Bachelorarbeit

im Bachelorstudiengang

Informationsmanagement im Gesundheitswesen

an der Hochschule für angewandte Wissenschaften Neu-Ulm

Comparing Different Pitch Detection Algorithms

Erstkorrektor:	Prof. Dr. Johannes Schobel
Zweitkorrektor:	Prof. Dr. Peter Kuhn
Betreuer:	*Maximilian Karthan*
Verfasser:	Tasmiah Hasnat (Matrikel-Nr.: 3131777)

Thema erhalten:	25.06.2023
Arbeit abgegeben:	15.11.2023

Acknowledgements

This thesis acknowledgement is a tribute to everyone who made my journey worthwhile. I would like to thank **Prof. Dr. Johannes Schobel** and **Prof. Dr. Peter Kuhn** for trusting me and providing me a challenging as well as interesting topic. The completion of this thesis owes a great deal to their unwavering support. I would also like to thank **Maximilian Karthan** for providing vital information and insight into the project, answering all my technical questions and being an essential part of each phase leading up to the successful development of the project. Special thanks are due to my family, whose unwavering encouragement fortified my resolve throughout the process of completing this dissertation.

Abstract

Effective speech therapy is fundamental in aiding individuals with speech and language challenges to achieve better communication. Precise pitch detection is a cornerstone of this endeavor, influencing intonation and expressive communication. This research undertakes a comparison of pitch detection algorithms, with an eye towards future integration into speech therapy apps for Logopedic practice. While immediate implementation is not the focus, the study sets the stage for enhancing therapeutic interventions, empowering Logopedic practitioners, and improving communications skills individuals facing speech and language challenges.

This Bachelor thesis presents a comparative study of pitch detection algorithms, with a specific focus on the implementation and comparison of the Autocorrelation Function (ACF) and ACF combined with parabolic interpolation algorithms in the Typescript programming language. The primary aim is to determine the superior algorithm in terms of accuracy and performance, providing valuable insights for potential future integration into speech therapy apps within the field of Logopedics.

Keywords: ACF, Pitch, Frequency, Music, Speech, Therapy

Table of Contents

1 Introduction

Addressing communication and quality of life issues for individuals with speech and language impairments, speech therapy is a crucial discipline within orthopedics. Pitch detection is significant not only for music analysis and voice recognition but also for medical diagnostics, disease diagnosis, and treatment associated with speech. Pitch detection algorithms have made significant advancements from the early conceptual stages. The history of pitch detection can be traced back to the 18th and 19th centuries when scientists and musicians began studying the physics of sound. Scientists like Joseph Fourier made significant contributions to the understanding of signal analysis and the decomposition of complex sounds into simpler sinusoidal components (Cf. Struik, 2023).

Effective communication is a vital skill essential for human interaction. Pitch detection, situated within the realm of audio signal processing, is a multifaceted task. The estimation of fundamental frequency involves determining the perceived pitch of an audio signal, whether it is a musical note, spoken vowel or any other sound. In the field of speech therapy, clear and expressive communication is of paramount importance, particularly for children. The main project is a Speech Therapy App, which is still under construction, and this thesis plays a small part in it. The thesis mainly concentrates in the pitch detection part, using Typescript programming to investigate and compare various pitch detection algorithms. The aim is to find the best match for the Application. The primary objective is to develop a robust tool for future speech therapy applications, specifically targeting at children receiving Logopedic Therapy.

1.1 Background and Motivation

The IQWiG (Institute für Qualität und Wirtschaftlichkeit in Gesundheitswesen) reports that approximately 8% of all children suffer from a specific language development disorder, with boys being twice as prone to it as girls (Cf. IQWiG, 2023). The German Journal of Health Monitoring states that 15.0% of recipients of speech and language treatment are children between the ages of 3-6. Over the past ten years, children and adolescents have used speech therapy far more frequently. Those from low socioeconomic backgrounds are more prone to seeking speech therapy services (Cf. Rommel et al., 2018). Later-life language challenges can significantly impair understanding, which can lower academic attainment and limit professional

opportunities. Youngsters who struggle with language may feel less confident, anxious, teased, and under psychological stress. Later on, endure reading and writing disorders. Treatment for language disorders typically involves speech therapy sessions, consisting of age-appropriate exercises aimed at improving pronunciation, sentence construction and furthermore.

The study of pitch detection has a long history in the field of audio signal processing. In recent years, technology and innovative algorithms have greatly advanced the field of speech analysis and therapy. Pitch, frequently defined as the perceived fundamental frequency of sound, is a valuable indicator for the diagnosis and treatment of speech disorders, making it a crucial part of speech therapy. Recognizing the emotional nuances and intentions behind spoken words becomes possible through the analysis of pitch variations. Consequently, the accuracy of pitch detection directly impacts the effectiveness of these applications. Beyond the world of music and speech, pitch detection finds applications in medical diagnostics, particularly in the field of Logopedics. Voice disorders and speech pathologies often manifest as alterations in pitch patterns. Speech pathologists and Logopedics utilize pitch analysis to diagnose and treat patients with various speech disorders, helping individuals regain their ability to communicate effectively (Cf. Behlau et al., 2021). Although the primary focus may not be the immediate implementation of these discoveries, the consequences that may arise are not noteworthy. The research's findings may influence the creation of new speech therapy technologies in the future.

Traditional pitch detection methods have enabled the development of advanced algorithms that offer improved accuracy and efficiency. This thesis aims to evaluate the efficacy and performance of two pitch detection algorithms, with a specific emphasis on their implementation in Typescript. The study's implications extend beyond academic research to greatly improve speech therapy practices in logopedics. The goal is to provide valuable insights into the strengths and weaknesses of these algorithms to guide the development of future speech therapy applications.

The motivation for this study arises from the paramount importance of accurate pitch detection in these diverse domains. While numerous pitch detection algorithms have been proposed, each has its unique strengths and limitations. The choice of an algorithm depends on the specific application and its requirements (Cf. Gunsteren and Berendsen, 1977). Therefore, it is imperative to undertake a systematic exploration and comparison of these algorithms to understand their suitability for various tasks.

1.2 Organization of the thesis

This thesis is structured to provide a comprehensive examination of pitch detection algorithms, their implementation, and an extensive comparative analysis. The content is organized into the following chapters:

- **Introduction:** This brief opening chapter offers an overview of the research topic, introduces its importance, and outlines the structure of the thesis.
- **Fundamentals**: This section delves into the foundational concepts and theories necessary to understand the research. It lays the groundwork for readers who may not be familiar with the subject matter.
- **Algorithms:** An introduction to various pitch detection algorithms is presented, even if not all of them are implemented. This section helps readers grasp the theoretical background of the chosen algorithms.
- **Implementation:** Details on the technical implementation of the selected algorithms are discussed, including code snippets and implementation strategies.
- **Results:** This section presents the research findings and outcomes, comparing the performance of the implemented algorithms based on the established criteria.
- **Discussion:** The results are analyzed and interpreted in this chapter, addressing the research questions, and providing insights into the significance of the findings.
- **Summary and Outlook:** The final chapter summarizes the key findings and contributions of the research, discusses their implications, and offers suggestions for the future research and potential applications.
- **References:** A comprehensive list of citations and references used throughout the thesis is provided.

Through this structure, the aim is to create valuable resource for researchers, engineers, and practitioners interested in gaining a deep understanding of pitch detection algorithms and making informed decisions about the choice of algorithms for their applications.

2 Fundamentals

This section provides definition for few terms such as: “Pitch”, “Frequency”, “Audio Signal Processing”, to enhance clarity and facilitate comprehension.

2.1 Pitch and Frequency

The definition of Pitch falls into two broad categories: those that make a reference to the association between pitch and the musical scale and those that avoid a reference to music (Plack and Oxenham, 2006). Pitch can be described as the perceptual attributes of a sound, which are denoted as “high” or “low”. It plays a significant role determining its position on the musical scale (Cf.Klapuri, 2006). It signifies the frequency of sound waves whereby a higher frequency generates a higher pitch and vice versa. Pitch holds significant importance in music, as it facilitates the creation of melodies, harmonies, and chords.

Variations in pitch during speech can convey varied meanings through intonation, such as making an inquiry, demonstrating astonishing or showing emphasis for example, a rising pitch at the end of a sentence often indicates a question, while a falling pitch signifies a statement. Oxenham state that, pitch is a fundamental component of auditory perception that is crucial for comprehending speech, enjoy music and selectively attending to specific sounds (Cf. Oxenham, 2023). Specifically in the inner ear's cochlea and the auditory cortex in the brain.

Pitch is affected by how quickly or slowly an object vibrates. To human hearing, sounds can sound differently depending on their pitches, frequencies, wavelengths, and loudness. The approach of the brain to speech and music differs. Music employs the rostromedial prefrontal cortex to understand musical notes and pitch, whereas speech uses the right side of the brain for pitch comprehension(Peretz and Coltheart, 2003). According to Peretz and Coltheart, music utilizes a greater number of brain regions and a more extensive network of brain regions, concentrating on the direction of pitch changes.

According to Byron Emerson Blair frequency is defined as the amount of cycles of a repeating event per unit of time, typically measured in hertz (Hz)(Blair, 1974 ,p.10). When a periodic occurrence completes one cycle within a second, its frequency is equivalent

to 1Hz. The frequency of a sound determines its pitch; as frequency increases, pitch rises, and as frequency decreases, pitch drops. A small bird's chirping has a higher pitch and frequency compared to the lower pitch and frequency of thunder rumbling. Frequency is an integral and precise parameter that can be regulated and gauged with minimal error, thereby making it an important tool both scientific and technological arenas.

