#include <stdint.h>
#include "stm32746g_discovery_lcd.h"
#include "signal_processing.h"
#include "stm32746g_discovery.h"
#include "arm_math.h"
#include "arm_const_structs.h"
#include "filter_coefficients.h"

/* ----------------------------------------------------------------------
** Defines for signal processing
** ------------------------------------------------------------------- */

#define AUDIO_BLOCK_SAMPLES             ((uint16_t)200)         // Number of samples (L and R) in audio block (each samples is 24 bits)
#define DFT_SIZE                        1024
#define DFT_CMPLX_DATA_SIZE             2048
#define FFT_TYPE                        arm_cfft_sR_f32_len1024

/* For Lab Exercise */
#define Lab_Execution_Type              4

// For the overlap-save method
#define L_chan_overlap_save_start_index (DFT_CMPLX_DATA_SIZE - (2*AUDIO_BLOCK_SAMPLES))


/* FIR Solution */
/* Important to have the structure outside of the execution so it can be initialized */
arm_fir_instance_f32 L_chan_FIR;
arm_fir_instance_f32 R_chan_FIR;

/* FFT Overlap-Add Solution */
static float32_t L_chan_FIR_state[AUDIO_BLOCK_SAMPLES + NUM_TAPS -
```

```
    1];
static float32_t R_chan_FIR_state[AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1];

float32_t FIR_response[DFT_CMPLX_DATA_SIZE * 2];
float32_t L_chan_data_for_DFT[DFT_CMPLX_DATA_SIZE];
float32_t R_chan_data_for_DFT[DFT_CMPLX_DATA_SIZE];

float32_t L_chan_DFT_cmplx[DFT_CMPLX_DATA_SIZE];
float32_t R_chan_DFT_cmplx[DFT_CMPLX_DATA_SIZE];

float32_t L_chan_DFT_mult_FIR_cmplx[DFT_CMPLX_DATA_SIZE];
float32_t R_chan_DFT_mult_FIR_cmplx[DFT_CMPLX_DATA_SIZE];

float32_t L_chan_data_from_IDFT[DFT_SIZE];
float32_t R_chan_data_from_IDFT[DFT_SIZE];

// Dummy to get filled
float32_t chan_data_overlap[DFT_CMPLX_DATA_SIZE];


/* FUNCTION DEFINITIONS BELOW */

/**
  * @brief  Initialize filter structures to be used in loops later
  * @retval None
  */
void initialize_signal_processing(void) {

  switch (Lab_Execution_Type)
  {
  case 0: // Passthrough case
    break;

  case 1: // FIR case
    /* Call FIR init function to initialize the instance structure. */
    arm_fir_init_f32(&L_chan_FIR, NUM_TAPS, (float32_t *) &Filter_coeffs[0], &L_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES);
    arm_fir_init_f32(&R_chan_FIR, NUM_TAPS, (float32_t *) &Filter_coeffs[0], &R_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES);
    break;

  case 2: // FFT Overlap-add
    arm_fill_f32(0, &FIR_response[0], DFT_CMPLX_DATA_SIZE); // Response may not be as long as DFT and so need to zero-pad

    // Place the filter coefficients in the real positions for the DFT
    for (uint32_t i = 0; i<NUM_TAPS; i++) {FIR_response[i*2] = Filter_coeffs[i];}

    arm_cfft_f32(&FFT_TYPE, &FIR_response[0], 0, 1); // In-place computation on FIR_response (now FFT'd)

```

```c
    /* Initialize the Left and Right channel data - zeroed */
    arm_fill_f32(0, &L_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);
    arm_fill_f32(0, &R_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);

    /* Initialize the overlap-add FIR states */
    arm_fill_f32(0, &L_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    arm_fill_f32(0, &R_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    break;

  case 3: // FFT Overlap-add with real-imag efficiency
    arm_fill_f32(0, &FIR_response[0], DFT_CMPLX_DATA_SIZE); // Response may not be as long as DFT and so need to zero-pad

