Explorations in Timbre

Joshua William La Pine

a thesis submitted for the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

05 June 2018

Abstract

Timbre is the quality of sound that enables us to recognise different instruments. The most successful methods for automatic instrument recognition are performed using information from sound that does not mean anything to the musician and is not directly related to the perception of timbre. Instead we start with an attribute of sound that is known to affect the way we perceive timbre. Two such attributes are the harmonic content of the sound and the "envelope" which describes the way volume rises and falls. This thesis explored several methods for detecting envelopes for the purpose of timbre description. A tool was developed that allows a user to match a synthetic wave to recorded audio by adding and adjusting amplitude and frequency control points. The resulting synthetic wave sounds subjectively close to the real audio. This provides avenues for future psychoacoustic research.

Acknowledgements

To my supervisors Geoff Wyvill and David Eyers for their insight and patience. To my Mum, Dad, Brooke, Rachael, and Grams without whom I would have achieved nothing. To Little Sparrow for always looking after me, and encouraging me when I needed it. To Nat for being a true dawg. To Matt for the beach trip. To Meredith for relating to the struggle. To all the staff and students of the graphics lab over the years, particularly Lewis, Reuben, Johnny, Hamza, and Steve, for your knowledge and deep capacity for rubber ducking. And to all of my family and friends. Thank you, everyone, I could not have done this without you.

Contents

List of Figures

Chapter 1

Introduction

1.1 What is Timbre?

Timbre is defined by the Acoustical Society of America (ASA) as "that attribute of auditory sensation which enables a listener to judge that two nonidentical sounds, similarly presented and having the same loudness and pitch, are dissimilar", and additionally, "Timbre depends primarily upon the frequency spectrum, although it also depends upon the sound pressure and the temporal characteristics of the sound" Association (1960). Less formally timbre can be thought of as the audible core of an instrument, its characteristic sound or 'tonal colour'. Anyone is capable of easily discerning that there is an audible difference between a note played on a guitar and a note played on the piano and the first part of this definition captures that. But timbre and what it is comprised of is much more complex. The above definition in its first part tells us essentially what timbre is *not* and the second part is vague.

One interpretation of the definition is that we should ignore pitch and volume when making similarity judgements, but that does not mean that pitch and volume are irrelevant to timbre perception, far from it. There are examples where a higher pitched note produces a different timbre than a lower pitched note for a given instrument. And we know that the rise and fall in the volume of a given note on a particular instrument has a pronounced effect on the way we perceive its timbre. This is approximately captured by the second part of the above definition but not in much detail and that is the core of the problem: we do not have a precise definition for timbre.

1.2 Why is Defining Timbre so Difficult?

1.2.1 Human Perception and Timbre Ambiguity

Because timbre is a function not only of physical phenomena but also of human perception there is a huge degree of subjectivity when making judgements of similarity. For example, a high pitched trumpet note can be hard to distinguish from a high pitched note played on a saxophone. The reason discerning the difference between notes at higher pitches becomes more difficult is that the overtones go beyond the range of human hearing, we simply aren't able to hear as much of the make-up of pitched sounds at higher frequencies. Skilled musicians have trained ears and therefore are capable of perceiving finer degrees of difference in timbre but this example still poses a challenge. To give a non musical example, chips frying in a deep fryer can sound remarkably similar to rain falling on a tin roof. The sources of these sounds are completely different and yet perceptually they are exceedingly similar and if we are perceiving them to be similar then there must be some fundamental similarity in the sounds themselves.

1.2.2 Range of Instrument Timbre

One might intuitively think of instrument timbre as being a discrete property that would allow us to describe a multidimensional space where each point in that space represents an instrument's timbre. However, this is not accurate. While the ASA's definition makes it sound as if a given instrument has a particular timbre, it is more apt to think of an instrument as having a range of different timbres. A guitar, for example, can produce a number of different timbres depending on what note is being played. If a note is played using a fret that is closer to the bridge, where the strings meet the body of the guitar, not only will the note be a higher pitch but the quality of the sound will differ from a note fretted closer to the headstock or somewhere in between. A note played by plucking near the middle of the string tends to sound more pure, more sinusoidal, but a note played by plucking near the bridge might be described as sounding more percussive, as the sound of the string being plucked is more audible relative to the sound of the note.

Idiosyncrasies like this are present for many instruments. For example, the volume of a note played on the piano has an audible change when the note is decaying where it suddenly seems to get louder. This is a phenomenon unique to the piano referred to colloquially as 'blooming' or in the literature as 'double decay' and, while the perception

is that the note gets louder, what is really happening is that the rate at which the note gets quieter has decreased (Weinreich, 1977).

Another difficulty and motivating factor for researching timbre is that the language used to describe sound is itself open to interpretation. An instrument, note, or musical passage might be described as sounding bright, dark, sad, mysterious, crunchy, or dancing, and these terms might mean different things to different people. If there was a precise definition for timbre then we could describe sounds in terms of the parameters of that definition.

