ENGN3706 Research Methods

Exploring the Relative Binaural Transfer Function as a Spatial Feature of Sound for Improving Acoustic Emotion Recognition

Punjaya Wickramasinghe
u6310965

Supervisors: Prasanga Samarasinghe & Manish Kumar

Semester 1 2021

Australian National University

Table of Contents

Acknowledgements

Firstly, I'd like to acknowledge my two supervisors, Dr Prasanga Samarasinghe and Mr Manish Kumar for their continuous support and guidance throughout this project. Thank you Prasanga for your patience with me trying to learn all the background knowledge, and your constant words of encouragement. And Manish, thank you for the countless hours you spent sitting and explaining concepts and code to me, as well as for providing me with the annotated dataset and the code for the ReBTF. I know you are both very busy people, so I appreciate the amount of time you put aside to help me with my project. Secondly, I would like to thank Professor Daniel Macdonald, for being so responsive and helpful in ensuring the logistics of the course ran smoothly. Most importantly, to my family and close friends, who supported me in every way to make sure I could spend enough time on the project to produce the result that I wanted. None of this would have been possible without your sacrifice and your understanding, so thank you.

Abstract

Unlike humans, machines lack the ability to perceive and interpret emotions. Acoustic emotion recognition (AER), however, is an emerging research space in the field of machine hearing aimed at addressing this issue. The current literature in AER approaches the problem with regard only to temporal and spectral audio features, with a lacking consideration for how spatial characteristics of sound can influence AER. Thus, in this study, a set of 5 temporal and spectral features - the Zero Crossing Rate, Root-Mean-Square Energy, Spectral Centroid, Spectral Rolloff and the Mel-Cepstral Frequency Coefficients - are combined with a single spatial feature of binaurally reproduced audio - the Relative Binaural Transfer Function (ReBTF) - to assess the impact of the ReBTF on AER performance for 5 basic emotions - Amusement, Fear, Happiness, Calmness and Sadness. Through 4 types of regression analysis - Linear, Ridge, Lasso and Elastic Net regression - it was found that the inclusion of the ReBTF into the feature set improved the performance and statistical accuracy of the AER model. This finding potentially opens up a new space in audio-based AER; however, there were many limitations in the experimental procedure of this study, suggesting that further testing and validation is needed with more sophisticated AER algorithms.

Introduction

Acoustic Emotion Recognition (AER) is a field of machine hearing involved in replicating the human auditory system, to allow machines to predict emotions through audio alone. The type of audio used may vary from music to speech to background noise. By extracting the fundamental features of audio signals, predictive models can be built for audio-based emotion recognition, with the use of various machine learning (ML) algorithms. Typically, AER is approached only through the lens of temporal and spectral audio features. While the accuracy of this approach is steadily increasing, there is potential for further improvement of AER models with the inclusion of spatial audio features. In particular, the Relative Binaural Transfer Function (ReBTF) is one such feature of binaurally reproduced audio, that may help to improve the performance of current AER models.

This study will investigate how the inclusion of a spatial feature, the ReBTF, on a set of commonly extracted temporal (time-related) and spectral (frequency-related) features, will impact the performance of various regression models on performing AER. The remainder of this chapter will provide an exploration of the current state of AER, as well as an explanation of the scope for real-life application. The 'Literature Review' chapter will introduce the relevant theoretical basics of AER. This includes a discussion on the definition of emotions as well as the details about the set of 5 temporal and spectral audio features to be extracted. This chapter will conclude with an introduction to the spatial feature in question, the ReBTF, and the opportunity it poses for improving the performance of AER technology. The 'Experimental Procedure' chapter will then discuss the process of collecting the audio samples, extracting the audio features, and then finally performing the regression analysis to assess the AER performance with and without the inclusion of the ReBTF. The 'Results and Discussion' chapter will present an organised summary of the findings from the investigation and a critical evaluation of the significance of the findings, along with discussing the limiting aspects of the study. Finally, the 'Future Work and Conclusion' chapter, will summarise the project with recommendations made for future avenues of exploration, to further expand and improve on the present study.

1. Motivation for this Study

Machine hearing is a field of artificial intelligence interested in advancing the auditory information retrieval system of machines. Some of the general goals of machine hearing include enabling a machine: to distinguish between speech, music, and background noises; to learn which sounds are noise and which carry noteworthy information; to learn associations between identifiable object names and corresponding sounds, and to infer which direction sound comes from [1]. Since the late 1950s, the majority of advances in this field have been primarily related to applications that increase the logical and practical intelligence of machines [2]. For example, speech recognition, the process of converting human speech into a sequence of words, enhances machines logical intelligence by enabling them to interpret the content of spoken. [3]. In recent years, however, there has been more investigation into progressing the emotional intelligence of machines to enable more natural interactions between man and machine, by teaching machines to detect emotions through sound [3]. More formally, this research space is known as acoustic emotion recognition (AER) and will be the focus of this paper.

1.1 Applications in Acoustic Emotion Recognition

AER has the potential to greatly advance the efficiency of modern interactive technologies. For example, with much more research into this field, there's scope for emotionally intelligent robots to be used for initial psychological consultations. This could be implemented both online and offline, which would greatly increase the accessibility of mental health support worldwide [1]. Furthermore, AER would contribute another dimension to lie detection, ultimately improving its accuracy. In the education sector, given the continuously growing need for online education, information about the emotional state of students through their speech would allow for feedback-driven improvements in teaching quality [4]. Furthermore, the exponential rate of current music production has called for an ever-increasing demand for simple and effective music organization, retrieval, and recommendation systems. Music organization and retrieval by emotion or mood has been considered a reasonable way of accessing music information. In fact, Spotify recently updated its service to enable music organisation by moods [5]. Therefore, incorporating automated emotion recognition systems into such services would greatly benefit both companies and users alike. Finally, while computer vision and machine hearing technologies are individually informative for AER their integration poses the potential for an even more powerful means of human-to-machine communication.

1.2 The Knowledge Gap

Sound can be decomposed into audio features which fit into one of three categories: temporal (time-domain), spectral (time-frequency domain) and spatial audio features. The majority of current research in AER is focused on using only the temporal and spectral audio features for developing a model for predicting emotions. As such. there is a lacking consideration in the literature of the impact of spatial features on AER accuracy [2, 3, 5, 6]. Therefore, this study aims to see if the inclusion of spatial features in a set of temporal and spectral features, would increase the AER performance of a regression model on a set of audio samples. The temporal features were the Zero Crossing Rate and the Root-Mean-Square Energy. The spectral features were the Spectral Centroid, Spectral Rolloff and Mel-Frequency Cepstral Coefficients. This set of 5 temporal and spectral features were chosen due to their recurring referral in numerous past studies [2, 3, 7, 8], which suggests that they are well-tested features. The spatial feature that was included in this set was the Relative Binaural Transfer Function (ReBTF). These features will be explained further in Section 3 and 4.

Literature Review

2. Definition of Emotions

To further discuss how a regression model may be used to perform AER, it is necessary to first define what emotions are, given how ambiguous they are by nature. There isn't a widely agreed-upon definition for what emotions fundamentally are; however, there are two general approaches for conceptualising emotions; the categorical and the dimensional approach. This study will use the categorical approach.

2.1 Categorical Conceptualisation of Emotions

The categorical approach proposes the existence of a handful of unique, universal emotions, from which all other secondary emotion classes may be derived [6]. The universal emotions are thought to be independent of culture and associated with distinct patterns of human psychological changes [9]. Various categorical models propose differing sets of primary emotions, such as Hevner's Eight Clusters or Schubert's Erlkonig [10, 11]; however, the approach used in this study is the 'palette theory'. The palette theory claims that emotions can be decomposed into primary emotions similar to how any colour can be decomposed into its most fundamental, primary colours [12]. According to the palette theory, the primary emotions are Anger, Joy, Fear, Disgust, Sadness and Surprise, which are supposedly the most dominant emotions in our lives. With the above in consideration as well as the scope of this project, a small pool of 5 emotion classes similar to those in the palette theory, was chosen - Amusement, Fear, Happiness, Calmness and Sadness.

3. Temporal and Spectral Audio Features

An audio signal is a digitised representation of sound, which encodes all the fundamental information needed to reproduce the original sound [13]. This information may be broken down into more fundamental features, which extract mathematical properties of the audio sample to describe the signal more quantitatively. Thus, these audio features are descriptors of sound and can be used in unison with machine learning, to formulate mappings between audio samples and emotions [13].

It is important to note that the range of the signal for which these features are extracted can greatly change the obtained results. Analysis can be local or global; a global analysis is concerned with the behaviour of the signal through the entire signal while a local analysis deals with short, windowed frames [2]. For this study, a frame-by-frame analysis is used. Frames are portions of perceivable audio segments. Sound signals are typically sampled at a rate of 22.05kHz which means each sample has a duration of 0.0455ms. The issue is that this duration is much lower than the resolution for which human ears can sample sounds, which is approximately every 10ms [14]. Therefore, we use frames and ensure the length of each frame is long enough for an analysis of the frame to have real-world meaning.

3.1 Time-Domain Features

Time-domain features are extracted from the waveform in the time-domain representation of the signal (Figure 1). We can directly observe the various events that happened in the sound. The time-domain features that will be extracted for this study include the Zero-Crossing Rate and the Root-Mean-Square energy.