    // Place the filter coefficients in the real positions for the DFT
    for (uint32_t i = 0; i<NUM_TAPS; i++) {FIR_response[i*2] = Filter_coeffs[i];}

    arm_cfft_f32(&FFT_TYPE, &FIR_response[0], 0, 1); // In-place computation on FIR_response (now FFT'd)

    /* Initialize the Left and Right channel data - zeroed */
    arm_fill_f32(0, &L_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);
    arm_fill_f32(0, &R_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);

    /* Initialize the overlap-add FIR states */
    arm_fill_f32(0, &L_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    arm_fill_f32(0, &R_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    break;

  case 4: // FFT Overlap-save with real-imag efficiency
    arm_fill_f32(0, &FIR_response[0], DFT_CMPLX_DATA_SIZE); // Response may not be as long as DFT and so need to zero-pad

    // Place the filter coefficients in the real positions for the DFT
    for (uint32_t i = 0; i<NUM_TAPS; i++) {FIR_response[i*2] = Filter_coeffs[i];}
    //FIR_response[0] = 1;

    arm_cfft_f32(&FFT_TYPE, &FIR_response[0], 0, 1); // In-place computation on FIR_response (now FFT'd)

    /* Initialize the Left and Right channel data - zeroed */
    arm_fill_f32(0, &L_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);
    arm_fill_f32(0, &chan_data_overlap[0], DFT_CMPLX_DATA_SIZE);

    break;
  }
}

/**
  * @brief  Process audio channel signals
  * @param  L_channel_in: Pointer to Left channel data input (float32_t)
  * @param  R_channel_in: Pointer to Right channel data input (float32_t)
  * @param  L_channel_out: Pointer to Left channel data output (float32_t)
  * @param  R_channel_out: Pointer to Right channel data output (float32_t)
  * @param  Signal_Length: length of data to process
  * @retval None
  */

void process_audio_channel_signals(float32_t* L_channel_in, float32_t* R_channel_in, float32_t* L_channel_out, float32_t* R_channel_out, uint16_t Signal_Length)
{
        char buf[70];
    BSP_LCD_SetFont(&Font8);
    BSP_LCD_SetTextColor(LCD_COLOR_CYAN);
    sprintf(buf, "Processing Signals" );
    BSP_LCD_DisplayStringAt(0, 200, (uint8_t *) buf, LEFT_MODE);

    //sprintf(buf, "L_In [0]:%17.1f, [1]:%17.1f", *L_channel_in, *(L_channel_in+1));
    //BSP_LCD_DisplayStringAt(0, 210, (uint8_t *) buf, LEFT_MODE);

  switch(Lab_Execution_Type)
  {
  case 0: // Passthrough case
    arm_copy_f32(L_channel_in, L_channel_out, AUDIO_BLOCK_SAMPLES);
    arm_copy_f32(R_channel_in, R_channel_out, AUDIO_BLOCK_SAMPLES);
    break;

  case 1: // FIR case
    /* Calls the FIR filters and processes the audio block */
    arm_fir_f32(&L_chan_FIR, L_channel_in, L_channel_out, AUDIO_BLOCK_SAMPLES);
    arm_fir_f32(&R_chan_FIR, R_channel_in, R_channel_out, AUDIO_BLOCK_SAMPLES);
    break;

  case 2: // FFT Overlap-add

    // Initialize data coming in to first be zeros
    arm_fill_f32(0, &L_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);
    arm_fill_f32(0, &R_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);

    // Bring data in and put it in the real portions of the vector
    for (uint32_t i = 0; i<Signal_Length; i++)
        {
            L_chan_data_for_DFT[i << 1] = *L_channel_in;
            R_chan_data_for_DFT[i << 1] = *R_channel_in;
            L_channel_in++;
            R_channel_in++;
        }

    // Perform FFT on data
    arm_cfft_f32(&FFT_TYPE, &L_chan_data_for_DFT[0], 0, 1); // In-place computation (now FFT'd)
    arm_cfft_f32(&FFT_TYPE, &R_chan_data_for_DFT[0], 0, 1); // In-place computation (now FFT'd)