1.2.3 The Musician's Style and Technique

As stated, the specific note being played on a given instrument has a large potential effect on the timbre produced, but of more significance is the style of, and techniques used by the individual musician. I will explain further using the guitar as an example due to my experience with the instrument. On a guitar you might pluck a note with the flesh of the thumb, or the nail of the index finger, or further still a plectrum could be used. These techniques produce easily discernible differences in timbre and the musician can make use of a vast number of other techniques in their playing that affect the timbre. On a guitar the musician might hammer-on a note which involves quickly pressing down on a fret without plucking the string. The sound this produces is more percussive than a plucked note. A related technique is the pull-off which typically involves fretting two notes on the same string, plucking the string, and then pulling the finger off the higher fret resulting in a percussive initial sound and, more importantly, the second note retains some of the quality of the first. Further still a musician might employ sweep picking where, using a plectrum, notes in an arpeggio or scale are plucked with the hand moving smoothly in a single direction. The timbre produced by this is very smooth as the individual plucking of each string is less pronounced and so does not punctuate the beginning of each note.

These are just a few examples; but the number of different techniques that produce different timbres on a guitar is large. And this is just one instrument; there are many where the scope of possible timbre is significantly larger. The cello is a good example as it is widely considered to have the largest range of possible timbre. So the question is how do we attempt to capture these nuances?

1.3 Music Information Retrieval

In an area of research called music information retrieval (MIR) much work has been done in an effort to classify musical timbre. Research in MIR can be split into categories such as genre and mood classification, artist identification, music annotation and instrument classification.

More broadly, music information retrieval is centred on extracting information from musical recordings. Digital musical recordings are created by sampling sound at particular intervals and recording the deviation from the average air pressure. So a sound file may be as simple as a sequence of numbers which each represent the deviation from average air pressure at the location of the detector at that instant in time. The interval between the samples is dependent on the sample rate which is a measure of how often the sound is sampled per second. These recordings, or 'waveforms' are what music information retrieval techniques operate on.

Each category of musical information retrieval makes use of various techniques to extract features from music. A feature is defined as any salient piece of information and there is a distinction between so-called low and high level features. Research into genre or mood classification, for example, is more focused on high level features such as pitch and rhythm, whereas instrument classification is achieved through low level features.

1.3.1 Instrument Classification

There are a large number of established features that can be extracted from waveforms by the use of various algorithms or filters but we are interested in a more conceptually complex subset. These low level features essentially work by calculating the result of some mathematical process where the input is a waveform. Low level features don't take into account musical information like pitch or rhythm and from the perspective of the musician, or even the scientist, the motivation for the use of some of these features can be unclear. One such example is the Mel-Frequency Cepstral Coefficients or MFCCs (Siedenburg, Fujinaga, and McAdams, 2016). This feature, or rather set of features, is obtained through a complex sequence of mathematical processes that result in a set of numbers which can be thought of as a compact representation of the spectrum of frequencies present in the waveform. These MFCCs originated in speech recognition and have found increasing use in MIR due to their usefulness in classification. MFCCs and other features might have a proven application in instrument recognition but in

the process nothing is learned about timbre. We get a result but are no closer to understanding why we got this result. We are no closer to that precise definition of timbre.

1.3.2 A New Timbre Model

There are several things known to influence timbre, including: the way the volume of a note rises and falls, the 'noise' or instrument specific sounds present throughout the note, and the harmonic content of the note. If we could build a system that detects some of these aspects of timbre and uses them to classify a subset of instruments then we might achieve something closer to a precise definition of timbre; a musically and perceptually meaningful description of timbre. To this end I devised a system for instrument classification based on known, perceivable elements of timbre.

The initial conception of the model involved a sequence of steps. Firstly, an approximate measure of the overall loudness of the sound would be obtained and used to normalise the sound. The next step would be to capture the sound's 'envelope', which can be thought of as a curve that describes the rise and fall of sound. And the final step of this system would be to analyse the sound's harmonic content, which is known to be the primary factor in distinguishing timbre. I began by researching and implementing envelope detection methods. This area was significantly more complicated than first anticipated which lead to envelopes being the core focus of my thesis. Envelopes are explored in Chapter 3. With respect to harmonic content, I also discovered that many instruments exhibit frequency variation over the course of sounding a single note.

As a result of this research, I propose a new partial model for timbre consisting of a single repeated cycle with an applied envelope curve and frequency adjustment. This model for timbre was explored through the development of a wave editing program that allows the user to match a synthetically generated wave to reference wave recorded from a musical instrument. This is described in Chapter 4.