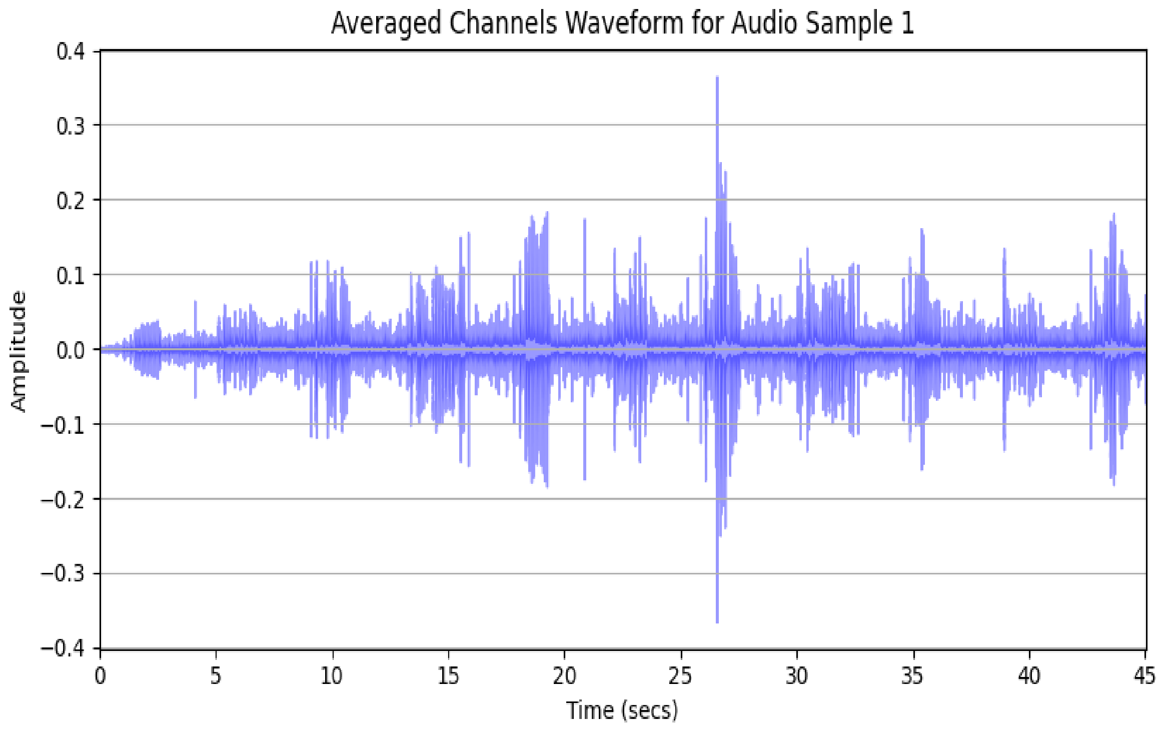


Figure 1 - Average of Left and Right Channels for Audio Sample 1

3.1.1 Zero Crossing Rate

The Zero-Crossing Rate (ZCR) for an audio frame is defined as the rate of change of sign across the mean of the signal (Figure 2).

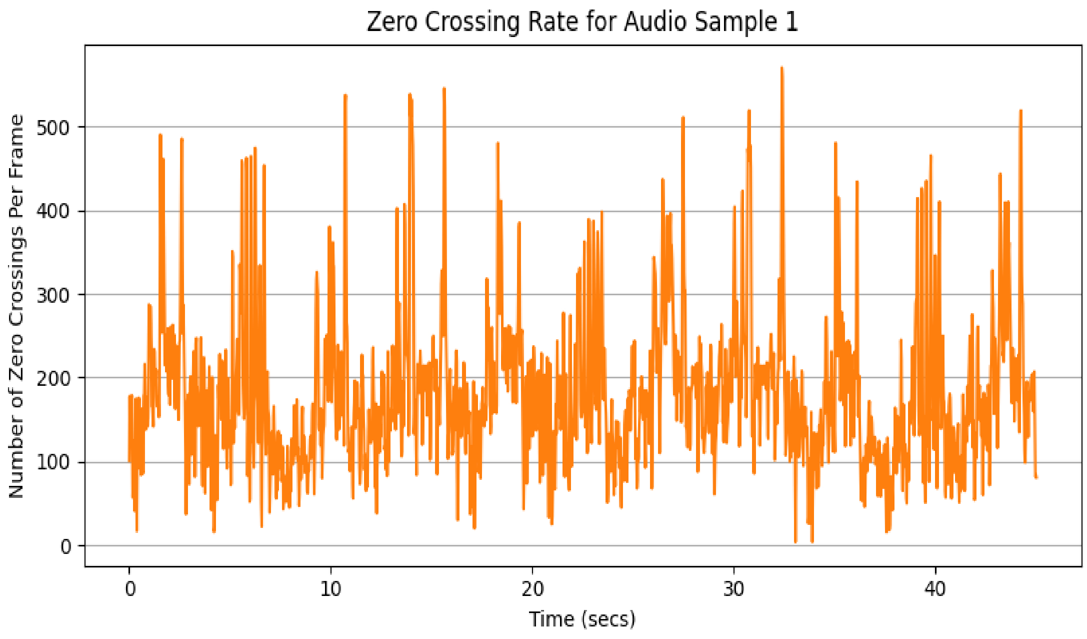


Figure 2 - Zero Crossing Rate for Audio Sample 1

ZCR can be useful in detecting whether a signal is voiced, unvoiced or silent [7]. A high measured value of the ZCR is typically an indication of an unvoiced signal due to the greater presence of background noise, while a low measured value may be an indication of a voiced signal. Music signals typically have a higher ZCR than speech signals; hence, the distinction between music and speech can be partly made with the ZCR [15]. Furthermore, ZCR can be used in monophonic pitch estimation, which is the relationship between ZCR and the pitch of a sound; the higher the ZCR, the higher the pitch [16]. Mathematically, the ZCR for a given frame of a signal is defined as follow, where: k is the sample number with respect to the entire signal, $s(k)$ is the amplitude of the kth sample, $sgn()$ is the sign function, t is the frame number and K is the frame size:

$$ZCR_t = \frac{1}{2} \sum_{k=t \cdot K}^{(t+1)\cdot K-1} |(sgn(s(k) - sgn(s(k+1))| \qquad \text{Eq. 1}$$

3.1.2 Root Mean Square Energy

The Root-Mean-Square (RMS) Energy for an audio frame is the square root of the average of the squares of the magnitude of the signal in the frame (Figure 3).

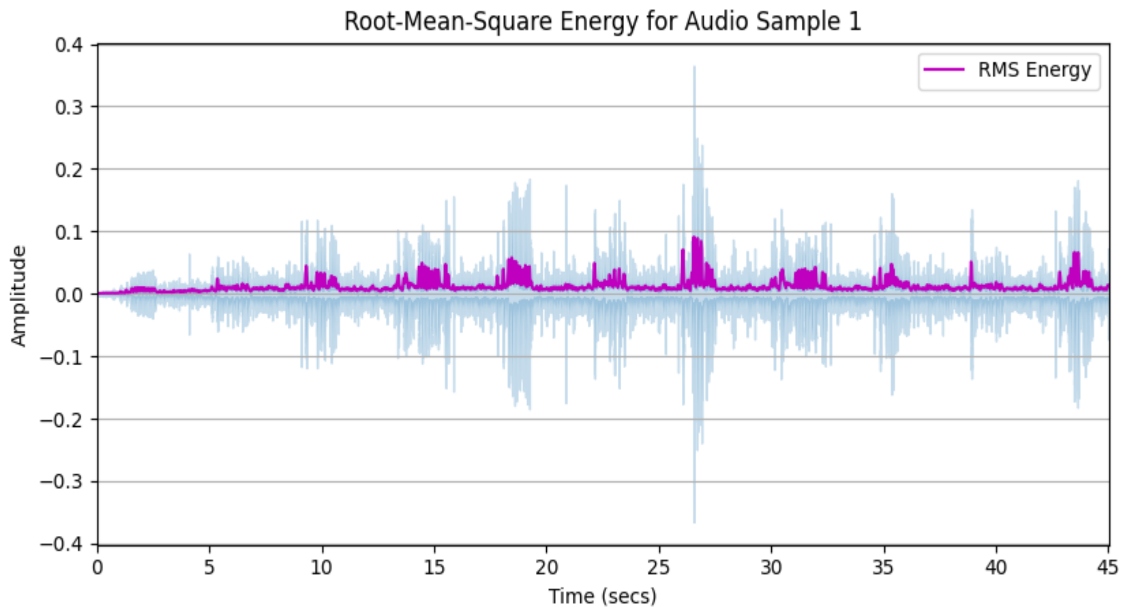


Figure 3 - Room-Mean-Square Energy for Audio Sample 1

Essentially, it provides a representation of the average energy present in a frame. It should be noted that signal energy is best represented in the frequency domain so that the energy of each frequency individually may be analysed, however, the benefit of the RMS Energy is that its calculation is a much faster alternative to converting the signal into the frequency domain [16]. The RMS Energy changes quite drastically on the occurrence of new audio events, and so, the RMS Energy can be leveraged to detect voiced versus unvoiced speech [15]. It also has applications in music genre classification as central genres of music have been found to have generalisable patterns of RMS Energy [16]. Mathematically, the RMS Energy for a given frame of a signal is defined as follows, where the variables are defined in the same way they were for the ZCR above:

$$RMS_t = \sqrt{\frac{1}{K}\sum_{k=t\cdot K}^{(t+1)\cdot K-1} s(k)^2} \qquad \text{Eq. 2}$$

3.2 Moving from Time to Frequency Domain

3.2.1 Fast Fourier Transform

The problem with time-domain features is that an audio signal can be characterised more fundamentally with reference to its frequency components, and temporal features are unable to capture this vital aspect of the signal. The conversion of a time-domain signal into the frequency-domain for a digital signal is done through the Fast Fourier Transform (FFT), which produces what is known as the complex Fourier coefficients. Ultimately, the magnitude of the coefficients assesses the extent of the contribution of each frequency to the overall signal; a higher magnitude is indicative of a greater contribution to the signal [17]. The output of an FFT is called the spectrum (Figure 4).

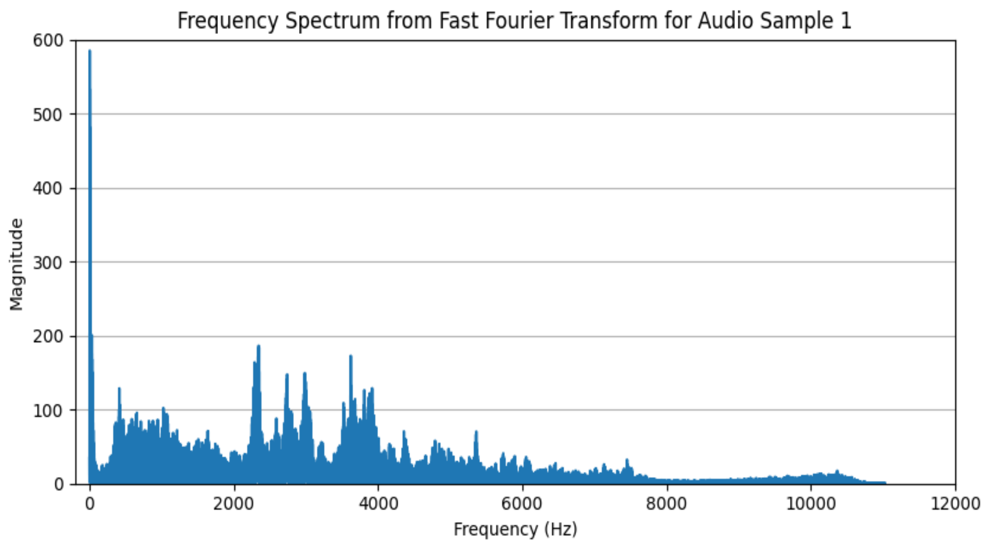


Figure 4 - Frequency Spectrum for Audio Sample 1

While we know what frequencies are present in the signal when looking at a spectrum, we do not know when they are present. This is a problem because audio data is highly dynamic and so, analysing the evolution of frequency components over time provides useful information. To get information about the frequency as a function of time, we introduce a modification to the FFT, which is the Short-Time Fourier Transform (STFT).