High-pitched sounds are produced by rapid vibrations that have high frequencies. Low frequencies are responsible for producing low-pitched sounds, which are produced by sluggish vibrations. The wavelength of a higher pitched sound is shorter, and the wavelength of a lower pitched sound is longer (**Figure 1**).
The source and the medium through which sound wave disperse influence how we hear it. The source itself must vibrate at anywhere between 20 Hz – 20000 kHz and it is the limit of human ear. This range is influenced by individual differences, such as age and hearing ability. And when it does, the back-and-forth motion of the vibration pushes air molecules.

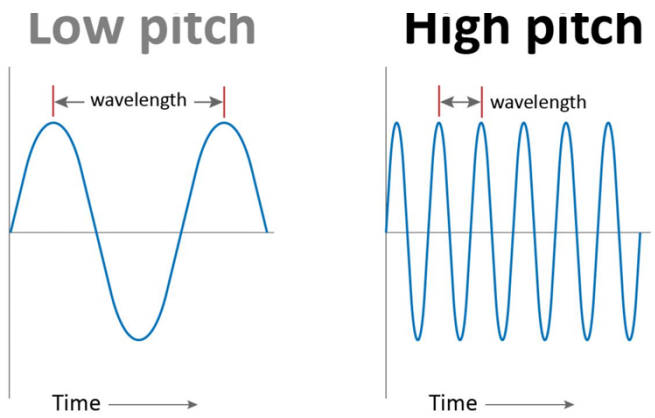

Figure 1: Low- and High pitch adapted from Hearing: Additional Information, source: (Genetic Science Learning Center, 2014)

During the first half of the oscillation- when the source moves forward, molecules are pushed forward and bunch together with molecules in front of them. This causes an increase in the density of air at that point. This region is called compression. It is represented by the crest of a sine wave. During the second half of the oscillation when the source moves back to its original position. It leaves a void, so there are fewer air molecules in that region and thus a decrease in the density of air at that point. This region is called rarefaction. It is represented by the trough of a sine wave. This is one single oscillation.

The quantity of wave peaks in a given amount of time is known as frequency. The pitch increases with the number of peaks in this period. The distance between two wave peaks or trough is known as a wavelength (**Figure 2**).

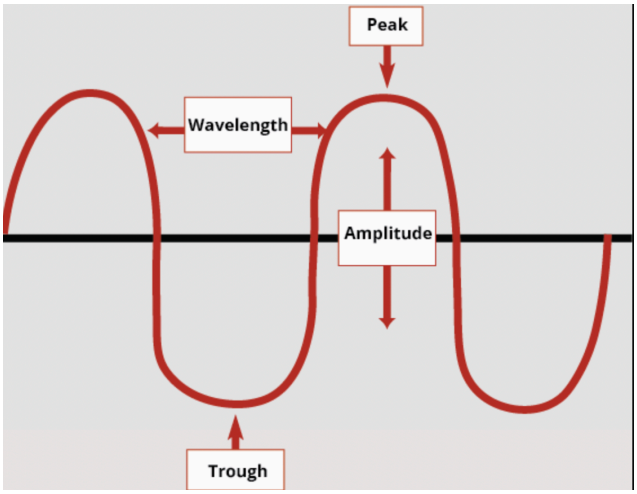

Figure 2: Pitch of Sound adapted from article "What is Pitch of Sound?", source: (Vedantu, 2023)

2.2 Audio Signal Processing

The following section about audio signal processing is based on the book section "Audio Signal Processing" from the book "Speech, Audio, Image and Biomedical Signal Processing using Neural Networks" (Cf. Rao, 2008).

Audio signal processing is a method utilized to interpret and manipulate audio signals, with the main objective of classifying audio. The procedure involves assessing general audio signal characteristics, utilizing time-frequency represents for audio, and examining useful features for classification. Additionally, the process entails removing features from audio signals to obtain relevant information. The objective of audio signal processing is mimic the abilities of the human auditory system in processing complex sound mixtures and creating sophisticated representations of the surroundings. This feature is advantageous in several areas, such as automatic music transcription, multimedia data search and retrieval, and speech recognition in noisy environment.

In the realm of music production, audio signal processing holds significant importance as it enables the creation of various audio effects and facilitates the classification of audio content. Given the expanding digital music repositories, it supports record indexing and

audio classification, improving retrieval and coding efficiency by matching compression methods to audio types.

Speech recognition applications require audio signal processing, especially in noisy settings. In the procedure, sources are separated and identified from the incoming composite audio signal. The technique is referred to as auditory scene analysis. This makes it feasible for the system to precisely transcribe and detect speech even in the presence of background noise by analyzing and isolating speech signals. Signal models that can be generated from perception, cognition, and sound creation should be used to process the audio signal. Audio signal processing often involves specialized software, and programming languages like Python, Java, MATLAB, and C/C++ (Cf. Devaney, 2021).

3 Pitch Detection Algorithms

3.1 Introduction to various Pitch Detection Algorithms

In this section, various pitch detection techniques will be introduced. Pitch detection is a key issue in audio signal processing, with numerous techniques available for reliably measuring the fundamental frequency (pitch) of an audio source. These algorithms display differences in complexity, accuracy, and applicability for different types of signals. Below is an overview of various widely used pitch detection algorithms:

3.1.1 Average Magnitude Difference Function

In this section, the discussion on Average Magnitude Difference Function is primarily based on the work of (Cf. Ross et al., 1974, Muhammad, 2011) and (Cf. Muhammad, 2011).

In speech and audio signal processing, the AMDF (Average Magnitude Difference Function) method is analogous to the autocorrelation method. The AMDF analysis is a sunset of ACF analysis. Rather than associating the input speech signal with distinct time delays, the AMDF approach includes establishing a difference signal between the original and delayed versions. The absolute magnitude of this difference signal is then determined for each delay value. The AMDF technique has a significant advantage, particularly in real-time applications, in that it does not require multiplication operations. As a result, it is a viable option for situations where computational efficiency is critical.

The Average Magnitude Difference Function (AMDF) is defined as follows:

$$D_{(\tau)} = \frac{1}{N-\tau-1} \sum_{n=0}^{N-\tau-1} |x(n) - x(n+\tau)|$$

The input signal is represented by the sequence x(n), which is essentially a series of samples from a speech or audio signal. The formula utilizes a positive integer, τ, which serves as a lag parameter. It denotes the temporal discrepancy or delay existing between the signal and its delayed variation. The signal length is denoted by N, defining x(n) for 0<= n <= N-1. For every value of τ, the formula computes the absolute difference

between x(n) and x (n+ τ) over a particular range of n. The range of n spans from 0 to N-τ-1.

This absolute disparity quantifies the variation of the signal over the time delay τ. The absolute differences are summed up within the defined range of n by applying the summation symbol $\sum$. The AMDF value for a specific lag τ is obtained by dividing the summation result by N-τ-1. This division works as a normalization step and guarantees that the AMDF value remains unaffected by the signal length, making it more reliable and facilitating comparisons between signals of varying lengths.

Table 1 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the AMDF.

Field and Applications	**Advantages**	**Disadvantages**
AMDF is employed in speech recognition to estimate the fundamental frequency (pitch) of the speech signal, which is a vital feature for speech recognition systems. By precisely detecting the pitch, AMDF enhances the performance of speech recognition algorithms. In the context of music analysis, the use of AMDF is prevalent for purposes such as melody extraction, pitch tracking, and note transcription. The technique facilitates the identification of the basic frequency of musical	Simple measurement: The AMDF offers a precise estimation of the pitch contour, making it a straightforward measurement. No multiply operations: Unlike other pitch extraction methods such as ACF (Autocorrelation Function) or CCF (Cross-Correlation Function), the AMDF does not necessitate multiply operations, thus significantly enhancing its computational efficiency. Suitable for implementation on different platforms:	A significant drawback of the AMDF method is its tendency to decrease in the latter half of a frame, which may result in incorrect pitch detection even in low noise conditions. Furthermore, the AMDF technique is susceptible to noise and variations in intensity, potentially yielding false pitch detection.

notes and the tracing of their fluctuations throughout time.	Additionally, it is suitable for implementation on diverse platforms. The nature of AMDF's operations renders it well-suited for development on programmable processors or in specialized hardware. Wide dynamic range: As the AMDF does not have restrictive dynamic range constraints, it is also compatible with 16-bit machines.	

Table 1: Fields, applications, advantages, and disadvantages of AMDF

3.1.2 YIN

In this section, the discussion on YIN is primarily based on the work of (Cf. De Cheveigné and Kawahara, 2002).

The YIN Algorithm was first introduced in 2002 by the scientists Alain de Cheveigné and Hideki Kawahara. It is an algorithm that can estimate the fundamental frequency (F0) of speech and music sounds with a high degree of accuracy. Notably, YIN can explore a broad frequency range without top-end limits. It applies the autocorrelation method, which it modifies to reduce error rates. Moreover, it can work well with high-pitched voices and music. Implementing YIN is undemanding, with low latency, and requires only minimal parameter adjustment. The signal model-based algorithm can be expanded to handle diverse forms of aperiodicity.

The YIN algorithm comprises multiple steps. Firstly, it computes the difference function, which denotes the squared difference between the signal and its version shifted at a lag of t. This is accomplished by summing up the squared differences between corresponding samples of the signal.

$$d(t) = \Sigma[\mathrm{s(n)} - s(n-t)]\text{^}2$$

Subsequently, the cumulative mean normalized difference function is calculated by dividing each value of the difference function by the cumulative sum of the squared difference function. This stage normalizes the difference function, thus facilitating error reduction.

$$d'(t) = d(t)/\Sigma\mathrm{d(k)}$$

Next, an absolute threshold, τ, is computed to establish the existence of a pitch period. This threshold is a fraction (α) of a total number of samples when the cumulative sum of the squared difference is divided.

$$\tau = \alpha * \Sigma\mathrm{d}(k)/\,T$$

The algorithm then locates the first value in the cumulative mean normalized difference function that falls below the threshold. This value determines the estimated period.
To achieve a more precise estimation, the algorithm utilizes parabolic interpolation to refine the estimated period.