Chapter 2

Background

Timbre is the core characteristic of sound, the attribute that allows us to tell the difference between different instruments when other attributes such as pitch, loudness, and duration are equal. The issue is that a precise description of timbre has not been determined. In other words, we do not know exactly what causes the difference in the way we perceive the sound of notes played on different instruments. While timbre is not only a property of the sounds made by musical instruments this thesis is focussed on musical instrument timbre. Additionally the work in this thesis is partially motivated by the lack of perceptual relevance of sound information used in instrument classification. So to better understand the work in this thesis it is necessary to discuss certain aspects of research related to timbre and automatic instrument classification, the goals of these categories of research, and how the two are related.

While the background on timbre is necessary to understand the motivation behind this thesis, the focus is envelopes. Existing envelope detection techniques are explored in section 2.4.

2.1 Timbre

Research relating to timbre is a complex space with work in different areas. The most relevant area of research is the field of psychoacoustics in which researchers are concerned with how sounds are perceived. A lot of the work done in psychoacoustics is related to finding the elements or "features" in sound that are most perceptually relevant and which will bring us closer to a precise definition of timbre. The ultimate question timbre researchers seek to answer is: What is the smallest set of information that can be extracted from sound that uniquely represents its timbre? The answer

to that question would dramatically enhance how we process audio. For example, provided segmentation of the instruments in a digital recording was a solved problem, we could perfectly identify the instruments present in recording. This would have a large impact on the field of Music Information Retrieval (MIR) which is discussed later in this chapter.

From the perspective of musical synthesis an exact description of timbre could enable us to more effectively explore the range of possible timbre resulting in new creative possibilities.

So how do timbre researchers attempt to discover the precise definition for timbre? One of the core problems in timbre research is the multidimensional nature of the attribute (Licklider, 1951). As there are a huge number of possible features in sounds that might affect the perceived timbre it is difficult to know where to begin. Some researchers start by taking aspects of sound, varying them methodically and then performing psychoacoustic experiments in order to determine how human participants respond to the changes in the stimuli. These experiments can determine whether a feature should be considered for further study as a potential aspect of timbre. For example Berger (1964) considered notes from a set of 10 musical instruments and studied how recognition was affected by removal of the rise and decay of the note, reversal of the note, and filtering out some of the higher frequencies. It was found that people were best at recognising unaltered notes, as one might expect. Reversed notes came in second, then notes with the rise and decay removed. Recognising the filtered notes was apparently the most difficult task.

2.1.1 Timbre Spaces

One of the major strategies for discovering the dimensions of timbre is by defining a 'timbre space'. A timbre space, as they are commonly known, is a multidimensional space where points in that space represent a particular timbre and the distances between points are perceptually based (Caclin, McAdams, Smith, and Winsberg, 2005). There are two techniques for defining these timbre spaces: Multidimensional Scaling (MDS) and Verbal Attribute Magnitude Estimation (VAME).

2.1.2 Multidimensional Scaling

Multidimensional Scaling refers to a family of techniques for representing the differences between stimuli in a dataset. MDS techniques in relation to timbre generate

a spatial representation of a user defined number of dimensions based on pairwise timbre similarity ratings obtained from study participants. The basic process for developing a timbre space using MDS is as follows. Start with a set of audio samples, typically recorded or synthesised instrument audio. Present every combination of two audio samples from the data set to each study participant and have them record how similar or not they perceive each pair of audio samples to be. Using these obtained measurements of similarity the MDS algorithm generates a space with the number of dimensions specified. The samples used to generate the similarity data have defined places within this space and the distances between these points is a measure of how distant the timbre of the samples are.

Now that there is a newly defined timbre space the question is what do the axes of this space represent? With the exception of the use of a variant of MDS called CONSCAL (Winsberg and Soete, Winsberg and Soete), the researcher has to attempt to determine the best acoustic correlation to these axes. Of course, it is unlikely that any correlation is going to be perfect given that timbre is a complex multidimensional attribute, but studying these timbre spaces can help researchers to ascertain which acoustic attributes might be most strongly linked to perception.

There has been much research published involving the use of MDS to generate timbre spaces but as timbre spaces specifically are not a focus of this thesis I will only provide a few examples to illustrate their use. Grey (1977) applied an MDS algorithm called INDSCAL introduced by Carroll and Chang (1970) to the similarity data obtained from 16 musical instrument tones. He obtained a 3-dimensional timbre space both to make the results easier to interpret and also as increasing the number of dimensions was not found to be beneficial. The first dimension was found to be correlated to the spectral energy distribution(SED), which is a measure of the spread of the frequencies present within the instrument tones. The second dimension was correlated with the attack and decay of upper harmonics, specifically how well the respective harmonic envelopes were aligned with each other. More simply, the dimension was a measure of the alignment of the amplitude change over time of the high harmonics. The final dimension was found to be related to the temporal pattern and frequency range of 'inharmonic energy'. This means the instruments were shown to exhibit components of varying frequency that were not part of the instruments pattern of overtones. These results are useful as they have isolated potential aspects of timbre that merit further research.