3.2.2 Short-Time Fourier Transform

The intuition behind the STFT is that the signal is windowed into smaller segments and FFT is applied to each of these segments. These windows are defined by two parameters; the frame size and the hop length. The frame size is the windowed range of samples that the FFT will be applied to, and the hop length is how far the windows move to the right after each FFT [18]. These windows are often overlapped as a means of mitigating the issue of spectral leakage [19]. The output of the STFT is a spectrogram, which is a signal in the time-frequency domain [20]. Visually, a spectrogram is a 3-dimensional figure, with conventional x and y axes for time and frequency and a colour-based axis for magnitude. That is, the contribution of any frequency is described by the brightness of the pixels at that point; the brighter the colour the more the contribution of that particular frequency band at that specific time. Taking the log of the frequency and magnitude axes, we produce a spectrogram that more accurately represents the human perception of frequency (Figure 5).

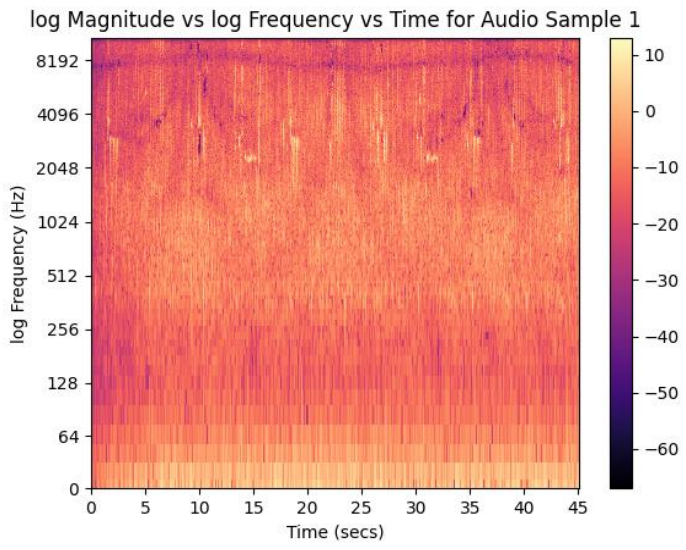


Figure 5 - Spectrogram for Audio Sample 1

3.3 Time-Frequency-Domain Features

Time-frequency-domain features are extracted from the spectrogram of an audio signal. It is recognised that the emotional content of a sound has an impact on the distribution of spectral energy across a range of frequencies. The time-frequency domain features that will be extracted for this study include the Spectral Centroid, Spectral Roll-Off and the Mel Frequency Cepstral Coefficients.

3.3.1 Spectral Centroid

The spectral centroid locates the frequency band where most of the energy is concentrated (Figure 6).

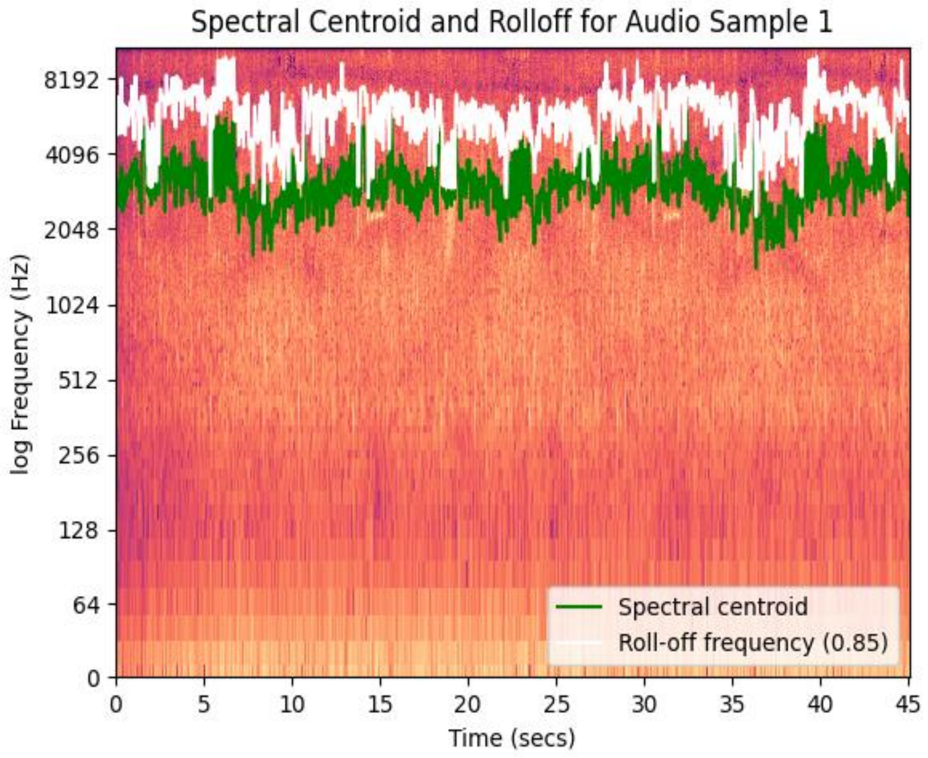


Figure 6 - Spectral Centroid and Rolloff for Audio Sample 1

Perceptually, the spectral centroid maps onto a prominent timbral feature called sound brightness, which has been formalised as an indication of the amount of high-frequency content in a sound [15, 21]. It is also connected to the pitch of a sound and can be used to distinguish between male and female voices. Furthermore, it may be concluded that changes in the spectral centroid may be related to changes in our perception of emotions, since timbre and pitch are both aspects of sound that affect emotions [22]. The spectral centroid (SC_t) for a given frame, t, is calculated as the weighted mean of the frequencies in the frame, where N is the frame size, t is the frame number, n is the frequency bin and $m_t(n)$ is the magnitude for that frequency bin:

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)} \qquad \text{Eq. 3}$$

3.3.2 Spectral Roll-Off

Spectral roll-off is defined as the Nth percentile frequency of frequency distribution. In other words, the roll-off is the frequency below which N% of the signal energy is concentrated and may be regarded as a measure of the skewness of a spectrogram (Figure 6). Typically, N takes a value of 85 or 95. The roll-off can be used for distinguishing between voiced and unvoiced signals. Unvoiced signals have a high proportion of their energy within the high-frequency bands while voiced signals tend to have the majority of their energy within the lower-frequency bands [15]. This means that unvoiced signals will have a higher roll-off while voiced signals will have a lower roll-off. Mathematically, it is found by solving for d such that:

$$\sum_{n=b_1}^{d} m_t(n) = N \sum_{n=b_1}^{b_2} m_t(n) \qquad \text{Eq. 4}$$

Where the variables are defined in the same way as for the Spectral Centroid above, except that b_1 and b_2 are the band edges of the bin, and d is the desired frequency bin, below which N% of energy is contained.

3.3.3 Mel Frequency Cepstral Coefficients

The Mel-scale modifies frequency to more accurately replicate humans' perception of frequency; that is, humans are better able to identify small changes in speech at lower frequencies. For example, two sounds played at 300 Hz vs 400 Hz would be easily distinguishable from each other, whereas another two sounds played at 10,100Hz and 10,200Hz would have almost no perceivable difference despite varying by the same difference of 100Hz. The Mel scale relates the perceived frequency of a tone, $Mel(f)$, to the actual measured frequency, f.

$$Mel(f) = 2595 \log\left(1 + \frac{f}{700}\right) \qquad \text{Eq. 5}$$

From the spectrum of a signal, the signal is transferred into the Mel scale. Then, the log of the magnitude is taken, followed by performing the Inverse Fourier Transform. The resulting spectrum is not in the time or the frequency domain, but rather, in what is called the quefrency domain and the resulting spectrum was known as the cepstrum [23]. The cepstrum contains information about the rate of change of the power from frequency band to frequency band.

The Mel-frequency cepstral coefficients (MFCCs) are the coefficients that represent the cepstrum (Figure 7). Interestingly, the MFCCs have been found to accurately represent the filtering effect of the human vocal tract (teeth, tongue, esophagus etc.) on speech. As such, the MFCCs are predominantly used in speech recognition applications.

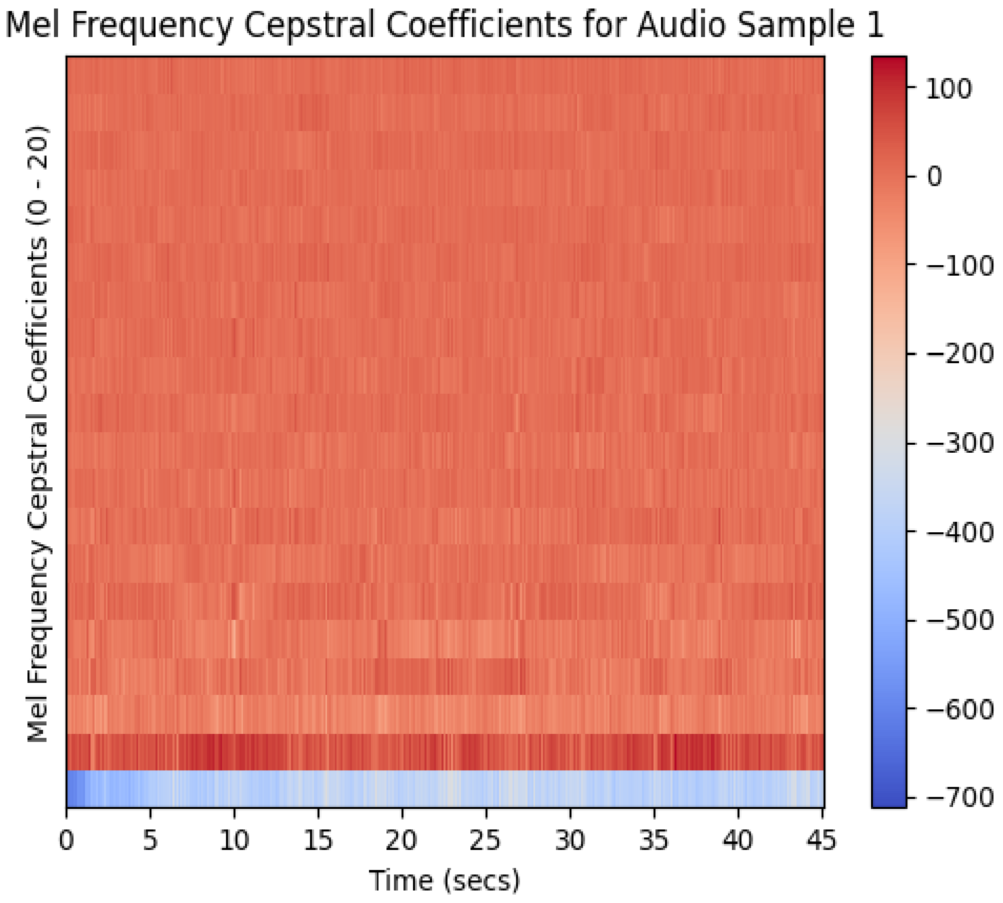


Figure 7 - Mel-Frequency Cepstral Coefficients for Audio Sample 1

4. Spatial Features of Sound

Studies suggest that 3D audio spatialisation can impact how emotions are induced [24, 25]. This section contains a brief discussion on the necessary theory of how spatial information can be contained within an audio signal and how spatial features may be extracted and used for the purpose of this study.