    // Complex multiply with the FFT of the FIR_response
    arm_cmplx_mult_cmplx_f32(&L_chan_data_for_DFT[0], &FIR_response[0], &L_chan_DFT_mult_FIR_cmplx[0], DFT_CMPLX_DATA_SIZE);
    arm_cmplx_mult_cmplx_f32(&R_chan_data_for_DFT[0], &FIR_response[0], &R_chan_DFT_mult_FIR_cmplx[0], DFT_CMPLX_DATA_SIZE);

    // Perform inverse FFT
    arm_cfft_f32(&FFT_TYPE, &L_chan_DFT_mult_FIR_cmplx[0], 1, 1); // In-place computation (now IFFT'd)
    arm_cfft_f32(&FFT_TYPE, &R_chan_DFT_mult_FIR_cmplx[0], 1, 1); // In-place computation (now IFFT'd)

    // Bring vector back to just real representation
    for (uint32_t i = 0; i<DFT_SIZE; i++)
        {
            L_chan_data_from_IDFT[i] = L_chan_DFT_mult_FIR_cmplx[i << 1];
            R_chan_data_from_IDFT[i] = R_chan_DFT_mult_FIR_cmplx[i << 1];
        }

    // Perform addition part of overlap-add
    arm_add_f32(&L_chan_data_from_IDFT[0], &L_chan_FIR_state[0], &L_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    arm_add_f32(&R_chan_data_from_IDFT[0], &R_chan_FIR_state[0], &R_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);

    // Copy out the filtered data to the audio channels
    arm_copy_f32(&L_chan_FIR_state[0], L_channel_out, AUDIO_BLOCK_SAMPLES);
    arm_copy_f32(&R_chan_FIR_state[0], R_channel_out, AUDIO_BLOCK_SAMPLES);

    // Update the FIR states by sliding the data
    for (uint32_t i=0; i<NUM_TAPS-1; i++)
    {
        L_chan_FIR_state[i] = L_chan_FIR_state[i+AUDIO_BLOCK_SAMPLES];
        R_chan_FIR_state[i] = R_chan_FIR_state[i+AUDIO_BLOCK_SAMPLES];
    }

    // Fill in zeros for the FIR states that will come
    arm_fill_f32(0, &L_chan_FIR_state[NUM_TAPS-1], AUDIO_BLOCK_SAMPLES);
    arm_fill_f32(0, &R_chan_FIR_state[NUM_TAPS-1], AUDIO_BLOCK_SAMPLES);
    break;

  case 3: // FFT Overlap-add with real-imag efficiency
```

```
    // Initialize data coming in to first be zeros
    arm_fill_f32(0, &L_chan_data_for_DFT[0], DFT_CMPLX_DATA_SIZE);

    // Bring data in and put it in the real portions of the vector
    for (uint32_t i = 0; i<Signal_Length; i++)
        {
            L_chan_data_for_DFT[i << 1] = *L_channel_in;
            L_chan_data_for_DFT[(i << 1) + 1] = *R_channel_in;
            L_channel_in++;
            R_channel_in++;
        }

    // Perform FFT on data
    arm_cfft_f32(&FFT_TYPE, &L_chan_data_for_DFT[0], 0, 1); // In-place computation (now FFT'd)
    //arm_cfft_f32(&FFT_TYPE, &R_chan_data_for_DFT[0], 0, 1); // In-place computation (now FFT'd)

    // Complex multiply with the FFT of the FIR_response
    arm_cmplx_mult_cmplx_f32(&L_chan_data_for_DFT[0], &FIR_response[0], &L_chan_DFT_mult_FIR_cmplx[0], DFT_CMPLX_DATA_SIZE);
    //arm_cmplx_mult_cmplx_f32(&R_chan_data_for_DFT[0], &FIR_response[0], &R_chan_DFT_mult_FIR_cmplx[0], DFT_CMPLX_DATA_SIZE);

    // Perform inverse FFT
    arm_cfft_f32(&FFT_TYPE, &L_chan_DFT_mult_FIR_cmplx[0], 1, 1); // In-place computation (now IFFT'd)
    //arm_cfft_f32(&FFT_TYPE, &R_chan_DFT_mult_FIR_cmplx[0], 1, 1); // In-place computation (now IFFT'd)