Table 2 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the YIN.

Fields and applications	Advantages	Disadvantages
The algorithm's low latency makes it suitable for use in interactive systems. Additionally, it shows promise in managing polyphony, a frequent occurrence in music compositions.	Broad frequency range: The YIN method is beneficial for musical applications in which the fundamental frequency (F0) can exceed high limits. This is because YIN does not necessitate upper limits in F0 search bounds, unlike other techniques.	One of the primary disadvantage of this method is its tendency to fail for F0s above one quarter of the sampling rate. As a result, its suitability for specific applications with extremely high F0s may be limited.

YIN's confidence measure enables efficient down-sampling, making it valuable for saving storage space, like in the MPEG7 standard. Moreover, it is versatile; it can be adapted to manage different forms of aperiodicity in specific applications.	Robustness: YIN is robust because it maintains relatively low error rates over a broad search range. YIN is highly flexible. Flexibility: The algorithm has the capacity to be extended in various ways to manage different types of aperiodicity experienced in specific applications, since it is founded on a periodic signal model. The potential of real-time applications is noteworthy; YIN is particularly suitable for interactive or live systems owing to its low latency. Its straightforwardness is notable; the algorithm demands relatively few parameters which require adjustment, resulting in comparative simplicity.	Additionally, while YIN can be expanded to address polyphony, this function has not undergone extensive testing. Therefore, its efficacy in managing polyphonic signals has not been fully established. Additionally, the algorithm includes a confidence measure for efficient down-sampling, however, this feature relies on accurate estimates to avoid potential errors. Furthermore, although YIN can be expanded to accommodate different forms of aperiodicity, none of these extensions yielded improvements in the error rates observed in the speech databases examined in this study. This implies that the algorithm may be inadequate in scenarios where the signal consistently strays from the periodic model.

Table 2: Fields, applications, advantages, and disadvantages of YIN

3.1.3 Harmonic Product Spectrum

In this section, the discussion on Harmonic Product Spectrum is primarily based on the work of (Cf. Sripriya and Nagarajan, 2013).

Another method for detecting pitch is the Harmonic Product Spectrum (HPS), an audio processing technique that determines the fundamental frequency of an audio signal. This method is particularly useful for signals that contain harmonic components, such as those produced by musical notes or tonal sounds. Technical terms such as HPS will be explained when first used to ensure comprehension.

N. Sripriya and T. Nagarajan from the Department of Information Technology at SSN College of Engineering in Chennai, India, proposed a robust framework to estimate pitch using the HPS, obtained from the discrete Cosine Transform (DCT) of the signal, which is a mathematical technique for signal processing, in particular, data compression and spectral analysis. Herewith DCT is applied to the audio signal to obtain the frequency spectrum and then HPS algorithm is applied for the pitch detection. The motivation to use DCT was that:

> "DCT decorrelates the signal well in the transform domain. Due to this, the peaks corresponding to the fundamental frequency and the higher harmonics are relatively sharper and greater in amplitude."

The formula for Harmonic Product Spectrum consists of:

$$\gamma(\omega) = \prod_{r=1}^{R} |\chi(\omega r)|$$

In the given equation, χ(ωr) represents the signal spectrum at the rth harmonic frequency ωr. The product ∏_(r=1) ^R|χ(ωr) | calculates the magnitudes of the spectrum at each harmonic frequency. The HPS algorithm utilizes either DFT (Discrete Fourier Transform) or DCT to obtain the frequency spectrum of the audio signal, which is subsequently downsized by eliminating every nth sample (n is a positive integer above 1). The smaller spectrum is then examined by the HPS algorithm for more processing. The downsizing is carried out to concentrate on the harmonics of the signal.

The HPS is determined by multiplying the initial spectrum and the downscaled spectra at every harmonic frequency. The product of all harmonics up to a certain maximum value R is calculated. The resulting HPS represents the harmony of harmonics in the signal. The highest value of the HPS across all frequencies is commonly utilized to approximate the fundamental frequency of the signal.
Table 3 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the HPS.

Fields and Applications	**Advantage**	**Disadvantage**
Speech processing: Harmonic product spectrum (HPS) is frequently utilized to estimate pitch in speech processing applications, such as speech coding, speech recognition, and speaker recognition tasks. Music analysis: HPS can be applied to music signals for tasks such as melody extraction, instrument recognition, and pitch tracking. Acoustic analysis: HPS can be employed in acoustic analysis for tasks such as animal vocalization analysis, environmental sound classification, and sound event detection.	Robustness to noise: HPS is recognized for its ability to maintain accuracy in noisy environments, making it appropriate for pitch estimation of speech signals that are severely degraded by noise. Accurate pitch estimation: HPS algorithm grants reliable estimation of the fundamental frequency of a signal, particularly when coupled with relevant preprocessing techniques. Straightforward implementation: HPS's implementation process presents relatively easy steps that contribute to the algorithm being computationally efficient and convenient for real-time applications.	One possible drawback that can be derived is the requirement for parameter tuning. The effectiveness of the suggested approach is susceptible to the choice of parameters, including the HPS order and the employed preprocessing techniques. Another potential downside could be the susceptibility of HPS to harmonics in the signal. Weak or poorly characterized harmonics may diminish the accuracy of pitch estimation.

Table 3: Fields, applications, advantages, and disadvantages of HPS

3.1.4 Spectral Analysis

In this section, the discussion on Spectral Analysis is primarily based on the book section “Spectral Analysis”, from the book “International Encyclopedia of the Social & Behavioral Science” (Cf. Rayner, 2001) and few information’s based on (Cf. Collimator, 2023).

A statistical method for characterizing and examining sequenced data is spectral analysis. Sequential data are observations made in one, two, or three dimensions in space and/or time. Using this technique, observations from the data domain are transformed into the spectral domain by breaking down a sequence into oscillations of various lengths or scales.

The process of Fourier analysis involves the calculation of the Fourier transform using the formula:

$$F(\omega) = \int f(t)\, e\text{^}(-i\omega t)\, dt$$

Which measures the correlation between observations at distinct points in time. The autocovariance function, yy[p], is utilized for measuring the correlation between observations using the formula:

$$\gamma(k) = E[(X\_t - \mu)(X\_(t+k) - \mu)].$$

Nevertheless, in the paper it is observed that spectral estimates lack consistency, implying that they may not converge towards the true value as the number of observations escalates.
Table 4 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the Spectral Analysis.

Fields and Application	Advantages	Disadvantages
Spectral analysis is essential for interpreting EEG and ECG signals, aiding in diagnosing neurological and cardiovascular disorders	The process involves breaking down a sequence into oscillations of varying lengths or scales, converting data observations into the	One significant limitation is that the observations must be equally spaced for the analysis to proceed smoothly.

like epilepsy. It is also applied in medical imaging. IT improves detection and processing in telecommunications, radar, and image processing by enhancing understanding of signal frequency and noise characteristics. Widely used in acoustics, spectral analysis provides data on resonant frequencies and modal properties of structures, enabling the optimization of design parameters for improved performance. By analyzing seismic signals, spectral analysis is pivotal in deducing subsurface details about the Earth, thereby benefiting oil and gas exploration, hazard mitigation, and earthquake prediction programmes.	spectral domain for easier manipulation. The scales provided are crucial statistical descriptors of the data and may indicate significant factors that influence or generate it. Spectral analysis uses Fourier or Harmonic Analysis to manipulate periodic data, employing relevant statistics and a nonperiodic approach to spectral analysis. Higher order spectra, or polyspectra, offer valuable insights for complex distributions requiring higher statistical moments and products.	Since spectral estimates are frequently inconsistent, as the number of observations rises, the estimates may not always converge to the true value. As a result of these numbers' lack of independence, spectral analysis's autocovariance function is difficult to interpret. The traditional method of spectral analysis makes the potentially unreal assumption that the unseen data is zero and the series ends have been smoothed off. Alternative strategies, which are covered in a lot of literature, call for more sophisticated methods like autoregression, moving averages, and maximum entropy techniques.

Table 4: Fields, applications, advantages, and disadvantages of spectral Analysis

3.1.5 Fast Fourier Transform

In this section, the discussion on Fast Fourier Transform is primarily based on the work of (Cooley and Tukey, 1965) and (Bergland, 1969). They explored the applications and algorithms of the Fast Fourier Transform in signal processing.

The FFT algorithm is commonly used for calculating the DFT of a signal or sequence. This algorithm, which was initially introduced by James W Cooley and John W Tukey in their paper "An Algorithm for the Machine Calculation of Complex Fourier Series", has transformed the digital processing of waveforms by reducing the time and cost required for computing a DFT. The basic concept behind the Fast Fourier Transform (FFT) algorithm is to break down the computation of the Discrete Fourier Transform (DFT) of a sequence with a length of N into two smaller DFTs with a length of N/2. This process is then repeated recursively until the sequence length is reduced to 1. The smaller DFT results are then combined to produce the DFT of the original sequence.

The formula for the Fast Fourier Transform (FFT) is as follows:

$$X(k) = \Sigma[0\ to\ N-1]\ x(n) * exp(-2\pi i * k * n / N)$$

Where X(k) stands for the intricate Fourier coefficient at frequency bin k, x(n) represents the input sequence or signal, N denotes the amount of data points, and i signifies the imaginary unit.

Table 5 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the FFT.