4.1 Sound Localisation

As humans, our ability to localise sound in space, to infer its direction and distance away from us, is known as sound localisation. Sound localisation can be divided into cues that depend on a single ear (monaural) and cues that depend on a pair of ears (binaural).

4.1.1 Monaural cues

Monaural cues rely on the various transformations made by a single ear upon receiving a signal. These monaural cues can generally be represented by the Head-Related Impulse Response (HRIR), which is the spatial function that models the transformation of an audio signal by the human body before entering the eardrums, which depends on the direction from which the sound came. The size and mass of the head, the shape of the ears, the length and diameter of the ear canal, all manipulate the incoming sound by boosting particular frequencies and attenuating others [25, 26]. Monaural cues are a key contributor to our ability to localise sound in the vertical plane. For example, the asymmetrically shaped pinna distorts incoming sound, such that humans can identify the approximate elevation of the incoming sound [25].

4.1.2 Binaural cues

On the contrary, binaural cues are defined as cues related to differences of a sound as perceived at the two ears. The most common binaural cues are the interaural amplitude difference (IAD) and the interaural time difference (ITD). IAD is the difference in the amplitude of the sound received at the two ears while ITD is the difference in time taken for a given signal to reach the two ears due to the distance between the two ears [25]. This typically occurs when the sound source is not perfectly in the front-to-back vertical plane relative to the listener. Binaural cues are predominantly responsible for sound localisation in the horizontal plane. These binaural cues such as the ILD and ITD can be modelled by the Room Impulse Response (RIR).

4.2 Binaurally Reproduced Audio

In relation to AER, studies that neglect the spatial dimension of sound presumably use monaural audio samples - audio that is recorded on a single microphone. Assuming the samples are played back through earphones, they will have no traces of ITD, IAD and the HRIR. Consequently, the audio samples lose all their spatial information, meaning the listener cannot gauge the incoming direction of a given sound. In this experiment, the audio samples used were a collection of binaurally reproduced recordings, which have elements of the ITD, IAD and the HRIR already integrated. That is, the recording microphones were placed at two spatial positions, presumably having an approximate separation similar to that between the two ears (21cm) to introduce the effects of IAD and ITD [27]. This process produces two channels of audio, which are then fed through a simulated HRIR. Since the audio samples were sourced from the internet, information regarding the IAD, ITD and the unique HRIR used is unknown, limiting the range of spatial features that could be extracted. Hence, the only spatial feature being investigated in this study is the ReBTF.

4.3 Relative Binaural Transfer Function

Given that the methodology for rendering the binaural recordings is unknown and only the sound field filtered with the HRIR is available, the only spatial feature that can be extracted is the ReBTF, which is an approximation of the Relative Transfer Function (ReTF). The ReTF is defined as the ratio of the transfer functions of the left and right channels of a binaural recording. A simplified derivation of the ReBTF is as follows:

As discussed, the input signal passes through two systems to produce the output signals received at the two ears: the HRIR and the RIR. Assuming the absence of noise, Figure 8 depicts an approximated general system response.

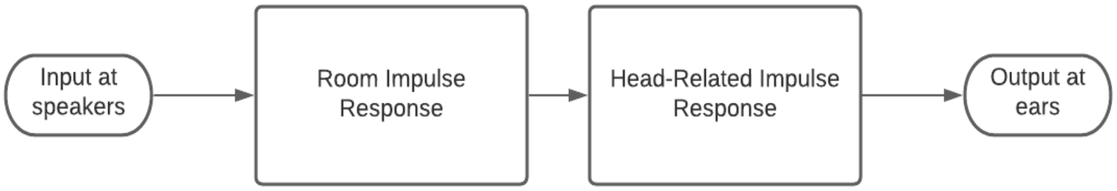


Figure 8 - Simplified approximate system response for binaurally produced audio

Consider the input signal at the speakers as $s(t)$ and the output signals at the left and right ears to be $p_1(t)$ and $p_2(t)$ respectively. Furthermore, consider the RIR to be $r(t)$ and the HRIR to be $h(t)$. Therefore, Figure 9 represents a more mathematical system response.

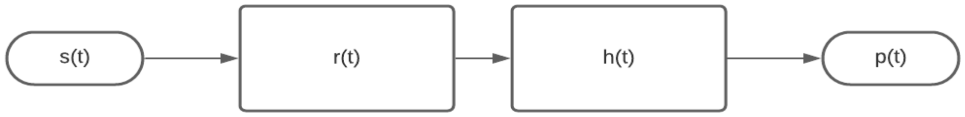


Figure 9 – Approximate system response for binaurally produced audio

As an equation, this system output can be represented as a series of convolutions:

$$p(t) \approx s(t) * r(t) * h(t)$$

By computing the Fourier Transform to enter the frequency domain, this equation becomes:

$$P(t) \approx S(t) \cdot R(t) \cdot H(t)$$

Where P(t) is the power spectral density of the output, $S(t)$ is the power spectral density of the input, $R(t)$ is the Room Transfer Function and $H(t)$ is the Head-Related Transfer Function. Applying this to the left and right output signals, we get:

$$P_1(t) \approx S(t) \cdot R_1(t) \cdot H_1(t)$$
$$P_2(t) \approx S(t) \cdot R_2(t) \cdot H_2(t)$$

By taking their ratio, we see that:

$$\frac{P_1(t)}{P_1(t)} \approx \frac{R_1(t) \cdot H_1(t)}{R_2(t) \cdot H_2(t)}$$

Recalling that the ReTF is the ratio of the transfer functions of two microphones, it is evident that $\frac{P_1(t)}{P_1(t)}$ approximates the ratio of the transfer functions. Note, it is only an approximation due to the assumption made to neglect the effect of noise. As such, this ratio of the output power spectral density functions is what is known as the Relative Binaural Transfer Function. Clearly, this feature contains information about the spatial aspect of the original recording, seeing as the transfer functions are spatial functions themselves.

5. Regression Analysis

Regression analysis is a predictive modelling methodology that investigates the relationships between independent variables (predictors) and dependent variables (targets), such that outputs for unseen inputs can be accurately interpolated or extrapolated [28]. Regression models use algorithms that select coefficients for the weightage of each independent variable such that a loss function is minimised. A common issue with regression analysis, however, is that of overfitting. Overfitting is a statistical modelling error that arises when the predictive model is too closely aligned with the training dataset [28]. Consequently, the model is likely to perform poorly in predicting accurate outputs for unfamiliar data.

Regularisation is used as a means of addressing the issue of overfitting with regression models. In short, regularisation involves the introduction of a shrinkage parameter, alpha (α), which penalises large coefficients of a regression function [28]. Various regression models incorporate regularisation into their analysis, such as Ridge, Lasso and ElasticNet Regression.

Ridge and Lasso Regression assign penalties to the coefficients of the model in different ways. Ridge Regression uses L2 regularisation; a factor of the summation of the square of the magnitude of the coefficients is added to the loss function [28]. The magnitude of alpha determines the level of penalisation assigned to the L2 regularisation factor.

Lasso Regression, on the contrary, stands for Least Absolute Shrinkage and Selection Operator and uses L1 regularisation where a factor equivalent to the summation of the magnitudes of the coefficient is added to the loss function [28]. Similarly, the value of alpha determines the level of penalisation assigned to the L1 regularization factor.

Finally, ElasticNet Regression combines the properties of both Ridge and Lasso regression. That is, it incorporates a penalty system in which it uses both L1 and L2 regularisation. The shrinkage factor can vary from 0 to 1, where $\alpha = 0$ is identical to performing a ridge regression, and $\alpha = 1$ is identical to performing a lasso regression [28].

Experimental Procedure

6. Pre-Processing

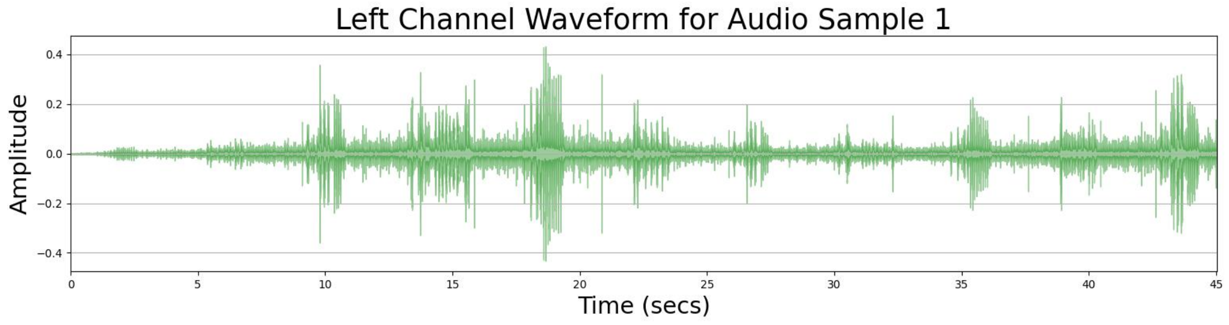
6.1 Data Collection

Binaurally annotated audio samples are rare in the scientific community, seeing as this field of research is in its infancy. Therefore, binaurally produced audio samples needed to be collected individually from YouTube. Note, the set of audio files were selected in such a way that there was a variety in the type of sounds present in the set; there were extracts from music, speech, and background noise. Participants then completed listening tests in which they recorded their emotional response to each of the 38 audio samples. The response options presented to the participants were Happiness, Sadness, Calmness, Fear and Amusement, with the respective labels of 1, 2, 3, 4 and 5. The most common response for each audio sample was the final label given to that sample. This process of sample collection and data labelling through experimental listening tests is entirely credited to Manish Kumar.