In a follow up paper Grey and Gordon (1978) repeated the experiment. This time

however, half of the set of instrument tones were paired up and had their SED was swapped. The resulting timbre space showed that each pair had swapped positions on the axis representing the SED. This validates the initial interpretation of the axis representing the spectral energy distribution of the instrument tones and further supports the fact that the SED is important to our perception of timbre. Additionally they found specifically that the mean of the spectral distribution was the most representative component and so used the term spectral centroid. Marozeau, de Cheveigné, McAdams, and Winsberg (2003) utilised MDS as a means of investigating the effect of fundamental frequency on timbre by performing a series of experiments where participants compared the timbre of 12 instruments. They treated the pitch of the instrument tones differently in each experiment. In the first they kept the pitch the same for each pair, in the second the pitch was different by a constant amount, and in the third the pitch difference varied. The timbre spaces they defined had 4 dimensions. The first dimension was a measure of 'impulsiveness', which is approximately a measure of how sustained a sound is, the second and third corresponded to the centroid and spread of the spectrum of the notes, and the fourth dimension in one experiment corresponded strongly with fundamental frequency. The correlation of the fundamental frequency was weak for the other experiments. Besides the strength of the fourth dimension, they found that the timbre spaces resulting from MDS of the similarity ratings in each experiment were very similar. This suggests that our perception of timbre is not affected by the pitch of the note. However they were working only with pitch differences within one octave.

There have also been non-MDS based approaches at determining the effect of pitch on timbre perception. Handel and Erickson (2001) and Steele and Williams (2006) both found that our ability to determine differences in timbre is affected when the difference in pitch between the two audio samples is larger than an octave. Steele and Williams (2006) found that the effect is weaker among musicians. Given the typically low dimensional nature of the timbre spaces that researchers generate with MDS techniques and the known multidimensional nature of timbre, some researchers have attempted to use variants of MDS to generate timbre spaces where each element in the data set has a position in the space but is also assigned a 'specificity' value. The specificity value is designed to capture elements of an instruments' timbre that are not well captured by one of the common dimensions and is a measure of how well the common dimensions can represent the timbre of the instrument; the higher the specificity the less well the instrument is described by the common dimensions.

The MDS variant was proposed by Winsberg and Carroll (1989) and first utilised by Krumhansl (1989). Krumhansl (1989) generated a timbre space of 3 dimensions which were found to correlate to temporal envelope, spectral envelope, and spectral flux. The temporal envelope is described in more detail as being "the rapidity of the attack". The spectral envelope is described as "brightness" which is a semantic term that has been shown to correlate with spectral centroid. This correlation between the term 'brightness' and the acoustic feature 'spectral centroid' is often attributed to Grey and Gordon (1978) but the term brightness does not appear in that paper. Schubert (2006) show that brightness and spectral centroid are well correlated but this is just one example. Spectral flux is described by Krumhansl (1989) as the way the spectral components change over time. In addition to these dimensions, the specificity values for each instrument are given. Krumhansl (1989) describes the interpretation of these values as "intuitive" but suggests that they require further study, adding that the findings support the idea that different timbres might have "unique characteristics".

This idea of extracting a measure of the unique aspects of an instrument's timbre was built upon by McAdams, Winsberg, Donnadieu, Soete, and Krimphoff (1995) using another MDS variant called CLASCAL developed by Winsberg and Soete (1993). Besides producing a timbre space and the positions of the timbres within it, this variant of MDS estimates latent classes within the timbre space as well as specificities and weights along each dimension for both the classes and specificities. Interestingly, McAdams *et al.* (1995) found that a timbre space of six dimensions without specificity values was most representative, but opted for a space with three dimensions and specificities due to the acoustic correlation being more sensible. However, the correlation between the space proposed by Krumhansl (1989) and their space of six dimensions was determined and they found that several of their dimensions correlated well with a single dimension of Krumhansl's space, and a single dimension of their space correlated with two of the dimensions of Krumhansl's space. As for the three dimensional model presented by McAdams *et al.* (1995), the first two dimensions correlated very strongly with the first two dimensions of the space presented by Krumhansl (1989), specifically attack time and spectral centroid. The third dimension of the space proposed by McAdams *et al.* (1995) did not correlate well with their interpretation of spectral flux from Krumhansl (1989) which they took to mean the variation of the spectral amplitude over time. Confusingly, they found that it correlated well with the description of spectral flux from a 1993 unpublished thesis by J.Krimphoff. This version of spectral flux was defined as "the average of the correlations between amplitude spectra in adja-

cent time windows: the smaller the degree of variation of the spectrum over time, the higher the correlation". They also attempted to informally account for the specificity values of the timbres by describing the timbres in plain language and suggested further research on the subject.