Fields and Application	Advantages	Disadvantages
Fast Fourier Transform (FFT) is utilized for the manipulation and transformation of different aspects of a digital signal in digital signal processing. The usage of FFT techniques helps in estimating the power distribution of a signal across its frequency range	Efficiency and Speed: The FFT algorithm can considerably decrease the time required to find a Discrete Fourier Transform (DFT) from numerous minutes to less than one second. Cost-Effective: The use of FFT can also decrease computing a DFT cost from	Understanding the Algorithm: Issues can arise from an insufficient or erroneous comprehension of the FFT algorithm. Value Restrictions: A few FFT programs have constrained values of N that ought to be powers of two.

in power spectrum estimation, which is an important method in signal processing. The application of FFT hardware implementations in the field of audio and electroacoustics is widespread. Fast Fourier Transform (FFT) is applied in real-time digital spectrum analysis, involving the real-time analysis of a spectrum. FFT is utilised in the convolution of series, which is a mathematical procedure applied to two functions to create a third function. FFT is utilised in the analysis of waveforms and spectra in both the time domain and frequency domain, as part of waveform and spectrum analysis.	several dollars to several cents. Versatility is another advantageous feature of this algorithm. The Fast Fourier Transform (FFT) algorithm is employed across various fields, including digital filtering, estimation of power spectra, and real-time analysis of digital spectra.	Contrasts from Continuous Fourier Transform: While the majority of properties seen in the continuous Fourier Transform are preserved, certain distinctions arise due to the restriction that the DFT must operate on sampled waveforms that are defined within finite intervals.

Table 5: Fields, applications, advantages, and disadvantages of FFT

3.1.6 Mel Frequency Cepstral Coefficients

In this section, the discussion on Mel Frequency Cepstral Coefficients is primarily based on the work of (Cf. Zheng et al., 2001).

MFCCs, or Mel-Frequency Cepstral Coefficients, are a prevalent technique for extracting features in speech recognition systems. It is a precise method that holds significant importance in speech recognition technology. The process begins with taking the discrete Fourier transform (DFT) of a speech signal's short-term power spectrum. This is followed by warping it on a mel-frequency scale and transforming it into the cepstral domain using a cosine transform. The MFCCs capture the spectral characteristics of the speech signal and are often included as input features in speech recognition algorithms.

The calculation of Mel-frequency cepstral coefficients (MFCC) undergoes various stages:
Speech signals are divided into short frames, typically lasting 20-40 milliseconds, with a small overlap between adjacent frames. Each frame is multiplied by a window function to reduce spectral leakage. The Discrete Fourier Transform (DFT) is used to obtain the power spectrum, which is filtered through a group of evenly spaced filters organized on the Mel-frequency scale.

The logarithm is applied to convert filter bank energies to the logarithmic scale, which represents loudness according to human perception. The Discrete Cosine Transform (DCT) is applied to logarithmic filter bank energies to obtain cepstral coefficients, which decorrelate filter bank energies and reduce the feature vector's dimensionality. The 0th coefficient, representing the overall energy of the frame, is excluded.

The calculation of MFCC can be summarized using the following formula:

MFCC = DCT (log (Filterbank (Power Spectrum)))

Table 6 presents a comprehensive overview of the advantages and disadvantages, as well as the different areas and usage of the MFCC.

Fields and Applications	Advantages	Disadvantages
Mel Frequency Cepstral Coefficients (MFCCs) are commonly used as input features in automatic speech recognition (ASR) systems, aiding in the modelling and identification of speech sounds. Speaker recognition tasks also utilise MFCCs to extract speaker-specific features for identifying or verifying a speaker's identity. In music information retrieval, MFCCs can also be applied. MFCCs have been utilised in various music information retrieval tasks, including genre classification, music recommendation, and audio similarity analysis.	Mel-frequency cepstral coefficients (MFCCs) are modelled after the human auditory system, making them less susceptible to background noise and other distortions in the speech signal. By utilizing discrete cosine transform (DCT) to reduce the dimensionality of the feature vector, MFCCs provide a concise account of the speech signal's spectral characteristics. MFCCs capture crucial spectral characteristics of speech, including formants and phonetic information, thus proving to be efficient for speech recognition tasks. Additionally, MFCC calculations involve simple mathematical operations such as Fourier transforms and DCT, resulting in computational efficiency.	Lack of temporal information: Mel-frequency cepstral coefficients (MFCCs) are computed based on brief speech frames, usually lasting 20-40 milliseconds. Therefore, MFCCs cannot solely capture temporal information beyond the frame level. Sensitivity to speaker variability: The effectiveness of MFCCs may be adversely influenced by variations in speaker characteristics such as accent or gender, as MFCCs principally capture spectral information.

Table 6: Fields, applications, advantages, and disadvantages of MFCCs

3.2 Autocorrelation Function

In this section, the discussion on Autocorrelation Function is primarily based on the work of (Cf. Kale and Limaye, 2014), (Cf. Skjei, 2011), (Cf. Rabiner, 1977) and (Cf. Indeed Editorial team, 2023).

Autocorrelation involves comparing a signal with itself to determine the similarity between observations, considering the time gap between them. This mathematical technique is used in signal processing to identify patterns that are repeated, such as periodic signals, even if obscured by noise or in the absence of a fundamental frequency in a signal. Pitch and tempo detection are significant applications of autocorrelation because it provides a measure of similarity between a signal and itself at a given lag.

The procedure entails assessing a reference window against a delayed window, with the reference window situated at the global maximum, which occurs at "lag 0". As the signal proceeds, the correlation declines, but in periodic signals, it will rise again and reach a local maximum. The interval between "lag 0" and the initial peak provides an approximation of the pitch/tempo.

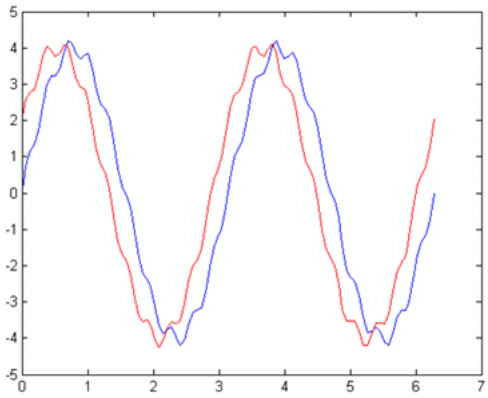

Figure 3: An original signal and a lagged copy of itself adapted from Real-Time Fundamental Frequency Estimation Algorithm for Disconnected Speech, source: (Skjei, 2011)

The similarity can be calculated by adding up the products of the signal, $\mathbf{X_j}$, and its lagged version, $\mathbf{X_{j+\tau}}$:

$$r_t(\tau) = \sum_{j=x}^{t+W-1} X_j \, X_{j+\tau}$$

The autocorrelation function at time t and lag τ is represented by $r_t(\tau)$. This function measures how similar a signal is to a time-delayed version of itself. The formula ∑(j=x) ^(t+W-1) requires adding up the values of variable j, which varies from x to t+W-1. Xj is the value of a discrete-time signal X at time j, while Xj+τ is the time-shifted version of the same signal evaluated at time j+τ. τ represents the lag, signifying the time delay between the original signal and the time-shifted version.

The lag range, where r(τ) is the fundamental frequency period and SR is the sampling rate, represents the frequency search range, which is defined by the maximum and minimum frequencies (fmax and fmin).

In the context of a random process X(t), the autocorrelation function (ACF) is defined as the expected value of the product of two random variables, $X_1 = X(t_1)$ and $X_2 = X(t_2)$, taken at times t_1 and t_2 respectively. The mathematical representation of the ACF is given by:

$$R_{xx}(t_1 t_2) = E\left[X_{(t_1)} X_{(t_2)}\right]$$

Where t2 is a delayed version of t1 by an amount τ . In other words:
$t_2 = t_1 + \tau$

$$R_{xx}(t, t + \tau) = R_{xx}(\tau) = E[X(t)\, X(t + \tau)]$$

Autocorrelation functions find use in several industries, including physics, engineering, meteorology, finance, health, and medicine. In the field of physics, they help scientists to measure and comprehend patterns in sound waves and light and sonic concepts, such as pitch, frequency, and tempo. Meteorologists use them to predict alterations in future weather conditions based on historical data patterns and assess the impact of variables on these trends. Autocorrelation functions are employed in finance to appraise stock values over time and make future predictions.

Additionally, medical imaging software in health and medicine, such as ultrasound imaging, utilizes these functions to generate visual representations of blood flow in patients. In epidemiology, they pinpoint trends in disease outbreaks over time, aiding the comprehension of patterns and the development of strategies to reduce or eradicate their impact on susceptible communities. Overall, autocorrelation functions are a crucial means of comprehending and forecasting future events within numerous industries.

The autocorrelation pitch detector is still one of the most robust and dependable approaches for pitch detection, even though many other approaches have been developed. For a variety of reasons, autocorrelation techniques have been quite successful at identifying pitch. The autocorrelation computation is a reasonably straightforward but time-consuming procedure that is carried out directly on the waveform. The autocorrelation computation is simple to implement in digital hardware, using only an accumulator and multiplier as computational elements, even though it requires high processing speeds. This computation is also essentially phase insensitive. As a result, this method works well for identifying speech that has been corrupted during transmission or that has been transmitted across a phone line.

However, there are several problems associated with this method, such as deciding which autocorrelation peaks correspond to the pitch period and the need for a window for computing the short-term autocorrelation function. The choice of a window and the effect of the window on the autocorrelation function's smooth tapering to 0 as the autocorrelation index increases, further complicate the problem. Formant peaks in the autocorrelation function, which occur at lower indices than the pitch period peak, tend to be of greater magnitude than the pitch period peak.