6.2 Feature Extraction

The feature extraction process was completed almost entirely on Python with the assistance of a python library called *Librosa*; an open-source, audio signals processing library for temporal and spectral features (Appendix A and B). Extracting the ReTF required separate code, which was provided by Manish Kumar.

Initially, a few important parameters were defined; the hop length was set to 512 samples and the frame length was set to 2048 samples. Then, the first audio file was loaded using the *librosa.load()* function which took the following inputs: the file path, and a Boolean 'mono' value for whether the audio file was a monaural or binaural signal. Since these are binaural recordings, the 'mono' value was set to 'False' which enabled the function to create a two-column matrix, containing the sampled values of the left and right channels of the audio file. These were visualised, as seen in Figure 10. On the contrary, setting 'mono' to True for a binaural audio file, causes the function to load the two channels by creating a single, averaged waveform (Figure 1).



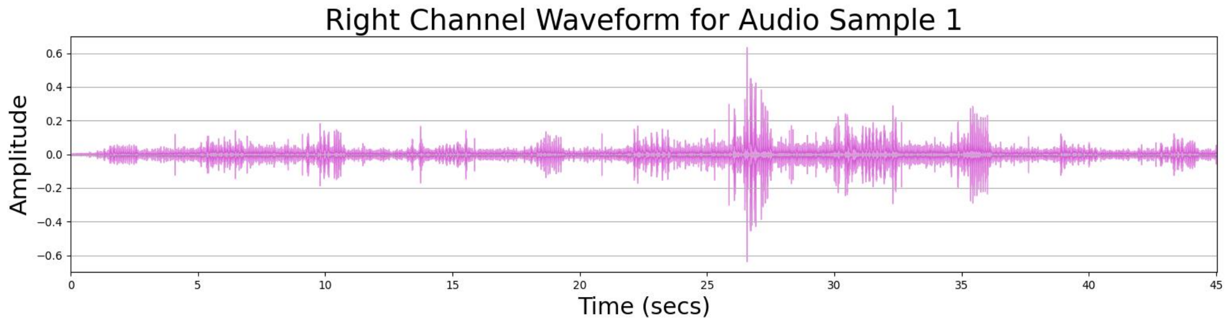


Figure 10 - Left and right channel waveforms for Audio Sample 1

Librosa's feature extraction functions work on a frame-by-frame basis. Therefore, the output of these functions are vectors containing the values for a given feature for each frame spanning the entire signal. The mean, standard deviation and variance of these vectors were found to produce the final features of interest. Note, this process of finding the mean, standard deviation and variance for the vector containing the extracted features, was repeated for all the temporal and spectral features. Furthermore, for all 5 of the features being analysed through Librosa, the features were extracted from the left, the right, and the averaged signal.

To extract the two temporal features, the ZCR and RMS Energy, the signal was passed into the following functions respectively: *librosa.features.zero_crossing_rate()* and *librosa.features.rms()*. The spectrogram was then produced by applying the *librosa.stft()* function. The magnitude and phase can be extracted with the *librosa.magphase()* function, which takes the spectrogram as input. To extract the 3 remaining time-frequency domain features, we use the magnitude of the spectrogram as input into *librosa.feature.spectral_centroid()*, *librosa.features.spectral_rolloff()* and *librosa.feature.mfcc()* functions.

Finally, seeing as no python library functions exist to extract the ReBTF, separate MATLAB code, provided by Manish Kumar, was used to perform the extraction. Figure 11 is a visualisation of the ReBTF for Audio Sample 1.

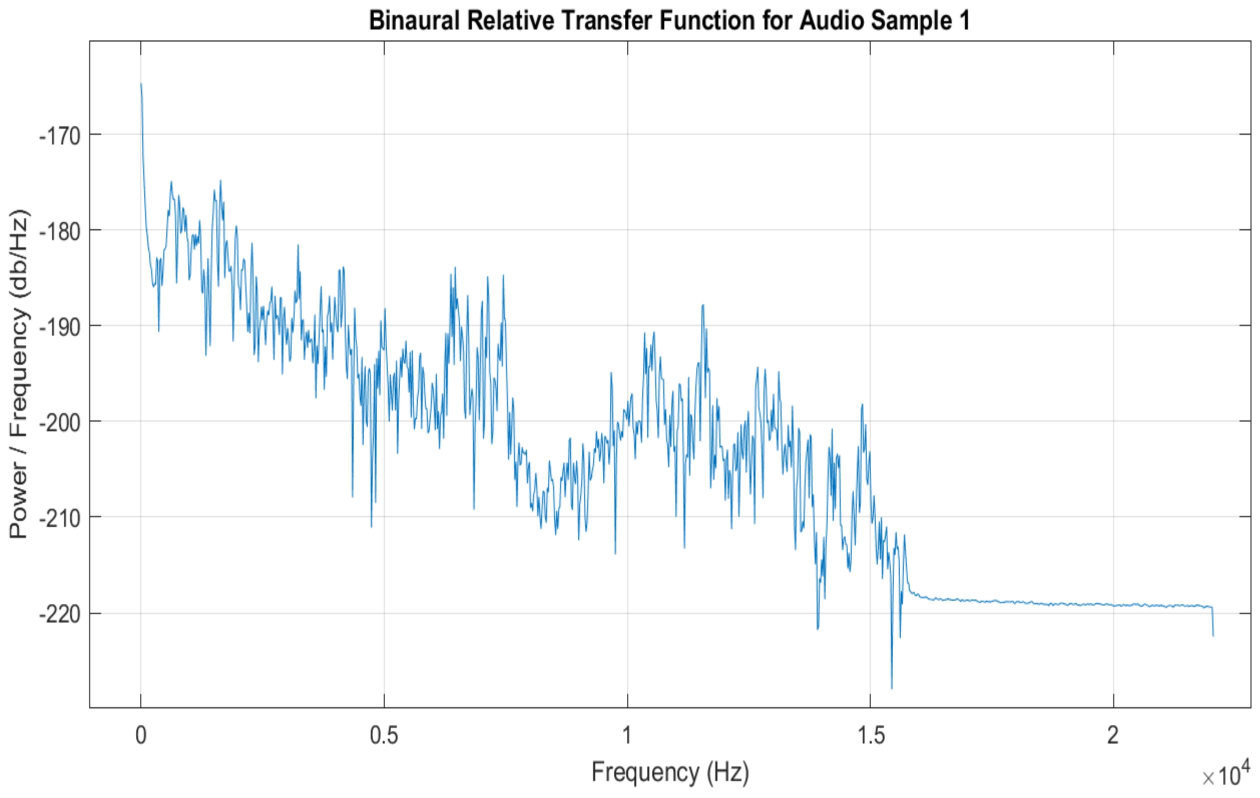


Figure 11 - Relative Binaural Transfer Function for Audio Sample 1

This entire procedure was then run through a loop, to iterate through the entire list of audio samples. With the features extracted from each audio sample, the data was stored in a CSV file which also contained the corresponding emotion labels for each audio sample.

7. Processing with Regression Modelling

The regression analysis was completed in Python with the *sklearn* library (Appendix C). A total of 3 control testing groups were created as shown in Table 1.

Table 1 - The 3 control testing groups used for regression modelling

Test 1A	The temporal and spectral features were extracted from both the L and R channels. This included the 5 features (excluding ReBTF), 3 statistical measures and 2 channels (5 x 3 x 2) resulting in a total of 30 features
Test 2A	The temporal and spectral features were extracted from only the L channel. This included the 5 features (excluding ReBTF), 3 statistical measures and 1 channel (5 x 3 x 1) resulting in a total of 15 features
Test 3A	The L and R channel signals were averaged, and the temporal and spectral features were extracted from this waveform. This included the 5 features (excluding ReBTF), 3 statistical measures and 1 averaged signal (5 x 3 x 1) resulting in a total of 15 features.

The option of choosing only the R channel was deemed redundant since the nature of the signal coming into the right ear versus the left ear is completely arbitrary. The machine would not justify choosing one channel over the other for analysis, and so, a random selection was made in choosing the left channel for analysis in Test 2. Albeit, for the sake of completeness, in future exploration this may be worth exploring to find if there is a difference between analysing the left and right channels.

These were the 3 control tests to see the accuracy of the AER before the inclusion of the spatial feature. For each of the tests, the mean, standard deviation, and variance of the ReBTF of each sample was then included into the feature sets to create Tests 1B, 2B and 3B, which had a feature size of 33, 18 and 18, respectively. This allowed for a direct comparison to be made between the tests in Group A and the tests in Group B, to assess the effect of including the ReBTF on the regression models accuracy. The metrics used to measure the accuracy of the regression models were R^2 and RMSE. R^2 is the fraction of the total sum of squares that is 'explained by' the regression. RMSE is a measure of the average deviation of the estimates from the observed values.

For each of the 6 total tests, the ‘predictor’ and ‘target’ vectors were created. These vectors contained the feature set and the emotion labels respectively, for all the audio samples. It was decided that the percentage of training data would be 80%. This training data was then run through Linear, Ridge, Lasso and ElasticNet regression models. For all the models besides the Linear Regression model, the value of shrinkage factor, alpha, needed to be optimised to give the best result. This was done iteratively, for alpha ranging from 0.001 to 100, increasing in increments of 0.001. This produced 100,000 tested values for alpha. The ideal values for the statistical measures of fit - R^2 and RMSE were then found. (Appendix C).

The limitations of this experimental procedure include the small sample size of 38 audio files, as well as the limited number of options of emotions participants had to choose from. Furthermore, there is a chance that the value of alpha may be further enhanced by looking at a more extensive range of values, as well as using a smaller interval between increments. These limitations are likely to reduce the accuracy of the results.

Results and Discussion

8. Results

Table 2 below shows a summary of the optimal models for each of the 6 tests that were run.

Table 2 - Summary of results from Linear, Lasso, Ridge and ElasticNet regression models for all 6 tests. Yellow highlights indicate the best performer for each test. Green highlight indicates the best performer from all 6 tests.