    // Bring vector back to just real representation for Left and Right
    for (uint32_t i = 0; i<DFT_SIZE; i++)
        {
            L_chan_data_from_IDFT[i] = L_chan_DFT_mult_FIR_cmplx[i << 1];
            R_chan_data_from_IDFT[i] = L_chan_DFT_mult_FIR_cmplx[(i << 1)+1];
        }

    // Perform addition part of overlap-add
    arm_add_f32(&L_chan_data_from_IDFT[0], &L_chan_FIR_state[0], &L_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);
    arm_add_f32(&R_chan_data_from_IDFT[0], &R_chan_FIR_state[0], &R_chan_FIR_state[0], AUDIO_BLOCK_SAMPLES + NUM_TAPS - 1);

    // Copy out the filtered data to the audio channels
    arm_copy_f32(&L_chan_FIR_state[0], L_channel_out, AUDIO_BLOCK_SAMPLES);
    arm_copy_f32(&R_chan_FIR_state[0], R_channel_out, AUDIO_BLOCK_SAMPLES);

    // Update the FIR states by sliding the data
    for (uint32_t i=0; i<NUM_TAPS-1; i++)
    {
            L_chan_FIR_state[i] = L_chan_FIR_state[i+AUDIO_BLOCK_SAMPLES];
        R_chan_FIR_state[i] = R_chan_FIR_state[i+AUDIO_BLOCK_SAMPLES];
    }

    // Fill in zeros for the FIR states that will come
    arm_fill_f32(0, &L_chan_FIR_state[NUM_TAPS-1], AUDIO_BLOCK_SAMPLES);
    arm_fill_f32(0, &R_chan_FIR_state[NUM_TAPS-1], AUDIO_BLOCK_SAMPLES);
    break;

  case 4: // FFT Overlap-save with real-imag efficiency

        static float32_t new_chan_data_overlap[DFT_CMPLX_DATA_SIZE];

    // Bring data in and put it in the real portions of the vector
    for (uint32_t i = 0; i<Signal_Length; i++)
        {
            L_chan_data_for_DFT[(i << 1) + (L_chan_overlap_save_start_index)] = *L_channel_in;
            L_chan_data_for_DFT[(i << 1) + (L_chan_overlap_save_start_index) + 1] = *R_channel_in;
            L_channel_in++;
            R_channel_in++;
        }

    // This prepares the data for the DFT by adding in the saved data. You can't use the same
    // structure over again because the DFT is in-place and does not produce another copy.
    arm_copy_f32(&new_chan_data_overlap[0], &L_chan_data_for_DFT[0], (L_chan_overlap_save_start_index));

    //Update the overlap_save input by sliding the data - preparing for the next round of processing.
    arm_copy_f32(&L_chan_data_for_DFT[2*AUDIO_BLOCK_SAMPLES], &new_chan_data_overlap[0], (L_chan_overlap_save_start_index));

    // Perform FFT on data
    arm_cfft_f32(&FFT_TYPE, &L_chan_data_for_DFT[0], 0, 1); // In-place computation (now FFT'd)

    // Complex multiply with the FFT of the FIR_response
    arm_cmplx_mult_cmplx_f32(&L_chan_data_for_DFT[0], &FIR_response[0], &L_chan_DFT_mult_FIR_cmplx[0], DFT_CMPLX_DATA_SIZE);

    // Perform inverse FFT
    arm_cfft_f32(&FFT_TYPE, &L_chan_DFT_mult_FIR_cmplx[0], 1, 1); // In-place computation (now IFFT'd)

  // Output non time_aliased samples
  for (uint32_t i = 0; i<Signal_Length; i++)
      {
          *L_channel_out = L_chan_DFT_mult_FIR_cmplx[(i << 1) +
              (L_chan_overlap_save_start_index)];
          *R_channel_out = L_chan_DFT_mult_FIR_cmplx[(i << 1) +
              (L_chan_overlap_save_start_index) + 1];
          L_channel_out++;
          R_channel_out++;
      }

  break;