3.3 Parabolic Interpolation

In this section, the discussion on Parabolic Interpolation is primarily based on the work of (Vandebogert, 2017).

Parabolic interpolation is a mathematical method employed to approximate a curve or function that accurately matches a set of three defined data points. This technique is a type of polynomial interpolation, that customarily fits a parabolic (quadratic) function to the provided data. Its application is particularly practical when dealing with limited data points and when trying to estimate values within them by using a parabolic curve.
The general form of a parabolic function is:

$$f(x) = ax2 + bx + c$$

In equation, f(x) represents the estimated function or curve, and a, b, and c coefficients need to be determined. To conduct parabolic interpolation, a set of three unique data

points is required. Each point must feature a known x and y value, and are labelled as such: (x1, y1), (x2, y2), and (x3, y3). The objective is to determine the coefficients a, b, and c, which can result in a well-fitting parabolic equation for the given data points. This can be achieved by establishing a system of three equations with the data points:

$$y1 = ax21 + bx1 + c$$
$$y2 = ax22 + bx2 + c$$
$$y3 = ax23 + bx3 + c$$

Solving the system of equations will provide the values of a, b, and c, which define the parabolic function that passes through the given data points. Once the coefficients have been determined, the parabolic function f(x)=ax2+bx+c can be utilized to approximate y values for any x falling within the range established by the data points. This constitutes the core principle of interpolation, whereby the approximated parabolic curve is employed to infer values between the data points.

The precision of parabolic interpolation relies on the suitability of a parabolic curve to approximate the data underneath. It performs optimally when the data points demonstrate a parabolic trend. However, its level of accuracy may decrease when the underlying behavior is more complicated.

3.4 Evaluation

In this thesis work, the choice was to utilize the autocorrelation function and autocorrelation function paired with parabolic interpolation. Autocorrelation demonstrates its effectiveness in identifying pitch, particularly in noisy environments. It proficiently detects cyclic patterns in signals that may be distorted or jumbled. The simplicity of ACF makes it well-suited for real-time applications and hardware setups.

Incorporating parabolic interpolation provides a precise method for identifying the highest peak in the autocorrelation function. This precision is vital for accurately estimating the fundamental frequency. The quadratic fit utilized in parabolic interpolation ensures a smoother and more precise representation of the data, especially when dealing with a restricted number of data points. This combination combines the robustness of autocorrelation with advanced precision in peak estimation. The aforementioned

collaboration proves advantageous in situations where precision is paramount, particularly in noisy environments.

Moreover, regarding the mentioned algorithms, the autocorrelation function and autocorrelation function combined with parabolic interpolation offer practical options for the thesis. To balance accuracy and simplicity while considering the restrictions of time and resources, selecting the most appropriate method is crucial. These two pitch detection algorithms strike a practical balance, providing meaningful results within the constraints given.

Only these two pitch detection algorithms have been implemented in the current project due to limitations in time and resources. Nonetheless, the plan is to include the rest of the pitch detection algorithms in the future.

4 Implementation

The goal of this section is to highlight how the implementation phase of the project went. The project focused on only two different pitch detection algorithms among other various types of pitch detection algorithm. (See Chapter 3)

4.1 Autocorrelation Function Algorithm

The subsequent code is intended to execute pitch detection by utilizing autocorrelation. Pitch detection is a crucial job in audio processing, applicable to various fields, including music and speech recognition. Autocorrelation helps identify periodic patterns in the audio signal that can be connected to pitch frequency (Cf. Kale and Limaye, 2014).

```
//const audioCtx = new (window.AudioContext)();
const sampleRate = 44100;
const audioCtx = new AudioContext({sampleRate});
```

Code Snippet 1: Initializing the audio context

This code initializes the **'AudioContext'** with a sample rate of 44100 Hz, a widely used sample rate for audio processing. The sample rate establishes the number of audio samples that are processed per second. Higher sample rates result in superior audio quality but demand more computing resources.

```
//create an empty three sec stereo buffer at the sample rate of the AudioContext

const myArrayBuffer = audioCtx.createBuffer(
    2,
    audioCtx.sampleRate * 3 * 2,
    audioCtx.sampleRate
);
```

Code Snippet 2: Creating an empty Audio Buffer

An empty buffer called **'myArrayBuffer'** is initialized, with a value of **'2'** indicating that it is a stereo buffer, featuring two channels (left and right). The buffer's complete size is determined by the second parameter, **'audioCtx.sampleRate * 3 * 2'**. In order to construct a 3 – second buffer, the sample rate is multiplied by 3, ascertaining the quantity of samples within that period. An added multiplication by 2 is performed to account for

stereo channels, effectively creating 6 seconds of audio data (3 seconds for each individual channel). The buffer's sample rate is specified by the final argument, **'audioCtx.sampleRate'**, which is set to 44100 Hz to align it with the audio context's sample rate. Within this code section, a buffer source node is produced and then linked to the audio destination within an AudioContext.

```
//convert the audio signal to a timedomain waveform. take raw data→convert

const source = audioCtx.createBufferSource();
source.buffer = myArrayBuffer;
source.connect(audioCtx.destination);
source.start();
```

Code Snippet 3: Converting Audio Signal to time domain waveform.

The above given code snippet commences by producing an **'AudioContext'** instance, **'audioCtx'**, which is essential for web applications that involve audio. It serves as the core object that manages audio operations. **'audioCtx.createBufferSource()'** is utilized to generate a buffer source node named **'source'**. It deos not produce sound by itself but rather plays preloaded audio from an AudioBuffer. The AudioBuffer **'myArrayBuffer'** should exclusively contain the raw audio data that requires playing. This audio buffer must be assigned to the source's **'buffer'** property so that it associates directly with the source node.

To commence the playback, one needs to execute the **'start ()'** function on the source node. This function triggers the commencement of the audio signal playback saved in the **'myArrayBuffer'.** The audio signal is then processed, played through the audio context, and directed to the output destination.

In this section of code, the implementation for calculating the autocorrelation of an audio signal is explained step by step:

```
//retrieve waveform data and compute its acf to get results
const waveform = source.buffer.getChannelData(0);
const acf = computeAutocorrelation(waveform);

//function for calculating acf
export function computeAutocorrelation(waveform: Float32Array): Float32Array{
    const acf = new Float32Array(waveform.length);

    //max lag is adjusted to match the length of waveform
    const maxLag = waveform.length;

    for (let lag = 0; lag< maxLag; lag++) {
        let sum = 0;

        for ( let i = 0; i<waveform.length-lag; i++) {
            //cal acf by adding products at diff positions
            sum += waveform[i] * waveform[i+lag];
        }
       acf[lag]= sum;
    }

    return acf;
}
```

Code Snippet 4: Computing Autocorrelation

The **computeAutocorrelation** procedure accepts a single parameter, **waveform**, which should be a **Float32Array**. This array denotes the signal input for the autocorrelation computation. A new **Float32Array** named **acf** is instantiated within the method to store the autocorrelation values. This array denotes the signal input for the autocorrelation computation. This array's size is equivalent to the input **waveform.**

The maximum lag value for calculating autocorrelation is determined by setting the variable **maxLag** to the length of the **waveform**. Since autocorrelation cannot be computed for lags exceeding the signal length, the maximum lag is equal to the signal length. The function then enters a loop, iterating through various lag values, starting from 0 and going up to **maxLag - 1**. Within the loop, the variable **sum** is initialized to zero. This value accumulates the autocorrelation value at the current lag. Autocorrelation at the current lag is calculated using a nested loop that iterates through the waveform array, considering time shifts or lags. At each time shift, corresponding samples at different times are multiplied, and their sum is added to the total. This method calculates the inner product of the signal with a time-shifted version of itself, which is a fundamental step in autocorrelation.

After the inner loop is finished, the calculated autocorrelation value for the current lag is saved in the **acf** array at the corresponding lag index. The outer loop proceeds until all lags are evaluated, and the autocorrelation values are computed for each lag. Ultimately,

the function yields the **acf** array, which comprises the calculated autocorrelation values. These values indicate the degree of similarity between the signal and its time-shifted versions at different lags, offering essential information for various signal processing tasks.

The above code segment demonstrates the calculation of autocorrelation for an audio signal by comparing the signal with itself at various time offsets (lags). The result is an array (**acf**) that holds autocorrelation values, which can reveal periodic patterns or frequencies present in the audio signal.

The Autocorrelation Function (ACF) may be expressed mathematically as follows:

$$ACF[lag] = \sum (x\,[i] * x\,[i + lag])$$

Whereas:

- At a given lag value, the autocorrelation is represented as ACF [lag].
- The signal value at time index i is represented by x[i].
- The signal value at time index i + lag is expressed as x [i +lag], where “lag” is the time shift.
- The total over all appropriate value of i is shown by $\sum$.

The ACF formula adds up the product of corresponding signal values at different time indices to get the autocorrelation over a range of time delays(lags). It measures how similar two signals are to one another at various time delays when they are time-shifted.

In this code section given below, the implementation for finding the peak of the autocorrelation is explained in detail:

```
//function to find peak of acf
export function findPitch (acf: Float32Array, sampleRate: number):number {
    //define min n max period for pitch calculation within range
    const minPeriod = Math.floor(sampleRate/2500); //min period to a max F of 4 kHz
    const maxPeriod = Math.floor(sampleRate/700);   //max period to a min F of 70 Hz
    console.log("SampleRate: ",sampleRate)
    let maxIndex = minPeriod;
    let maxValue = acf[minPeriod];

 //iterate from just after the min period to max period using a loop

    for (let i=minPeriod + 1; i ⩽ maxPeriod; i++) {

  // if the value in autocorrelation exceeds previous max, then update
        if (acf[i]> maxValue) {

            maxValue = acf[i];
            maxIndex = i;
        }
    }

    //position of highest value
    console.log("MaxIndex: ", maxIndex);

    //magnitude of highest value
    console.log("Amplitude: ", acf[maxIndex]);

    //highest f
    console.log("Frequency: ", sampleRate/maxIndex);

   //calculated f
    return sampleRate/maxIndex;
}
```

Code Snippet 5: Finding the peak of ACF.