TEST	Regression Model	RMSE	R^2	Optimised α
Test 1A L and R channels ReBTF excluded	LINEAR	2.413	-1.598	N/A
	RIDGE	1.744	-0.358	8.000
	LASSO	1.740	-0.351	0.123
	ELASTIC	1.736	-0.344	0.184
Test 1B L and R channels ReBTF included	LINEAR	2.047	-0.869	N/A
	RIDGE	1.690	-0.274	3.085
	LASSO	1.633	-0.190	0.071
	ELASTIC	1.650	-0.214	0.109
Test 2A L channel only ReBTF excluded	LINEAR	3.279	-3.796	N/A
	RIDGE	1.769	-0.397	7.933
	LASSO	1.758	-0.380	0.106
	ELASTIC	1.759	-0.380	0.189
Test 2B L channel only ReBTF included	LINEAR	2.753	-2.382	N/A
	RIDGE	1.680	-0.260	1.122
	LASSO	1.633	-0.189	0.058
	ELASTIC	1.645	-0.207	0.088
Test 3A Average of L and R channels ReBTF excluded	LINEAR	2.614	-2.049	N/A
	RIDGE	1.751	-0.369	5.933
	LASSO	1.745	-0.359	0.121
	ELASTIC	1.745	-0.358	0.209
Test 3B Average of L and R channels ReBTF included	LINEAR	2.438	-1.651	N/A
	RIDGE	1.677	-0.255	1.674
	LASSO	1.700	-0.290	0.066
	ELASTIC	1.671	-0.245	0.093

For Test 1A, the best values of RMSE and R^2 were found in the ElasticNet regression model and were 1.736 and -0.344 respectively. The value of α that optimised the model to achieve these values was 0.184. For Test 1B, the best values of RMSE and R^2 were found in the Lasso regression model and were 1.633 and -0.190 respectively. The value of α that optimised the model to achieve these values was 0.071. Test 1B performed better than 1A, suggesting the ReBTF improved performance for this test.

For Test 2A, the best values of RMSE and R^2 were found in the Lasso regression model and were 1.758 and -0.380 respectively. The value of α that optimised the model to achieve these values was 0.106. For Test 2B, the best values of RMSE and R^2 were found in the Lasso regression model and were 1.633 and -0.189 respectively. The value of α that optimised the model to achieve these values was 0.0.058. Test 2B performed better than 2A, suggesting the ReBTF improved performance for this test.

For Test 3A, the best values of RMSE and R^2 were found in the ElasticNet regression model and were 1.745 and -0.358 respectively. The value of α that optimised the model to achieve these values was 0.209. For Test 3B, the best values of RMSE and R^2 were found in the ElasticNet regression model and were 1.671 and -0.245 respectively. The value of α that optimised the model to achieve these values was 0.093. Test 3B performed better than 3A, suggesting the ReBTF improved performance for this test.

The best performing test was Test 2B with an RMSE of 1.633 and an R^2 of -0.189. Test 1B had the same RMSE value of 1.633 but performed marginally worse in terms of R^2 with a value of -0.190.

It can be observed that the linear and ridge regression models were never the best performing models for any of the respective tests.

9. Discussion

9.1 Findings

Based on the results, when comparing the tests that excluded the ReBTF (Tests 1A, 2A and 3A) with those that included it (Tests 1B, 2B and 3B), it is clear to see that the inclusion of the ReBTF always improved the performance of the model. That is, R^2 always increased and RMSE always decreased, as is desired. Table 3 below demonstrates the percentage change of the RMSE and R^2 values for each test, once the ReBTF was included in the feature set.

Table 3 - Percent change of control test results after the inclusion of the ReBTF

Comparison of Tests	**Percent change**	
	RMSE	**R^2**
1A to 1B	Decrease by 5.93%	Increase by 44.77%
2A to 2B	Decrease by 7.15%	Increase by 50.08%
3A to 3B	Decrease by 4.24%	Increase by 31.49%

This was a comparison between the optimal regression model for each respective test. For example, for the first row, the optimal model for Test 1A was the ElasticNet model, while the optimal model for Test 1B was the Lasso model - these were the results that were compared in the first row of Table 3 above.

Note, all the R^2 values were negative for the various models, which seems contradictory given that it is a squared value. The reason for this result is that R^2 is defined to be the proportion of variance explained by the fit, meaning that if the data could be better modelled by a horizontal line as compared to the produced model, then R^2 will be negative. To improve the model, a constant term should be summed to the model [29].

While the study presents interesting findings which potentially opens up an entirely new research space for audio-based emotion recognition, the models performed poorly in comparison to the current standard of AER models, giving rise to questions about the reliability of these findings [2,6]. The poor performance is likely a result of the many assumptions and simplifications made in the experimental procedure, that limit the reliability of the findings.

9.2 Limitations

9.2.1 Using a More Sophisticated ML Algorithm

Primarily, the major improvement that can be made to the present study, is to use a more sophisticated ML algorithm, such as a classification model, for building a model for predicting emotions. As was discussed, the fit of the regression models onto the dataset was poor, as suggested by the negative values for R^2. Furthermore, while regression analysis is an adequate supervised ML technique for introductory ML programming, it is not suited for dealing with a target dataset (the labelled emotions) that is discrete in nature. For both reasons, the use of a more sophisticated classification model would be more appropriate for the task at hand, to produce more reliable results.

9.2.2 Size of Feature Set and Feature Selection Optimisation

The present study only explored the effect of including the ReBTF on a set of 5 temporal and spectral features. While the use of these features was justified by their widespread use in many AER related studies, there is still room to expand the list of features temporal and spectral features used. As an example, Teager-energy-operator (TEO) based audio features have been found to be useful for detecting stress in speech and is one example of a potentially valuable feature to include in the set [2]. In saying this, it is equally important to extract only the features that efficiently and accurately characterise different emotions from an audio signal. Therefore, feature selection optimisation should be another important consideration for improving the current study. There are a few potential techniques for performing such feature selection, including the filter method, wrapper method, the embedded method and forward selection [30, 31].

9.2.3 Greater Granularity in the Emotion Set Used

There were only 5 emotions used in this study to label each audio sample, but this vastly oversimplifies the richness of human emotion to a simple set of 5. There are likely more words for emotions that would have more accurately depicted the emotions felt by participants when listening to the audio samples, and so, this could be a potential improvement in future exploration. On a similar note, transitioning from the categorical conceptualisation of emotions to a dimensional one would innately allow for a more dense set of possible emotions to map onto.

9.2.4 Issues with the Audio Samples

Studies in AER typically have databases of collected audio samples that span into the hundreds [2, 6]. This study was limited in that only 38 audio samples were used, restricting the ability to appropriately train the regression models. Therefore, a larger set of audio samples should be used in future experimentation. The length of each of the audio samples may also have been too long. That is, a large duration of the sample increases the likelihood of there being too large a variety of different sounds in any given sample, which may confuse the regression model.

Further Work and Conclusions

10. Further Work

The present study was the first of its kind, exploring the influence of the spatial characteristics of sound on audio-based emotion recognition. As such, there is an abundance of areas for extension and further exploration.

10.1 Accessing More Spatial Features

Given this study was focused on the impact of spatial audio features on AER, future investigations could explore the impact of other spatial features, such as the ILD and the ITD on AER. This was not possible for this study, since the audio samples were collected from YouTube, meaning the spatial signature of the atmosphere in which the samples were recorded was not accessible. To navigate this dilemma in future, the collection of audio samples could be recorded and sampled directly by the experimenter, to preserve information about the ILD and the ITD of the signal. The difficulty with this is that the experimenter will need access to the recording facilities needed to binaurally reproduce sound, which limits the feasibility of this path of exploration.

10.2 Using a Single Type of Sound

It was intentional that the choice of audio samples varied from music to speech to background noise extracts, however, narrowing this selection to only one type of sound file may be a better approach for preliminary findings. By integrating these three major types of sounds into a single study, the predictive task for the regression model is made more difficult. Rather, focusing on just a single type of sound would allow for the model to be optimised for this particular sound, which can then be built upon further with the addition of other sound types.

11. Conclusion

The present study explored the hypothesis of whether the inclusion of a spatial audio feature, the Relative Binaural Transfer Function, into a set of temporal and spectral features could improve the performance of an AER model. Through the data collection, feature extraction and regression analysis processes, it was shown that the ReBTF does, in fact, improve AER performance. These are, however, only preliminary findings, seeing as the standard of the regression models was extremely poor when compared to previous studies in AER. Nevertheless, this study presents an interesting finding upon which future experiments could further test and validate, to improve the standard of AER and ultimately, contribute to enabling a more personable human-to-machine interaction.

Bibliography

1) Lyon, R. F. (2010). "Machine hearing: An emerging field [exploratory dsp]." IEEE signal processing magazine 27(5): 131-139.

2) El Ayadi, M., et al. (2011). "Survey on speech emotion recognition: Features, classification schemes, and databases." Pattern Recognition 44(3): 572-587.

3) Seo, Y.-S. and J.-H. Huh (2019). "Automatic Emotion-Based Music Classification for Supporting Intelligent IoT Applications." Electronics 8(2).

4) Kerkeni, L., et al. (2019). Automatic Speech Emotion Recognition Using Machine Learning: https://www.intechopen.com/online-first/automatic.

5) Kim, Y., et al. (2010). "Music emotion recognition: A state of the art review." Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010.

6) Yang, Y.-H. and H. H. Chen (2012). "Machine Recognition of Music Emotion." ACM Transactions on Intelligent Systems and Technology 3(3): 1-30.

7) Ramdinmawii, E., et al. 2017. Emotion recognition from speech signal, IEEE.

8) Er, M. B. (2020). "A Novel Approach for Classification of Speech Emotions Based on Deep and Acoustic Features." IEEE Access 8: 221640-221653.

9) Rainville, P., et al. (2006). "Basic emotions are associated with distinct patterns of cardiorespiratory activity." International Journal of Psychophysiology 61(1): 5-18.

10) Hevner, K. (1935). "Expression in music: a discussion of experimental studies and theories." Psychological Review 42(2): 186-204.

11) Eerola, T. (2010). "Analysing Emotions in Schubert'S Erlkönig: a Computational Approach." Music Analysis 29(1/3): 214-233

12) Cowie, R., et al. (2001). "Emotion recognition in human-computer interaction." Signal Processing Magazine, IEEE 18: 32-80.

13) Peeters, G. (2004). "A large set of audio features for sound description (similarity and classification) in the CUIDADO project."

14) Valerdo, V., 2020 [1]. Understanding Audio Signals for Machine Learning. [video] Available at: <https://www.youtube.com/watch?v=daB9naGBVv4>

15) Alías, F., et al. (2016). "A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds." Applied Sciences 6: 143.

16) Valerdo, V., 2020 [2]. Understanding Time Domain Audio Features. [video] Available at: <https://www.youtube.com/watch?v=SRrQ_v-OOSg>.

17) Heckbert, P. S. (1998). Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm.