}
  /* Change font back */
  BSP_LCD_SetFont(&Font16);
}
```

Bibliography

[1] E. Smith, J. Lemke, M. Taylor, H. L. Kirchner, and H. Hoffman, “Frequency of voice problems among teachers and other occupations,” J. Voice **12,** 480–488 (1998).

[2] L. M. da Rocha, S. de Lima Bach, P. L. do Amaral, M. Behlau, and L. D. de Mattos Souza, “Risk factors for the incidence of perceived voice disorders in elementary and middle school teachers,” J. Voice **31,** 258–e7 (2017).

[3] M. Sliwinska-Kowalska, E. Niebudek-Bogusz, M. Fiszer, T. Los-Spychalska, P. Kotylo, B. Sznurowska-Przygocka, and M. Modrzewska, “The prevalence and risk factors for occupational voice disorders in teachers,” Folia Phoniatrica et Logopaedica **58,** 85–101 (2006).

[4] C. P. Schmidt, M. L. Andrews, and J. W. McCutcheon, “An acoustical and perceptual analysis of the vocal behavior of classroom teachers,” J. Voice **12,** 434–443 (1998).

[5] L. C. Cantor-Cutiva, R. E. Banks, and E. J. Hunter, “The effect of upper airway ailments on teachers’ experience of vocal fatigue,” J. Voice **36,** 226–231 (2022).

[6] M. Angelillo *et al.*, “Prevalence of occupational voice disorders in teachers,” J. Prev. Med. Hyg. **50,** 26–32 (2009).

[7] E. J. Hunter and I. R. Titze, “Quantifying vocal fatigue recovery: dynamic vocal recovery trajectories after a vocal loading exercise,” Annals of Otology, Rhinology, & Laryngology **118,** 449–460 (2009).

[8] E. J. Hunter and R. E. Banks, “Gender differences in the reporting of vocal fatigue in teachers as quantified by the vocal fatigue index,” Annals of Otology, Rhinology, & Laryngology **126,** 813–818 (2017).

[9] J. Kristiansen, S. P. Lund, R. Persson, H. Shibuya, P. M. Nielsen, and M. Scholz, “A study of classroom acoustics and school teachers’ noise exposure, voice load and speaking time during teaching, and the effects on vocal and mental fatigue development,” Int. Arch. Occup. Environ. Health **87,** 851–860 (2014).

[10] L. C. C. Cutiva, I. Vogel, and A. Burdorf, “Voice disorders in teachers and their associations with work-related factors: a systematic review,” J. Commun. Disord. **46,** 143–155 (2013).

[11] M. Kleiner, B.-I. Dalenbäck, and P. Svensson, “Auralization-an overview,” J. Audio Eng. Soc. **41,** 861–875 (1993).

[12] P. Bottalico, S. Graetzer, and E. J. Hunter, “Effects of speech style, room acoustics, and vocal fatigue on vocal effort,” J. Acoust. Soc. Am. **139,** 2870–2879 (2016).

[13] R. M. Bermúdez de Alvear, F. J. Barón, and A. G. Martínez-Arquero, “School teachers’ vocal use, risk factors, and voice disorder prevalence: guidelines to detect teachers with current voice problems,” Folia Phoniatrica et Logopaedica **63,** 209–215 (2011).

[14] D. Isetti, L. Xuereb, and T. L. Eadie, “Inferring speaker attributes in adductor spasmodic dysphonia: Ratings from unfamiliar listeners,” American Journal of Speech-Language Pathology **23,** 134–145 (2014).

[15] P. H. Dejonckere, M. Remacle, E. Fresnel-Elbaz, V. Woisard, L. Crevier-Buchman, and B. Millet, “Differentiated perceptual evaluation of pathological voice quality: reliability and correlations with acoustic measurements.,” Revue de Laryngologie-Otologie-Rhinologie **117,** 219–224 (1996).

[16] J. Sundberg and J. Gauffin, “Waveform and spectrum of the glottal voice source,” Frontiers of Speech Communication Research **19,** 35–50 (1978).