These are just a few examples of the use of multidimensional scaling techniques for timbre description. Others include Gregory (1994), Miller and Carterette (1975), Lakatos (2000), Elliott, Hamilton, and Theunissen (2013), Wessel (1979), and Iverson and Krumhansl (1993). The timbre spaces produced across the literature vary to some degree, which can likely be partially attributed to differences in the instrument tone data sets (Marozeau *et al.*, 2003). What is interesting is the common dimensions present in the literature such as spectral centroid and attack time. What often seems to happen is that the third dimension is related to the spectral attributes of timbre and it is difficult to find a strong acoustic correlate. Considering the couple of one to multiple mappings of the spectral dimensions between the six dimensional model presented by McAdams *et al.* (1995) and the three dimensional model presented by Krumhansl (1989), it seems to me that our perception of the spectral attributes of sound is more complex than can be represented with only three dimensions. The use of specificity values supports this opinion.

This idea highlights one of the issues with timbre spaces. Low dimensionality makes the spaces easier to interpret in terms of having clear acoustic correlations, but the low dimensionality is not sufficient to capture all of the nuance of the spectral attributes. So while timbre spaces are useful constructs for researching timbre, it would seem they are not a complete solution for determining the precise definition for timbre.

2.1.3 Verbal Attributes

Another approach to researching timbre is through the use of verbal attributes. Use of verbal attributes in timbre research dates back to Helmholtz and Ellis (1895) where they classified tones as either simple or musical and described the tones with adjectives such as bright, rich, and sweet, and accompanied these labels with descriptions of the harmonic make-up of the sounds they accompany. In modern research based on verbal attributes, study participants are presented with a set of sounds and asked to rate them on 'semantic scales'. These scales are based on words one might use to describe the timbre of a given sound such bright, dull, or sharp. There are two methods for utilising semantic scales: semantic differentials and verbal attribute magnitude estimation (VAME) with the latter more prominent. An important application of semantic differ-

entials to timbre was done by von Bismarck (1974). The semantic differential method involves semantic scales with "pairs of opposite attributes"(von Bismarck, 1974) at each end, for example, smooth at one end and rough at the other. In the study 30 such scales were used for a set of 35 sounds and factor analysis was used to determine the scales that were most representative of the set of sounds. Factor analysis is similar to MDS in the sense that the goal is to reduce a set of data points to an underlying set of variables or dimensions. von Bismarck (1974) found a four factor representation of the scale ratings with the dominant factor represented by the scale between dull and sharp.

Kendall and Carterette (1993) reproduced the work of von Bismarck (1974) on pairs of simultaneous wind instrument recordings and found that the same method and verbal attributes used in the previous study failed to distinguish the timbres they used. This was attributed to a number of things including the use of semantic differentials and, interestingly, the list of adjectives might have held different meanings to different subjects due to different mother tongues and the resulting translation. In their second experiment they used VAME, which differs from semantic differential in that the semantic scales are not made of pairs of adjectives but of a single adjective and it's negation. For example, a scale might have 'rough' on one end and 'not rough' on the other. Kendall and Carterette (1993) found that the verbal "ratings clustered into two groups, complex, heavy, hard, loud, and sharp versus pure, compact, and dim", meaning that the labels in a given set are associated with perceptually similar timbres.

Similar work is done by Zacharakis and Pastiadis looking into the difference different languages have on verbal attribute ratings (Zacharakis, Pastiadis, and Reiss, 2014, 2015). They also attempt to define a model for timbre that combines perceptual, verbal, and acoustic information (Zacharakis and Pastiadis, 2016).

2.2 Music Information Retrieval and Instrument Classification

Where the field of psychoacoustics is concerned with determining the underlying principles of timbre perception, music information retrieval (MIR) is oriented towards achieving success in particular tasks.

Such tasks include classifying musical genre, predicting the perceived 'mood' of the music, identifying which instruments are present in a given musical passage, annotating music with relevant information or identifying the artist that created the music (Fu,

Lu, Ting, and Zhang, 2011). For each of these tasks timbre is theoretically relevant but to varying degrees. Timbre is perhaps most relevant to the task of instrument classification. As timbre is essentially defined as the difference we perceive between different sound sources it is natural to assume that instrument classifiers work by determining the timbre present in a given audio file and comparing it to some standardised description of what the timbre of known instruments are. However, as we do not have a precise definition for timbre we do not know what information we should extract in order to represent a given timbre, and so instrument classification cannot work in that way.

Despite the lack of understanding of timbre and its elements there is still a need to classify instruments from audio recordings and identify genres. It is necessary to develop methods that solve these problems without trying to determine the precise definition for the timbre(s) present in the music. As part of my research was focussed on instrument classification for a time the following discussion will be primarily concerned with instrument classification. However, the concepts of feature sets and classifiers, which I will explain shortly, are used throughout music information retrieval.