The function **findPitch** has two parameters, acf (an array of autocorrelation values) and sampleRate (the audio signal's sample rate). The code sets minimum and maximum periods for pitch detection based on the **sampleRate**, determining the frequency range within which pitch will be detected. The minimum period is 4 kHz, while the maximum period is 70 Hz. To track the peak in the autocorrelation function with the highest value, the code initializes two variables: **maxIndex** and **maxValue**, which store the index and the value of this peak. These variables are updated during the code's search for the peak.

The code proceeds to enter a loop, iterating through the autocorrelation values within the range of **minPeriod** and **maxPeriod**, ultimately attempting to locate the index with the highest autocorrelation value, denoting the detected pitch. The loop compares the present autocorrelation value **(acf[i])** with the highest value discovered thus far (maxValue). If the current value exceeds the maximum, it updates maxValue and maxIndex to the new values. The code records data regarding the detected pitch, such

as the sampling rate, the index of the highest peak (**maxIndex**), the amplitude of the peak (**acf[maxIndex]),** and the calculated frequency (**sampleRate / maxIndex**). These recordings can be beneficial for debugging and analysis. The function identifies the peak in autocorrelation values and calculates the corresponding frequency, which is returned as the detected pitch. Overall, the function provides valuable data without subjective evaluations or complex descriptions.

The **findPitch** function analyses autocorrelation values to detect the pitch of an audio signal within a specific frequency range. The function identifies the peak in autocorrelation values and calculates the corresponding frequency, which is returned as the detected pitch. This information is useful for tasks such as pitch detection in audio processing applications.

4.2 Autocorrelation Function Algorithm combined with Parabolic Interpolation

The following code is designed to perform pitch detection using autocorrelation combined with Parabolic interpolation.

```
// function to calculate autocorrelation of a signal
export function autocorrelation(waveform: Float32Array): Float32Array {
const acf: Float32Array = new Float32Array(waveform.length);
const N = waveform.length;

for (let lag = 0; lag <N; lag++) {
  let sum = 0;

  //calculate ACF by adding products at different positions
  for (let i= 0; i < N-lag; i++) {
    sum += waveform[i] * waveform[i+lag];
  }
  acf[lag] =sum; //store the ACF value for present lag
}
return acf;
}
```

Code Snippet 6: Calculate Autocorrelation

The autocorrelation function is utilized to determine the autocorrelation of an input signal by operating on a waveform, which is a Float32Array containing the audio or signal data.

A new Float32Array, named acf, is generated to store the calculated autocorrelation values. It is initially set to zero and has the same length as the input waveform. The acf array is employed to contain the autocorrelation values for different time lags. The length of the input wave is acquired and recorded in the N variable, representing the number of samples present within the signal.

A loop is executed by the code that iterates through various time delays. The lag variable signifies the time lag and ranges from 0 to N. The code computes autocorrelation by summing the product of signal values present at diverse time instances, while accounting for time delays for each respective delay time. The sum variable is initialized to zero within the inner loop. This variable aggregates the sum of products for the designated time delay.

The nested loop traverses the waveform array, considering the time delay. It determines the product of the signal values at time i and time i + delay, accumulating these products in the sum variable. When the inner loop is finished, the calculated autocorrelation value for the current delay is deposited in the acf array at the corresponding delay index. The algorithm loops until all time lags are evaluated and computes the autocorrelation values for various lags. The resulting acf array includes the autocorrelation values, which illustrate the signal's similarities to time-shifted versions of itself at different lags.

In brief, the autocorrelation function computes the autocorrelation of an input signal by iterating through various time lags and calculating the sum of products for each lag. The obtained autocorrelation values are useful for different signal processing applications, such as pitch detection and periodic pattern analysis.

The findMaxIndexInRange function has been created to locate the maximum value's index within a particular range in an input Float32Array. It can prove to be valuable in multiple signals processing jobs, ranging from identifying peaks to determining dominant frequencies within a defined frequency range. Below is a comprehensive code explanation:

```
//function to find the index of the maxi peak in specific F range
function findMaxIndexInRange (array: Float32Array, minFrequency: number, maxFrequency: number, sampleRate: number): number {
  let maxIndex = -1; //default value if no peak is found within the range
  let maxValue = 0;

//corresponding lag calculation for the min and max frequency
const minLag = Math.floor (sampleRate/maxFrequency);
const maxLag = Math.floor (sampleRate/minFrequency);

//ACF array within the specified frequency range
for (let i = minLag; i < maxLag; i++) {
  if (array[i] > maxValue) {
    maxValue = array [i];
    maxIndex = i;
  }
}
return maxIndex;
}
```

Code Snippet 7: Function to Find Index of Maximum Value in Range

The function findMaxIndexInRange accepts the subsequent parameters:
Array, which is a Float32Array used for searching the maximum value within a specific range, minFrequency, representing the lowest frequency of interest in the designated range, maxFrequency, signifying the highest frequency of interest in the assigned range, and sampleRate, which is used in the computation of lag values corresponding to the stated frequencies. Two variables, maxIndex and maxValue, are initially assigned values. maxIndex is given the default value of -1, indicating that no peak has been detected in the specified range. Technical term abbreviations are explained when first used.

Meanwhile, maxValue is initially set to 0. The program calculates the lag index in the array based on the specific minimum and maximum frequencies, considering the sampleRate. The minimum frequency corresponds to the minimum lag. These values represent the calculated lag values within specified frequency range.

Subsequently, the function initiates a loop, iterating through the array within the range of lags between the minimum and maximum values.

The code comprises a loop that constantly compares the value at a specific index in an array with the current maximum value. The current highest value is saved in the “maxValue” variable. If the value at index “i” is greater than the current “maxValue”, the code updates the “maxValue” variable with the new highest value and records the index “i” as the new “maxIndex”. Throughout each iteration of the loop, this comparison and updating process continually occurs. If the code identifies a higher value, it replaces the previous maximum value with this new one and updates the corresponding “maxIndex” variable.

```
//acf detection with parabolic interpolation
export function acfPitchDetection (
  waveform : Float32Array,
  sampleRate: number
) : number {

//compute acf of a signal
const acf = autocorrelation(waveform);

//define frequency range to expect Hz
const minFrequency = 700;  //min f in Hz
const maxFrequency = 2500; //max f in Hz

//find index of the max peak in the acf, within F range
let maxIndex = findMaxIndexInRange (acf, minFrequency, maxFrequency, sampleRate);

//no peak found, then set a default value of maxIndex
if (maxIndex == -1) {
  maxIndex = -1; //default
}

//parabolic around max peak
const prevPeak = acf [maxIndex - 1];
const currentPeak = acf [maxIndex];
const nextPeak = acf [maxIndex + 1];
const interpolatedIndex = maxIndex + (0.5 * (prevPeak - nextPeak))/ (prevPeak - 2 * currentPeak + nextPeak );

//period calculation from interpolatedindex
const period = interpolatedIndex;

console.log('prevPeak:', prevPeak);
console.log('curretPeak:', currentPeak);
console.log('nextPeak:',nextPeak);
console.log('period:', period);

//f calculation from period in Hz
const frequency = sampleRate/period;

//results → console
console.log('MaxIndex:', maxIndex);
console.log('Amplitude:', currentPeak);
console.log('Frequency:', frequency);

return frequency;

}
```

Code Snippet 8: ACF Pitch Detection with Parabolic Interpolation

After iterating through the specified range, the function returns the "maxIndex," which is the index of the maximum value within the specified frequency range. If no peak is found within the range, "maxIndex" remains -1.

To summarise, the findMaxIndexInRange function is utilised to seek the index of the highest value in a designated frequency range contained within an input array. It determines the lag values for the designated frequencies and proceeds to traverse through the array within the specified range with the objective of locating the maximum value's index. This is beneficial for activities such as signal processing which require frequency peak identification.

The acfPitchDetection function has been specifically developed to estimate the fundamental frequency (pitch) of an audio signal. The code processes audio data represented as a Float32Array, along with the sample rate information. It computes the autocorrelation values of the audio data using an autocorrelation function, which returns the values in array named “acf”. These values are used later in the code to determine the pitch of the audio signal. Additionally, the function outlines a specific frequency range that it expects the fundamental frequency to be confined to. The minimum and maximum frequencies of interest, in Hz, are represented by the minFrequency and maxFrequency values.

The findMaxIndexInRange function is called by the function to determine the index of the maximum peak in the autocorrelation array within the specified frequency range. If no peak is found within this range, a default maxIndex of 1 is assigned. To estimate a more accurate index, parabolic interpolation is performed around the maximum peak. This method of interpolation considers the values of the peak being analysed and those of its neighbouring values. It calculates the interpolated index to define the period of the signal in samples. The distance between the maximum point and any of its neighboring points is represented by the numerator (0.5*(prevPeak−nextPeak)). The denominator, which is (prevPeak−2*currentPeak+nextPeak), is associated with the parabolic curve's curvature. The interpolated index is obtained by multiplying this fraction by the maximum point index (maxIndex).

The Parabolic Interpolation may be expressed mathematically as follows:

$$f(x) = ax2 + bx + c$$

whereas:

- F(x) corresponds to the “interpolatedIndex”.
- X corresponds to the “maxIndex”.
- The coefficients a, b, and c are not explicitly represented in the code, but they are implicitly captured in the values of “prevPeak”, “currentPeak”, and “nextPeak”.