[17] G. Le, C. Shih, and Y. Tang, “Distortion in Tone Production due to the Lombard Effect,” In *Proceedings of 1st International Conference on Tone and Intonation (TAI)*, pp. 76–80 (2021).

[18] W. V. Summers, D. B. Pisoni, R. H. Bernacki, R. I. Pedlow, and M. A. Stokes, “Effects of noise on speech production: Acoustic and perceptual analyses,” J. Acoust. Soc. Am. **84,** 917–928 (1988).

[19] E. J. Hunter, M. L. Berardi, and M. van Mersbergen, “Relationship between tasked vocal effort levels and measures of vocal intensity,” Journal of Speech, Language, and Hearing Research **64,** 1829–1840 (2021).

[20] E. J. Hunter, M. L. Berardi, and S. Whitling, “A semiautomated protocol towards quantifying vocal effort in relation to vocal performance during a vocal loading task,” J. Voice **38,** 876–888 (2022).

[21] M. Yadav, D. Cabrera, and W. L. Martens, “A system for simulating room acoustical environments for one’s own voice,” Applied Acoustics **73,** 409–414 (2012).

[22] D. Cabrera, H. Sato, W. L. Martens, and D. Lee, “Binaural measurement and simulation of the room acoustical response from a person’s mouth to their ears,” Acoustics Australia 37 (2009).

[23] M. Yadav, J. Kim, D. Cabrera, and R. de Dear, “Auditory distraction in open-plan office environments: The effect of multi-talker acoustics,” Applied Acoustics **126,** 68–80 (2017).

[24] M. Yadav and D. Cabrera, “Two simultaneous talkers distract more than one in simulated multi-talker environments, regardless of overall sound levels typical of open-plan offices,” Applied Acoustics **148,** 46–54 (2019).

[25] M. Yadav and D. Cabrera, “Autophonic loudness of singers in simulated room acoustic environments,” J. Voice **31,** 388–e13 (2017).

[26] D. Cabrera, M. Yadav, L. Miranda, R. Collins, and W. L. Martens, “The sound of one’s own voice in auditoria and other rooms,” In *International Symposium on Room Acoustics*, (2013).

[27] M. Yadav *et al.*, “AUDITORY ROOM SIZE PERCEIVED FROM A ROOM ACOUSTIC SIMULATION WITH AUTOPHONIC STIMULI.,” Acoustics Australia 39 (2011).

[28] J. K. Whiting, "Development of a Real-Time Auralization System for Assessment of Vocal Effort in Virtual-Acoustic Environments" (2018). Theses and Dissertations. 7056. https://scholarsarchive.byu.edu/etd/7056.

[29] M. Rollins, "The influence of room acoustics on the voice," Bachelor of Science Capstone Project Report, Brigham Young University, Provo, UT (2016). https://physics.byu.edu/docs/thesis/753.

[30] B. Wu and B. E. Anderson, “Building a real-time convolution system for assessing vocal health of teachers,” J. Acoust. Soc. Am. **155,** A210–A210 (2024).

[31] R. Agarwal and J. Cooley, “New algorithms for digital convolution,” IEEE Transactions on Acoustics, Speech, and Signal Processing **25,** 392–410 (1977).

[32] A. V. Oppenheim, *Discrete-time signal processing* (Pearson Education India, 1999).

[33] M. Orlikowski, “Single Producer–Multiple Consumers Ring Buffer Data Distribution System with Memory Management,” In *Trends and Innovations in Information Systems and Technologies*, **1160,** 3–13 (2020).

[34] M. Malka, N. Amit, M. Ben-Yehuda, and D. Tsafrir, “rIOMMU: Efficient IOMMU for I/O devices that employ ring buffers,” ACM SIGPLAN Notices **50,** 355–368 (2015).

[35] J. H. McClellen, A. V. Oppenheim, and R. W. Schafer, *Computer-Based exercises for signal processing using MATLAB 5* (Prentice Hall PTR, 1997).

[36] Y. Chen and V. Dinavahi, “FPGA-based real-time EMTP,” IEEE Transactions on Power Delivery **24,** 892–902 (2008).