2.2.1 Machine Learning

Given the task based nature of MIR and the lack of a precise definition for timbre there is widespread use of machine learning techniques in the field. The process of using machine learning for MIR tasks and specifically for instrument classification consists of several steps.

The first step is deciding what information to extract from the audio signals you wish to classify, this is sometimes referred to as the representation of audio signals (Siedenburg *et al.*, 2016). The pieces of information extracted from audio signals, as discussed previously, are called features and there are many examples (Vinet, 2002; Peeters, Giordano, Susini, Misdariis, and McAdams, 2011). The features are developed and utilised only for their ability to classify different sound sources; the perceptual relevance of a given feature is not relevant to the task, which is one of the key differences between psychology and MIR. One example of such a feature is the zero crossing rate of a signal, which is a measure of how often the signal crosses the x-axis. It is particularly useful when used to classify percussive sounds (Gouyon, Pachet, and Delerue, 2000). Another example are the Mel-Frequency Cepstral Coefficients (MFCCs), a set of numbers obtained through a sequence of mathematical transformations and are thought of as representing the spectral envelope of a sound at different levels of detail (Siedenburg

et al., 2016).

Once a set of features has been chosen and extracted the next step is often to refine that set of features (Fu *et al.*, 2011). This is necessary because the features that will provide the most predictive power are not known ahead of time and vary according to the particular data set, and also in order to reduce redundancy within the dimensionality of the feature set (Siedenburg *et al.*, 2016). There are different methods for reducing the feature set based whether the initial set was chosen manually or automatically but these are outside the scope of this thesis.

The final step is to select and train an automatic classifier. There are several different types of classifier including Gaussian mixture models (GMM), hidden Markov model (HMM), k-nearest neighbours (k-NN), support vector machines (SVM), and neural networks. The training process is different for each of these classifiers but basically involves presenting the elements of the data set to the model, and adjusting the parameters of the model based on the presented data.

These are all supervised classifiers, meaning that they are trained according to labelled input. When classifying instruments, researchers have a set of instruments that are of interest. The classifier is used to distinguish according to these predetermined classes. To train the classifier, all of the elements of the training set, each consisting of a list of features, are presented to the classifier along with their known class, in this case the instrument that produces the source audio. That known class is then used to inform the training of the model. A fully trained model when presented with a new audio signal will provide a prediction on which class the audio signal belongs to. This is the basic idea behind the machine learning approach used in MIR and there are many examples of such systems achieving high classification accuracy (Kaminskyj and Czaszejko, 2005; Benetos, Kotti, and Kotropoulos, 2006; Deng, Simmermacher, and Cranefield, 2008).

2.2.2 Problems in MIR

The various audio features are developed and used based on their 'predictive power', i.e. their ability to perform well in classification tasks. Because of this, and no doubt also due to the complex and multidimensional nature of timbre, a huge number of features have been developed. It has been found that features can be classified into separate classes that provide decent predictive power but within a given class were "highly collinear" (Siedenburg *et al.*, 2016; Peeters *et al.*, 2011). In my opinion there seems to be a connection here between these highly collinear classes and the concepts

of 'specificities' in MDS schemes and the varying contribution of spectral descriptors to the third axis in many MDS timbre spaces. So there are a large number of features, but there is significant overlap between features in terms of the information they represent.

Suppose we are developing a machine learning based instrument classification scheme. For the data set, we collect recordings from a number of musicians each playing a different instrument and train a classifier using this data. An important question to ask is 'how can we be sure that the machine learning approach we have taken is even classifying the thing that we think it is classifying?'. A strong result in terms of classification accuracy does not necessarily mean that the classifier is classifying according to the concept the researcher has imposed upon it (Sturm, 2013). For example, Sturm (2012) looked at two genre classification systems and found that small changes to the spectral aspects of the signals fed into the system produced different classification results. This implies that the classifiers are not actually classifying the genre of the signals.

2.3 A New Model

There is a distinct separation between the motivation behind and the techniques used in MIR and psychoacoustics. Where psychoacoustics aims at discovering the underlying principles of timbre perception, MIR is concerned with discovering ways to achieve better results for the tasks in the field. While timbre is important to both, only psychoacoustics explicitly seeks to further our knowledge of it. Siedenburg *et al.* (2016) provide a comprehensive overview of the differences between MIR and psychology. This raises the question: is it possible to seek knowledge of the underlying principles of timbre while still performing classification tasks?