18) Kehtarnavaz, N. (2008). CHAPTER 7 - Frequency Domain Processing. Digital Signal Processing System Design (Second Edition). N. Kehtarnavaz. Burlington, Academic Press: 175-196.

19) Lyon, D. (2009). "The Discrete Fourier Transform, Part 4: Spectral Leakage." Journal of Object Technology 8

20) Valerdo, V., 2020 [3]. Short-Time Fourier Transform Explained Easily. [video] Available at: <https://www.youtube.com/watch?v=-Yxj3yfvY-4>

21) Schubert, E. and J. Wolfe (2006). "Does Timbral Brightness Scale with Frequency and Spectral Centroid." Acustica 92: 820.'

22) Valerdo, V., 2020 [4]. Frequency-Domain Audio Features. [video] Available at: <https://www.youtube.com/watch?v=3-bjAoAxQ9o>

23) Martinez, J., et al. (2012). "Speaker recognition using Mel Frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques."

24) Cuadrado, F., et al. (2020). "Arousing the Sound: A Field Study on the Emotional Impact on Children of Arousing Sound Design and 3D Audio Spatialization in an Audio Story." Frontiers in Psychology 11(737).

25) Fletcher, M., 2011. The Effect Of Spatial Treatment Of Music On Listener's Emotional Arousal. Journal on the Art of Record Production, [online] (05). Available at: <https://www.arpjournal.com/asarpwp/the-effect-of-spatial-treatment-of-music-on-listener%E2%80%99s-emotional-arousal/>

26) Cheng, C. I. and G. H. Wakefield (2001). "Introduction to head-related transfer functions (HRTFs): Representations of HRTFs in time, frequency, and space." AES: Journal of the Audio Engineering Society 49:

27) Laitinen, M.-V. and V. Pulkki (2009). "Binaural reproduction for Directional Audio Coding." 2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics: 337-340.

28) Maina, S., 2021. Preventing Overfitting with Lasso, Ridge and Elastic-net Regularization in Machine Learning. [online] towards data science. Available at: <https://towardsdatascience.com/preventing-overfitting-with-lasso-ridge-and-elastic-net-regularization-in-machine-learning-d1799b05d382>

29) Figueiredo, D., et al. (2011). "What is R2 all about?" Leviathan-Cadernos de Pesquisa Polútica 3: 60-68.

30) Nishant Shah. (2020). "Feature Selection Techniques". Available at: https://medium.datadriveninvestor.com/feature-selection-techniques-1a99e61da222 .

31) Ingale, A. and Chaudhari, D., 2012. Speech Emotion Recognition. International Journal of Soft Computing and Engineering, [online] 2(1). Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.457.2825&rep=rep1&type=pdf

Appendices

Appendix A: Code for Feature Visualisation

```
import warnings
import numpy as np
import librosa.display
import math
import matplotlib.pyplot as plt



''' HOUSE KEEPING '''

# ignore warnings
warnings.filterwarnings("ignore")

# loading the audio file with librosa
file = 'ER Binaural Dataset/1.wav'      # define the file path
signal, sr = librosa.load(file)         # samples / sr = 45 sec
binaural_signal, sr = librosa.load(file, mono=False)
left = binaural_signal[0]
right = binaural_signal[1]
average = (left + right) / 2


# important parameters
FRAME_LENGTH = 1024                                 # the number of samples considered
in each frame of the analysis
HOP_LENGTH = 512                                    # number of samples shifted to rig
ht to define each new frame
TOTAL_SAMPLES = len(signal)                         # 993270 for audio 1
FRAME_NUMBER = math.ceil(TOTAL_SAMPLES / HOP_LENGTH)    # 1940 for audio 1
frames = range(FRAME_NUMBER)                        # range(1940) is 0 to 1939 for aud
io 1
t = librosa.frames_to_time(frames=frames, hop_length=HOP_LENGTH)
# splits 1940 values across the time duration of the signal which is 993270 / 22050 ~= 45
 sec since 22050 samples take
# one second to be sampled and there are 993270 samples in total so it takes 993270 / 2205
0 seconds to sample useful
# helper functions

def pltplot(y, title, xlabel, ylabel):
    plt.plot(y)
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

''' TIME DOMAIN FEATURES '''

# Waveform
```

```
plt.figure()
librosa.display.waveplot(signal, sr=sr, alpha=0.4, color='b')
plt.xlabel("Time (secs)")
plt.ylabel("Amplitude")
plt.title("Averaged Channels Waveform for Audio Sample 1", fontsize=13)
plt.grid(axis = 'y')
plt.show()

plt.figure()
librosa.display.waveplot(left, sr=sr, alpha=0.4, color='g')
plt.xlabel("Time (secs)", fontsize=21)
plt.ylabel("Amplitude", fontsize=21)
plt.title("Left Channel Waveform for Audio Sample 1", fontsize=26)
plt.grid(axis = 'y')
plt.show()

plt.figure()
librosa.display.waveplot(right, sr=sr, alpha=0.4, color='m')
plt.xlabel("Time (secs)", fontsize=21)
plt.ylabel("Amplitude", fontsize=21)
plt.grid(axis = 'y')
plt.show()

# Zero Crossing Rate
zcr_signal = librosa.feature.zero_crossing_rate(signal, frame_length=FRAME_LENGTH, hop_len
gth=HOP_LENGTH)[0]
                                        # since shape is (1, 1940), zero index to access i
nner array
plt.figure(figsize=(15, 17))
plt.plot(t, zcr_signal * FRAME_LENGTH, color = 'tab:orange')
plt.title("Zero Crossing Rate for Audio Sample 1")
plt.xlabel("Time (secs)")
plt.ylabel("Number of Zero Crossings Per Frame")
plt.grid(axis = 'y')
plt.show()


# RMSE
rms_signal = librosa.feature.rms(signal, frame_length=FRAME_LENGTH, hop_length=HOP_LENGTH)
[0]
plt.figure(figsize=(15, 17))
t = librosa.frames_to_time(frames=frames, hop_length=HOP_LENGTH)
librosa.display.waveplot(signal, alpha=0.25)
plt.plot(t, rms_signal, label='RMS Energy', color="m")
plt.legend(loc='upper right')
plt.title("Root-Mean-Square Energy for Audio Sample 1")
plt.xlabel("Time (secs)")
plt.ylabel("Amplitude")
plt.grid(axis = 'y')
plt.show()



''' FREQUENCY DOMAIN FEATURES '''
```

```
# take fft (spectrum)
fft = np.fft.fft(signal)        # numpy 1D array which has as many values as the number of
samples (22050 * 45 sec).
mag = np.abs(fft)               # gets the positive values of fft


# PLOT: spectrum vs samples -> fourier transform WITHOUT complex values
plt.figure(figsize=(15, 17))
pltplot(mag, "Magnitude of Fast Fourier Transform vs Samples", "Sample", "Amplitude")
plt.plot(color = 'c', alpha=0.4)
plt.show()

# PLOT: (relevant) magnitude of spectrum vs frequency -> WITHOUT complex (phase) values
plt.figure(figsize=(15, 17))
freq = np.linspace(0, sr, len(mag))         # 0 to 22050Hz with as many intervals as there
are samples
relevant_mag = mag[:int((len(mag))/2)]      # ft duplicated around nyguist frequency
relevant_freq = freq[:int((len(freq))/2)]
plt.plot(relevant_freq, relevant_mag, color = 'r', alpha=0.4)
plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude")
plt.grid(axis = 'y')
plt.xlim(-200, 12000)
plt.ylim(0, 600)
plt.title("Frequency Spectrum from Fast Fourier Transform for Audio Sample 1")
plt.show()

# the 'frequency' and 'magnitude' arrays together are telling us how much each freq is con
tributing to the overall
# sound. Produces the power spectrum focusing on only half of the sample rate. The energy
 is distributed between the
# lower and mid freq range.
# PROBLEM: this is a static snapshot of the whole sound, and its averaging the different f
requency bands throughout the
# whole sound. What we want to do is to see how these individual frequency bands contribut
e to the overall sound over time.
# We do using the short time fourier transform (STFT) -> spectrogram gives info about ampl
itude as function of both
# frequency and time where it takes the fourier transform over smaller 'windows' of the wh
ole signal


''' TIME-FREQUENCY DOMAIN FEATURES '''

# spectrogram
plt.figure()
stft = librosa.core.stft(signal, hop_length = HOP_LENGTH, n_fft=FRAME_LENGTH)
spectrogram = np.abs(stft)
mag, phase = librosa.magphase(spectrogram)
librosa.display.specshow(spectrogram, sr=sr, hop_length=HOP_LENGTH, x_axis="time", y_axis
="linear")
plt.xlabel("Time (secs)")
plt.ylabel("Frequency (Hz)")
plt.title("Frequency Spectrogram from Short-Time Fourier Transform for Audio Sample 1")
plt.colorbar()                        # heat map of how amplitude varies with time and fre
quency.
                                      # there's a way of visualising this in a way that ma
kes more sense in the way we percieve
                                      # loudness, which isnt linear (as it is here), but i
t's rather, logarithmic.
                                      # hence, the log spectrogram
plt.show()

# log amplitude spectrogram -> amplitude is from 0 to 20 but most of the signals have ener
gy between -60db to 10db
plt.figure()
log_spectrogram = librosa.power_to_db(spectrogram) # used powertodb instead of amplitudeto
db because on the librosa
                                                    # website it says that the stft functio
n creates a POWER spectrogram
librosa.display.specshow(log_spectrogram, sr=sr, hop_length=HOP_LENGTH, x_axis="time", y_a
xis="linear")
plt.xlabel("Time (secs)")
plt.ylabel("Frequency (Hz)")
plt.title("log Magnitude vs Frequency vs Time for Audio Sample 1")
plt.colorbar()
plt.show()

# amplitude log frequency spectrogram
plt.figure()
librosa.display.specshow(log_spectrogram, sr=sr, hop_length=HOP_LENGTH, y_axis="log", x_ax
is="time",)
plt.xlabel("Time (secs)")
plt.ylabel("log Frequency (Hz)")
plt.title("log Magnitude vs log Frequency vs Time for Audio Sample 1")
plt.colorbar()
plt.show()

# spectral centroid and rolloff
plt.figure()
centroid = librosa.feature.spectral_centroid(signal, sr=sr)
librosa.display.specshow(log_spectrogram, y_axis='log', x_axis='time')
plt.plot(t, centroid.T, label='Spectral centroid', color='g')
plt.title("Spectral Centroid and Rolloff for Audio Sample 1")
rolloff = librosa.feature.spectral_rolloff(signal, sr=sr, roll_percent=0.95)[0]
rolloff_min = librosa.feature.spectral_rolloff(signal, sr=sr, roll_percent=0.85)[0]
librosa.feature.spectral_rolloff(S=mag, sr=sr)
plt.plot(librosa.times_like(rolloff), rolloff_min, color='w',
        label='Roll-off frequency (0.85)')
plt.legend(loc='lower right')
plt.ylabel("log Frequency (Hz)")
plt.xlabel("Time (secs)")
plt.show()


# MFCCs
plt.figure()
```

```
MFCCs = librosa.feature.mfcc(signal, n_fft = FRAME_LENGTH, hop_length = HOP_LENGTH, n_mfcc
= 20)
librosa.display.specshow(MFCCs, x_axis='time', sr=sr, hop_length=HOP_LENGTH)
plt.xlabel("Time (secs)")
plt.ylabel("Mel Frequency Cepstral Coefficients (0 - 20)")
plt.title("Mel Frequency Cepstral Coefficients for Audio Sample 1")
plt.colorbar()
plt.show()
```