Subsequently, the function calculates the frequency in Hz by dividing the sample rate by the interpolated period. The results, which incorporate the maximum index, amplitude (peak value), and estimated frequency, are recorded for debugging and analysis purposes. Ultimately, the estimated fundamental frequency (pitch) is returned as the outcome of the function.

In brief, the acfPitchDetection algorithm approximates the pitch of an audio signal through autocorrelation computation, finding the highest peak in a designated frequency range, and employing parabolic interpolation to boost accuracy. This technique is frequently implemented in speech recognition and music analysis for pitch detection.

5 Results

This section examines the methodology of collecting data and creating the experimental structure to analyze the frequency response of both implemented algorithms. Additionally, the implemented algorithms are compared with emphasis on their distinct characteristics and performance measures.

5.1 Data Collection and Experimental Setup

The process of gathering data for pitch detection algorithms is crucial in understanding their performance across a spectrum of frequencies. The experiments conducted for this study involved the implementation and evaluation of two pitch detection algorithms: Autocorrelation function and Autocorrelation with parabolic interpolation.

Autocorrelation Function Algorithm:

The idea behind the Autocorrelation function pitch detection algorithm is to identify periodicity in the signal. In order to simulate a variety of vocal pitches, sinusoidal waves spanning from 700 Hz to 2500 Hz were generated for the experimental setup. The synthetic signals were subsequently analyzed using the Autocorrelation Function algorithm to determine their corresponding frequencies.

The **figure 4** shows the outcomes of the autocorrelation function algorithm. The estimated frequency is presented in the “Measured Frequency”, while “Tested Frequency” displays the known frequency used in the synthetic signal. Frequencies of particular significance, namely 750 Hz, 1050 Hz, and 1400 Hz, were intentionally chosen to assess the algorithm’s proficiency across a spectrum of pitch levels.

Autocorrelation with Parabolic Interpolation Algorithm:

The second approach combines Autocorrelation with parabolic interpolation to overcome the limitations of the standalone Autocorrelation function. The goal of this hybrid strategy is to improve pitch recognition accuracy, particularly in situations when the autocorrelation function can have trouble. Synthetic signals were produced with frequencies ranging from 750 Hz to 2500 Hz, same like in the first method. After that, the signals were processed using the technique for autocorrelation with parabolic

interpolation. The results show the measured frequencies for the matching tested frequencies, as shown in the **figure 5.**

The algorithms were developed in Typescript programming language, which is appropriate for signal analysis, and the experiments were carried out in a digital signal processing environment. To maximize each algorithm's performance, parameters like windows strategies, interpolated parameters, and signal duration were carefully selected.

The collected data were analyzed by comparing the measured frequencies with the known frequencies of the synthetic signals. This analysis involved calculating error metrics, such as mean absolute error and root mean square error, to quantify the accuracy and precision of each algorithm across the tested frequencies. In order to contribute to a thorough understanding of the performance characteristics of both pitch detection algorithms, the experimental setting sought to shed light on their advantages and disadvantages.

5.2 Comparison of the Implemented Algorithms

The Autocorrelation function and the Autocorrelation with parabolic interpolation pitch detection algorithms were evaluated based on their respective performance across various tested frequencies. The comparative analysis highlights the strengths and limitations of each approach.
The Autocorrelation function algorithm is particularly effective at frequencies such as 750 Hz and 1050 Hz, as evidenced by the accurate pitch detection of 700 Hz and 816.67 Hz, respectively.

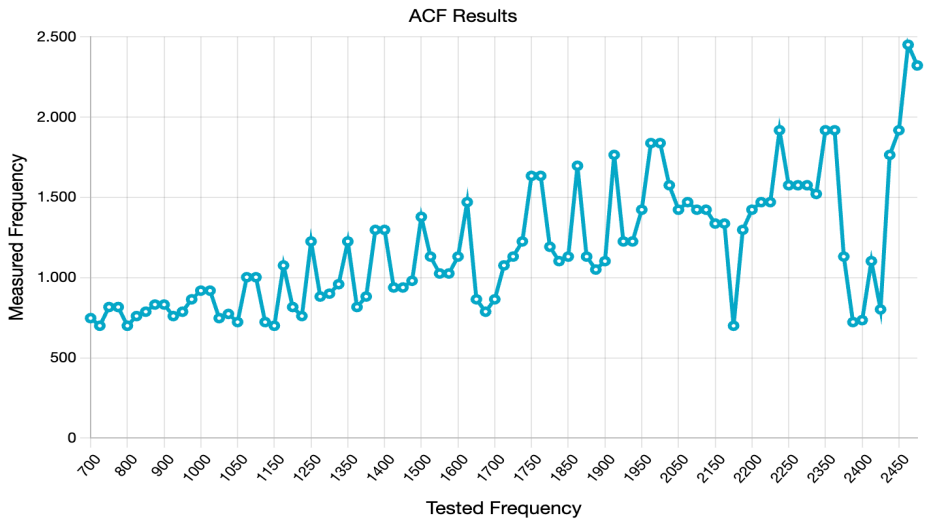

Figure 4: ACF Pitch Detection: Measured vs. Tested Frequency

Limitations in capturing certain pitch characteristics become apparent at 1050 Hz due to variations in measured frequencies such as 747.46 Hz and 773.68 Hz. While the algorithm demonstrates proficiency in some situations, it lacks precision, particularly at higher frequencies. The Mean Absolute Error (MAE) is 486.5281 and Root Mean Square Error (RMSE) is 616.2342. On average, ACF algorithm has an absolute pitch estimation error of approximately 486.5281 Hz. The RMSE suggests a higher dispersion of errors, with larger deviations from the true pitch frequencies.

The aim of the Autocorrelation with parabolic interpolation technique is to overcome the limitations of relying solely on the Autocorrelation function. Notably, frequencies of 1150 Hz and 800 Hz are significant. Results show that when tested at 800 Hz, the frequencies 1349.02 Hz, 1182.79 Hz, and 1247.34 Hz have considerably enhanced accuracy compared to the Autocorrelation function alone. Similarly, the recorded frequencies at 1150 Hz comprise 1164.58 Hz and 1423.26 Hz, indicating an increased ability to detect nuances in pitch.

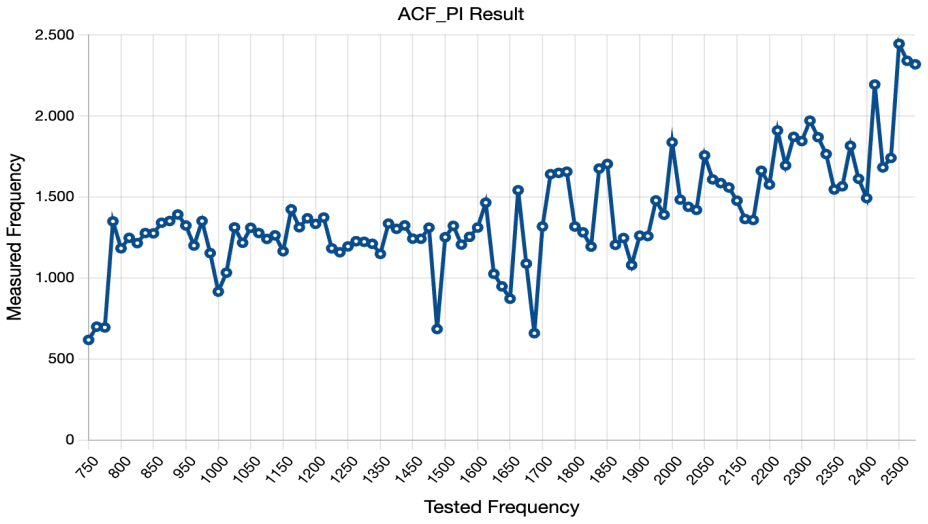

Figure 5: ACF with Parabolic Interpolation Pitch Detection: Measured vs. Tested Frequency

Upon comparison of the two algorithms, it becomes evident that the Autocorrelation function with parabolic interpolation surpasses the standalone Autocorrelation function, particularly in challenging circumstances. The measured frequencies of 617.64 Hz and 699.47 Hz provide evidence that the enhanced algorithm yields more precise measurements at a frequency of 750 Hz. Similarly, the method enhances accuracy at a frequency of 1050 Hz, with recorded frequencies of 1311.85 Hz and 1217.01 Hz. The Mean Absolute Error (MAE) is 375.1296 and Root Mean Square Error (RMSE) is 451.0616. The MAE of 375.1296 indicates a lower average absolute error, suggesting a more accurate pitch estimation. The RMSE of 451.0616 further confirms a reduces overall error dispersion compared to the ACF algorithm alone.

It is crucial to note that both algorithms have advantages and disadvantages dependent on the frequency evaluated. Although the Autocorrelation function operates effectively in certain contexts, it has limitations in others, especially when frequencies are high. Although the combined approach surpasses the individual function, it could still face problems during pitch identification that demands accuracy.

The Autocorrelation with parabolic interpolation technique performs better than the standalone Autocorrelation function, as per the examination of the utilized pitch detection algorithms. Additionally, the accuracy of the improved algorithm is enhanced, particularly in challenging frequency ranges. To select the optimum method for a specific application, one must comprehend the singular advantages and disadvantages of each algorithm.

6 Discussion

The presented result showcases the outcomes of two distinct pitch detection algorithms (see Chapter 5). This section highlights the performance of the algorithms and their possible uses.