To attempt to answer that question would entail developing a model for timbre that could be used to classify instruments, a perceptually-based model rather than one for which the features are chosen only according to how accurate the classification results are. Over the course of my research this is the path I have taken. Initially the goal was to start by detecting the envelope of a given sound, which is the way the signal rises and falls and has a connection to the perceived volume, and to follow that with some kind of spectral analysis. The study of envelopes was far more complex than first anticipated so a significant portion of this thesis is focussed on them. Through the study of envelopes the beginnings of a new model for timbre was discovered which is detailed in Chapter 4. Below is a survey of the literature on envelopes.

2.4 Envelopes

The envelope of a given signal is as a curve that varies slowly over time, and is representative of the change in amplitude of the signal over that time. Extracting this curve from a signal is not trivial and there are a number of different methods that seek to solve the problem. Some researchers use terminology that describes signals in terms of an envelope which varies slowly, and temporal fine structure that changes quickly (Moore, 2008). Another set of researchers describe signals as being comprised of the quickly moving 'carrier' component and the slowly moving 'modulator' component and refer to the process of envelope extract as 'demodulation' (Turner and Sahani, 2011).

Envelopes are useful for several purposes such as note onset detection (Caetano and Rodet, 2011), calculating the attack time of a note (Caetano and Rodet, 2011), speech recognition (Turner and Sahani, 2011), segmenting cardiac sound signals (Choi and Jiang, 2008), and they are used in cochlear implants (Wilson and Dorman, 2008). The importance of envelopes to this research is that they are known to have a significant impact on the perception of timbre, or, more precisely, the way the volume of a given audio signal rises and falls is known to affect timbre perception and envelopes are used to attempt to characterise this (Berger, 1964; Smith, Delgutte, and Oxenham, 2002; Zeng, Nie, Liu, Stickney, Del Rio, Kong, and Chen, 2004). Envelopes are studied in different fields and the problem of envelope extraction is ill-posed. Perhaps because of this, the curves extracted using different techniques differ in their criteria. For example, some algorithms have the curve follow the peaks of the waves very closely but do not allow the curve to intersect the wave. As I am interested in the way the envelope affects how we perceive timbre, I want a method for envelope extraction that gives a curve that most closely aligns with how we perceive the rise and fall of the overall volume of the signal. Additionally the envelope extraction method should be automatic. Envelope criteria are discussed in more detail in section 3.2. The remainder of this chapter relates to envelope extraction techniques from the literature.

2.4.1 Envelope Followers

Envelope followers are seen in musical equipment, such as guitar pedals, and are also present in music software such as digital audio workstations (DAWs). They are used in situations where some desired outcome is predicated on the change in amplitude of the signal over time. Essentially they work by generating a control voltage from the change in amplitude, that control voltage is then used to modulate effects.

It is difficult to find much information regarding the design of these envelope followers due to their proprietary nature. The only way to get a good sense of how accurate these followers are would be to sample different components and DAWs. And that is assuming that the follower is accessible directly rather than only being available as part of another component. This is outside the scope of this thesis.

2.4.2 Low Pass Filtering

A low pass filter removes the frequencies above a given 'cut-off' from a signal. Performing a pre-processing step such as half-wave or full-wave rectification, or squaring followed by low pass filtering is a method for extracting the envelope of a signal and is used by Caetano and Rodet (2011) for comparison against their own algorithm. Additionally, Wilson and Dorman (2008) discuss the use of envelopes obtained through low pass filtering being used in cochlear implants.

There are two problems with this envelope extraction method. Firstly there are many ways to design a low pass filter and secondly the researcher must choose a cut-off frequency. Both of these variables have an effect on the resulting envelope (Caetano and Rodet, 2011).

2.4.3 Root Mean Square

The root mean square (RMS) is, as its name implies, the square root of the mean of the squared sample values of a signal. Using this concept, but with a sliding window rather than the whole signal, gives an estimate of a signal's envelope (Caetano and Rodet, 2011). The problem with this method is how to determine the size of the sliding window. Filter size is explored in Chapter 3.

2.4.4 Hilbert Envelope / Analytic Signal

An analytic signal is a representation of a signal obtained via the Hilbert transform that allows one to derive functions representing the phase, frequency, and most importantly the amplitude envelope of the original signal (Loughlin and Tacer, 1996), (Vakman, 1996). The resulting envelope is referred to by different names in the literature such as, 'Hilbert Envelope', 'Amplitude of Analytic signal', and 'Magnitude of Analytic Signal', or simply 'Analytic signal'. The analytic signal is used often as an envelope extraction tool (Smith *et al.*, 2002), (Vakman, 1996), (Loughlin and Tacer, 1996). However, when considering musical tones the envelopes obtained from the analytic signal are

not representative of the slow rise and fall of a given wave. This is due to a number of problems, including: the resulting envelope containing pitch information when the wave is not a pure sinusoid and the envelope being affected by noise in the signal (Turner and Sahani, 2011).