[37] I. S. Uzun, A. Amira, and A. Bouridane, “FPGA implementations of fast Fourier transforms for real-time signal and image processing,” IEE Proceedings-Vision, Image and Signal Processing **152,** 283–296 (2005).

[38] M. Psarakis, A. Dounis, A. Almabrok, S. Stavrinidis, and G. Gkekas, “An FPGA-based accelerated optimization algorithm for real-time applications,” Journal of Signal Processing Systems **92,** 1155–1176 (2020).

[39] A. D. Kingsley, J. M. Clift, B. E. Anderson, J. E. Ellsworth, T. J. Ulrich, and P.-Y. L. Bas, “Development of software for performing acoustic time reversal with multiple inputs and outputs,” In *Proceedings of Meetings on Acoustics*, 46 (2022).

[40] S. R. Bistafa and J. S. Bradley, “Reverberation time and maximum background-noise level for classrooms from a comparative study of speech intelligibility metrics,” J. Acoust. Soc. Am. **107,** 861–875 (2000).

[41] B. Seep, R. Glosemeyer, E. Hulce, M. Linn, and P. Aytar, “Classroom Acoustics: A Resource for Creating Learning Environments with Desirable Listening Conditions. ASA, Technical

Committee on Architectural Acoustics and University of Kansas Architectural Engineering Program,", 2003.

[42] F. Wefers, *Partitioned convolution algorithms for real-time auralization* (Logos Verlag Berlin GmbH, 2015), Vol. 20, pp. 68–71.

[43] W. G. Gardner, "Efficient convolution without input-output delay," J. Audio Eng. Soc. **43,** 127–136 (1995).

[44] W. Ahnert, S. Feistel, and W. Richert, "Merging Room-Acoustic and Electro-Acoustic Measurement Methods," In *Audio Engineering Society Convention 116*, (2004).

[45] N. Roy, R. M. Merrill, S. Thibeault, S. D. Gray, and E. M. Smith, "Voice disorders in teachers and the general population," (2004).

[46] GRAS. KEMAR Manikin Type 45BA: Product Data and Specifications. https://www.grasacoustics.com/files/m/a/man_45BB_45BC.pdf (Accessed April 2023).

[47] L. L. Beranek and H. P. Sleeper Jr, "The design and construction of anechoic sound chambers," J. Acoust. Soc. Am. **18,** 140–150 (1946).

[48] G. Fairbanks, *Voice and Articulation Drillbook, 2nd Edition* (New York: Harper Row, 1960), p. 127.

[49] E. H. Rothauser, "IEEE recommended practice for speech quality measurements," IEEE Transactions on Audio and Electroacoustics **17,** 225–246 (1969).

[50] P. Bottalico, S. Graetzer, and E. J. Hunter, "Vocal effort and the effect of room acoustics in noisy environments," In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, **250,** 3578–3588 (2015).

[51] E. Vilkman, “Occupational safety and health aspects of voice and speech professions,” Folia Phoniatrica et Logopaedica **56,** 220–253 (2004).

[52] P. Bottalico, I. I. Passione, S. Graetzer, and E. J. Hunter, “Evaluation of the starting point of the Lombard effect,” Acta Acustica United With Acustica **103,** 169–172 (2017).

[53] N. B. Mahato, D. Regmi, M. Bista, and P. Sherpa, “Acoustic analysis of voice in school teachers,” Journal of the Nepal Medical Association (JNMA) **56,** 658 (2018).

[54] J. C. Stemple, J. Stanley, and L. Lee, “Objective measures of voice production in normal subjects following prolonged voice use,” J. Voice **9,** 127–133 (1995).

[55] N. Hodoshima, *Reverberation-induced speech improves intelligibility in reverberation: Effects of taker gender and speaking rate* (Universitätsbibliothek der RWTH Aachen, 2019).

[56] Z. Zhang, “Mechanics of human voice production and control,” J. Acoust. Soc. Am. **140,** 2614–2635 (2016).