Autocorrelation Function Algorithm:

The Autocorrelation function algorithm, as evidenced by the **figure 4**, demonstrates varying degrees of accuracy across different tested frequencies. At lower frequencies, such as 750 Hz, the algorithm performs relatively well, accurately estimating the pitch. However, dissimilarity emerge as the frequency increases, with instances like 1050 Hz displaying larger errors. The algorithm's struggle at higher frequencies might be attributed to challenges in identifying periodicity in shorter waveform cycles.
Additionally, certain frequencies, such as 1150 Hz and 1600 Hz, exhibit significant deviations between the tested and measured frequencies. These dissimilarities could be influenced by harmonics, noise, or limitations inherent to the Autocorrelation function when faced with complex signal structures.

Autocorrelation with Parabolic Interpolation Algorithm:

The second figure (**Figure 5**) introduces results from the Autocorrelation function combined with parabolic interpolation, showcasing a refined approach to pitch detection. The integration of parabolic interpolation aims to address the limitations observed in the standalone Autocorrelation function. Notably, the hybrid algorithm yields more accurate estimations across various frequencies. For instance, at 800 Hz, the measured frequencies are in close proximity to the tested values, indicating improved accuracy. The algorithm also exhibits enhanced performance at 1050 Hz and 1500 Hz, where the Autocorrelation function alone struggled.

Comparative Analysis:

When comparing the two algorithms, it is clear that the Autocorrelation function with parabolic interpolation is superior to its standalone equivalent in reducing errors associated to higher frequencies. The hybrid approach excels in capturing the nuanced characteristics of complex signals, leading to more accurate pitch estimations. It is

essential to acknowledge that both algorithms have their strengths and limitations. The Autocorrelation function, while simpler, might be enough for certain applications and lower frequency ranges. On the other hand, the hybrid algorithm proves advantageous when dealing with diverse and intricate audio signals, making it a valuable tool for applications in music, speech analysis, and sound processing.

The choice between these algorithms depends on the specific requirements of the application. The Autocorrelation function, despite its limitations, remains a viable option in scenarios where simplicity is paramount. Meanwhile, the Autocorrelation function with parabolic interpolation emerges as a robust solution for tasks demanding higher accuracy, especially in complex audio environments. The findings of this study contribute to the broader discourse on pitch detection methodologies, offering valuable insights for researchers and practitioners in the field of digital signal processing.

7 Summary and Outlook

This thesis explores different pitch detection algorithms. The Autocorrelation Function (ACF) and the hybrid Autocorrelation with Parabolic Interpolation algorithm are key focal points, each offering distinct approaches to pitch detection challenges. The Autocorrelation Function is efficient in lower frequencies and can detect periodic patterns in signals, making it suitable for real-time applications. However, it faces challenges at higher frequencies due to the intricacies of pitch detection in complex signal structures.

The hybrid approach, incorporating Parabolic Interpolation, enhances accuracy in challenging and noisy environments, especially in the complexity in pitch detection. The quadratic fit used in parabolic interpolation ensures a smoother and more precise representation of data. The thesis's careful selection of these algorithms strikes a balance between accuracy and simplicity, considering time and resource constraints.

The thesis aims to expand the horizon by including a wide range of pitch detection algorithms in the research, enriching the comparative analysis, and providing a comprehensive toolkit for speech therapists to address the diverse needs of individuals undergoing Logopedic Therapy. Collaboration with speech therapy experts is a promising way to integrate these algorithms into practical tools, potentially advancing the quality of life for those navigating speech and language impairments.

As audio signal processing evolves, there is a scope to explore adaptive algorithms, real-time processing enhancements, and machine learning techniques to refine pitch detection methodologies and tailor them to the dynamic and individualized requirements of speech therapy. This thesis serves as a valuable contribution to understanding pitch detection and as a catalyst for future research.

List of Abbreviations

ACF	Autocorrelation Function
AMDF	Average Magnitude Difference Function
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
F0	Frequency
HPS	Harmonic Product Spectrum
Hz	Hertz
MFCC	Mel-Frequency Cepstral Coefficient
MAE	Mean Absolute Error
RMSE	Root Mean Square Error

List of Figures

List of Tables

List of Code Snippets

References

References

Uncategorized References

BEHLAU, M., MADAZIO, G., VAIANO, T., PACHECO, C. & BADARÓ, F. 2021. Voice evaluation–contribution of the speech-language pathologist voice specialist–SLP-V: part B. Acoustic analysis, physical examination and correlation of all steps with the medical diagnoses. *Hearing, Balance and Communication,* 19**,** 318-324.

BERGLAND, G. D. 1969. A guided tour of the fast Fourier transform. *IEEE spectrum,* 6**,** 41-52.

BLAIR, B. E. 1974. *Time and frequency: theory and fundamentals*, US National Bureau of Standards.

COLLIMATOR. 2023. *What is spectral analysis?* [Online]. Available: https://www.collimator.ai/reference-guides/what-is-spectral-analysis [Accessed 13.10.2023].

COOLEY, J. W. & TUKEY, J. W. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation,* 19**,** 297-301.

DE CHEVEIGNÉ, A. & KAWAHARA, H. 2002. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America,* 111**,** 1917-1930.

DEVANEY, J. 2021. Digital Audio Processing Tools for Music Corpus Studies. *ArXiv,* abs/2111.03895.

GENETIC SCIENCE LEARNING CENTER. 2014. *Hearing: Additional Information* [Online]. Available: https://learn.genetics.utah.edu/content/senses/hearing/ [Accessed 25.10 2023].

GUNSTEREN, W. F. V. & BERENDSEN, H. J. C. 1977. Algorithms for macromolecular dynamics and constraint dynamics
. *Molecular Physics*.

INDEED EDITORIAL TEAM, I. 2023. *Autocorrelation Function: How It Works and How To Use It* [Online]. Indeed. Available: https://www.indeed.com/career-advice/career-development/autocorrelation-function [Accessed 18.10 2023].

IQWIG. 2023. *Umschriebene Entwicklungsstörungen in der Kindheit und Jugend* [Online]. Available: https://www.gesundheitsinformation.de/entwicklungsstoerungen-der-sprache.html#:~:text=Entwicklungsst%C3%B6rungen%20der%20Sprache%20zeigen%20sich,eine%20umschriebene%20Entwicklungsst%C3%B6rung%20der%20Sprache. [Accessed 05.10.2023].

KALE, H. & LIMAYE, D. S. 2014. Autocorrelation of a sound signal. *IOSR Journal of Electrical and Electronics Engineering (IOSE-JEEE)***,** 50-53.

KLAPURI, A. 2006. Introduction to music transcription. *Signal processing methods for music transcription.* Springer.

MUHAMMAD, G. 2011. Extended average magnitude difference function based pitch detection. *The International Arab Journal of Information Technology,* 8**,** 197-203.

OXENHAM, A. J. 2023. Questions and controversies surrounding the perception and neural coding of pitch. *Frontiers in Neuroscience,* 16.

PERETZ, I. & COLTHEART, M. 2003. Modularity of music processing. *Nature neuroscience,* 6**,** 688-691.

PLACK, C. J. & OXENHAM, A. J. 2006. *Overview: The present and Future of Pitch* [Online]. Springer eBooks, NY. Available: https://doi.org/10.1007/0-387-28958-5_1 [Accessed 09.10 2023].

RABINER, L. 1977. On the use of autocorrelation analysis for pitch detection. *IEEE transactions on acoustics, speech, and signal processing,* 25**,** 24-33.

RAO, P. 2008. Audio Signal Processing. *In:* PRASAD, B. & PRASANNA, S. R. M. (eds.) *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks.* Berlin, Heidelberg: Springer Berlin Heidelberg.

RAYNER, J. N. 2001. Spectral Analysis. *In:* SMELSER, N. J. & BALTES, P. B. (eds.) *International Encyclopedia of the Social & Behavioral Sciences.* Oxford: Pergamon.

ROMMEL, A., HINTZPETER, B. & URBANSKI, D. 2018. Inanspruchnahme von Physiotherapie, Logopädie und Ergotherapie bei Kindern und Jugendlichen in Deutschland–Querschnittergebnisse aus KiGGS Welle 2 und Trends. *Journal of Health Monitoring,* 3.

ROSS, M., SHAFFER, H., COHEN, A., FREUDBERG, R. & MANLEY, H. 1974. Average magnitude difference function pitch extractor. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 22**,** 353-362.

SKJEI, T. 2011 *Real-time Fundamental Frequency Estimation Algorithm for Disconnected Speech* Masters of Science, Virginia Commonwealth University.

SRIPRIYA, N. & NAGARAJAN, T. Pitch estimation using harmonic product spectrum derived from DCT. 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013), 2013. IEEE, 1-4.

STRUIK, D. J. 2023. *Joseph Fourier* [Online]. Available: https://www.britannica.com/biography/Joseph-Baron-Fourier [Accessed 01.10.2023].

VANDEBOGERT, K. 2017. *Method of Quadratic Interpolation* [Online]. University of South Carolina. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://people.math.sc.edu/kellerlv/Quadratic_Interpolation.pdf [Accessed 18.10 2023].

VEDANTU. 2023. *What is Pitch of Sound?* [Online]. Available: https://www.vedantu.com/evs/facts-about-pitch-of-sound [Accessed 01.10 2023].

ZHENG, F., ZHANG, G. & SONG, Z. 2001. Comparison of different implementations of MFCC. *Journal of Computer science and Technology,* 16**,** 582-589.

Declaration

I declare that I have written this thesis independently, that I have not submitted it for examination purposes to anyone else, that I have not used any sources or aids other than those stated, that I have marked all direct and indirect quotations as such, and that I have allowed it to be checked using anti-plagiarism software.

Vöhringen, 15.11.2023

Place, Date Signature