Appendix B: Code for Feature Extraction

```
import numpy as np
import librosa.display
import csv
import warnings

warnings.filterwarnings("ignore")

with open('extracted_features.csv', 'w', newline='') as f:
    thewriter = csv.writer(f)
    thewriter.writerow([  'sample #',
                          'zcr_mean',
                          'L_zcr_mean', 'L_zcr_std', 'L_zcr_var',
                          'R_zcr_mean', 'R_zcr_std', 'R_zcr_var',
                          'L_rms_mean', 'L_rms_std', 'L_rms_var',
                          'R_rms_mean', 'R_rms_std', 'R_rms_var',
                          'L_centroid_mean', 'L_centroid_std', 'L_centroid_var',
                          'R_centroid_mean', 'R_centroid_std', 'R_centroid_var',
                          'L_rolloff_mean', 'L_rolloff_std', 'L_rolloff_var',
                          'R_rolloff_mean', 'R_rolloff_std', 'R_rolloff_var',
                          'L_mfcc_mean', 'L_mfcc_std', 'L_mfcc_var',
                          'R_mfcc_mean', 'R_mfcc_std', 'R_mfcc_var'
                          'retf_mean', 'retf_std', 'retf_var'
                        ])

    # folder with audio samples
      dataset = 'ER Binaural Dataset'

    # important parameters
      N_FFT = FRAME_LENGTH = WIN_LENGTH = 2048  # samples in each frame
      HOP_LENGTH = 512  # samples hopped for windowing

    file = 1

    while (file < 39):

        # define the path
          path = dataset + "/" + str(file) + ".wav"
          print(path)

        # load the left and right channels from a binaural signal matrix
          binaural_signal, sr = librosa.load(path, mono=False)
          left = binaural_signal[0]
          right = binaural_signal[1]
```

```
#load signal without binarual consideration
  mono_signal, _ = librosa.load(path, mono=True)
  zcr = librosa.feature.zero_crossing_rate(y=mono_signal,
                                          frame_length=FRAME_LENGTH,
                                          hop_length=HOP_LENGTH)
  zcr_mean = np.mean(zcr)


# ZCR (left)
  L_zcr = librosa.feature.zero_crossing_rate(y=left,
                                            frame_length=FRAME_LENGTH,
                                            hop_length=HOP_LENGTH)
  L_zcr_mean = np.mean(L_zcr)
  L_zcr_std = np.std(L_zcr)
  L_zcr_var = np.var(L_zcr)

# ZCR (right)
  R_zcr = librosa.feature.zero_crossing_rate(y=right,
                                            frame_length=FRAME_LENGTH,
                                            hop_length=HOP_LENGTH)
  R_zcr_mean = np.mean(R_zcr)
  R_zcr_std = np.std(R_zcr)
  R_zcr_var = np.var(R_zcr)


# RMS energy (left)
  L_rms = librosa.feature.rms(y=left,
                             frame_length=FRAME_LENGTH,
                             hop_length=HOP_LENGTH)
  L_rms_mean = np.mean(L_rms)
  L_rms_std = np.std(L_rms)
  L_rms_var = np.var(L_rms)

# RMS energy (right)
  R_rms = librosa.feature.rms(y=right,
                             frame_length=FRAME_LENGTH,
                             hop_length=HOP_LENGTH)
  R_rms_mean = np.mean(R_rms)
  R_rms_std = np.std(R_rms)
  R_rms_var = np.var(R_rms)


# spectral centroid (left)
  L_centroid = librosa.feature.spectral_centroid(y=left,
                                                n_fft=N_FFT,
                                                hop_length=HOP_LENGTH)
  L_centroid_mean = np.mean(L_centroid)
  L_centroid_std = np.std(L_centroid)
  L_centroid_var = np.var(L_centroid)
```

```
# spectral centroid (right)
  R_centroid = librosa.feature.spectral_centroid(y=right,
                                                  n_fft=N_FFT,
                                                  hop_length=HOP_LENGTH)
  R_centroid_mean = np.mean(R_centroid)
  R_centroid_std = np.std(R_centroid)
  R_centroid_var = np.var(R_centroid)


# spectral rolloff (left)
  L_rolloff = librosa.feature.spectral_rolloff(y=left,
                                                n_fft=N_FFT,
                                                hop_length=HOP_LENGTH,
                                                roll_percent=0.85)
  L_rolloff_mean = np.mean(L_rolloff)
  L_rolloff_std = np.std(L_rolloff)
  L_rolloff_var = np.var(L_rolloff)

# spectral rolloff (right)
  R_rolloff = librosa.feature.spectral_rolloff(y=right,
                                                n_fft=N_FFT,
                                                hop_length=HOP_LENGTH,
                                                roll_percent=0.85)
  R_rolloff_mean = np.mean(R_rolloff)
  R_rolloff_std = np.std(R_rolloff)
  R_rolloff_var = np.var(R_rolloff)


# mfccs (left)
  L_mfcc = librosa.feature.mfcc(y=left,
                                n_fft=N_FFT,
                                hop_length=HOP_LENGTH,
                                n_mfcc=20)
  L_mfcc_mean = np.mean(L_mfcc)
  L_mfcc_std = np.std(L_mfcc)
  L_mfcc_var = np.var(L_mfcc)

# mfccs (right)
  R_mfcc = librosa.feature.mfcc(y=right,
                                hop_length=HOP_LENGTH,
                                n_mfcc=20)
  R_mfcc_mean = np.mean(R_mfcc)
  R_mfcc_std = np.std(R_mfcc)
  R_mfcc_var = np.var(R_mfcc)
```

```
# write to csv file
  thewriter.writerow([file,
                            zcr_mean,
                            L_zcr_mean, L_zcr_std, L_zcr_var,
                            R_zcr_mean, R_zcr_std, R_zcr_var,
                            L_rms_mean, L_rms_std, L_rms_var,
                            R_rms_mean, R_rms_std, R_rms_var,
                            L_centroid_mean, L_centroid_std, L_centroid_var,
                            R_centroid_mean, R_centroid_std, R_centroid_var,
                            L_rolloff_mean, L_rolloff_std, L_rolloff_var,
                            R_rolloff_mean, R_rolloff_std, R_rolloff_var,
                            L_mfcc_mean, L_mfcc_std, L_mfcc_var,
                            R_mfcc_mean, R_mfcc_std, R_mfcc_var
                      ])

# increment to the next file
  file += 1
```

Appendix C: Code for Regression Analysis and Optimisation of Alpha

```
import csv
import warnings

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from yellowbrick.regressor import PredictionError

# suppress warnings for clear workspace

warnings.filterwarnings('ignore')


with open('3_alpha_test_final.csv', 'w', newline='') as f:
    thewriter = csv.writer(f)
    thewriter.writerow([  'alpha',
                          'lin rmse',
                          'lin r2',
                          'ridge rmse',
                          'ridge r2',
                          'lasso rmse',
                          'lasso r2',
                          'enet rmse',
                          'enet r2',
                          ])

    '''reading the data and performing basic data checks'''
    data = pd.read_csv('3_extracted_features.csv')

    ''' creating arrays for the features and the response variable'''

    # extract the target column
    target_column = ['label']
    sample_num_column = ['sample #']

    # define the data as a set
    data_set = list(data.columns)

    # remove the target column from data, to create the predictor set. sets are used as th
ere cannot be duplicates in sets
```

```
    predictors = list(set(data_set) - set(target_column) - set(sample_num_column))
    print(len(predictors))

    # normalising each column
    data[predictors] = data[predictors] / data[predictors].max()

    '''creating the training and test datasets'''

    # create an array (38 rows x 31 columns) for just the values of each feature
    X = data[predictors].values

    # create an array (38 rows x 1 column) for the target which is the label
    y = data[target_column].values

    # create the test (80% ~= 31 audios) and training (20% ~= 7 audios) datasets.
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.80, random_state
=40)
    print(X_train.shape); print(X_test.shape)

    '''build, predict and evaluate the regression model'''

    alpha = 0.001

    # lasso regression

    while (alpha <= 100):

        print(alpha)
        # linear regression mode
        lr = LinearRegression()
        lr.fit(X_train, y_train)
        pred_test_lr = lr.predict(X_test)
        lin_RMSE = np.sqrt(mean_squared_error(y_test, pred_test_lr))
        lin_r2 = r2_score(y_test, pred_test_lr)

        # ridge regression

        rr = Ridge(alpha=alpha)
        rr.fit(X_train, y_train)
        pred_test_rr = rr.predict(X_test)
        ridge_RMSE = np.sqrt(mean_squared_error(y_test, pred_test_rr))
        ridge_r2 = r2_score(y_test, pred_test_rr)

        # lasso regression

        model_lasso = Lasso(alpha=alpha)
        model_lasso.fit(X_train, y_train)
        pred_test_lasso = model_lasso.predict(X_test)
        lasso_RMSE = np.sqrt(mean_squared_error(y_test, pred_test_lasso))
        lasso_r2 = r2_score(y_test, pred_test_lasso)

        # Elastic Net
        model_enet = ElasticNet(alpha=alpha)
        model_enet.fit(X_train, y_train)
        pred_test_enet = model_enet.predict(X_test)
        enet_RMSE = np.sqrt(mean_squared_error(y_test, pred_test_enet))
        enet_r2 = r2_score(y_test, pred_test_enet)


        thewriter.writerow([alpha,
                            lin_RMSE, lin_r2,
                            ridge_RMSE, ridge_r2,
                            lasso_RMSE, lasso_r2,
                            enet_RMSE, enet_r2,
                            ])

        # increment to the next file
        alpha += 0.001
```