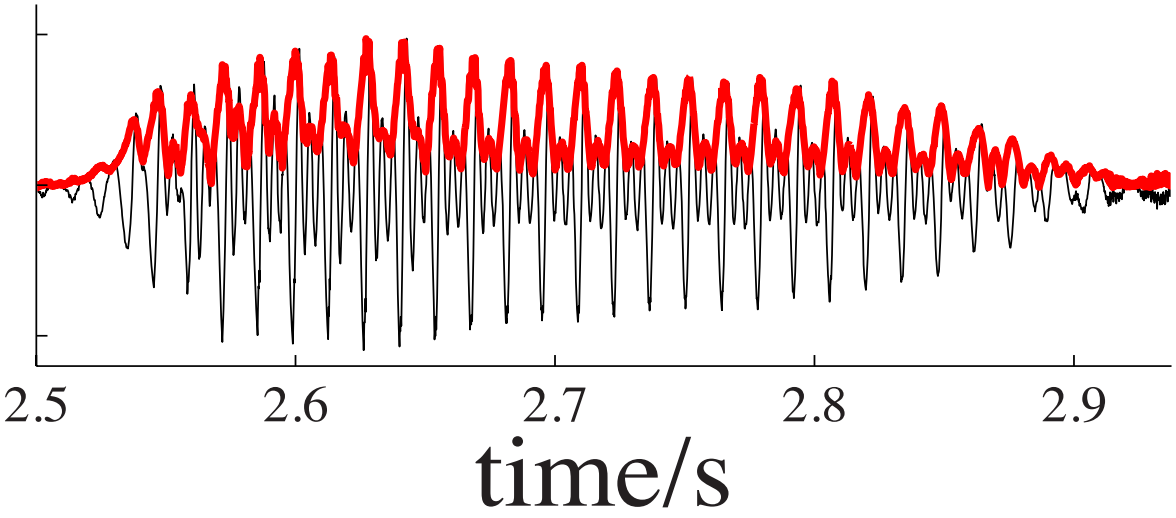


Figure 2.1: An example of the Amplitude of Analytic Signal extracted from Turner and Sahani (2007)

2.4.5 Probabilistic Amplitude Demodulation

This method for envelope extraction is performed with the view that a signal is composed of a rapidly varying carrier component and a slowly varying modulator component. The problem is viewed as 'amplitude demodulation' and the goal is to separate a signal into its carrier and modulator components. In a sequence of papers, Turner and Sahani (2007), Turner and Sahani (2010), Turner and Sahani (2011), Turner and Sahani describe a family of increasingly complex solutions to this problem under the banner of 'Probabilistic Amplitude Demodulation' (PAD). In essence, they use probabilistic models to determine the envelope and carrier pair that are most likely to produce the given signal.

It is difficult to compare their results to the results I report in Chapter 3 as their test data was speech and not isolated musical tones and in Turner and Sahani (2010) they state that their approach is "considerably slower than traditional feed-forward approaches to demodulation.". The advantages of my method are that it is very simple to implement and produces a result quickly.

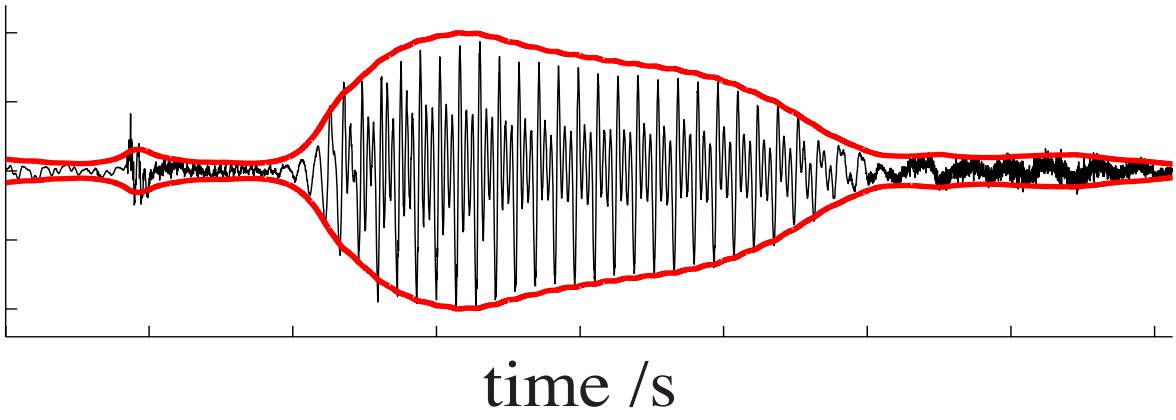


Figure 2.2: An example of PAD extracted from Turner and Sahani (2007)

2.4.6 Frequency Domain Linear Prediction

This method for envelope extraction was developed for speech processing and is based on a technique called 'Linear Prediction' which is used for spectral envelope modelling (Makhoul, 1975). Linear Prediction uses knowledge of previous inputs to and outputs of a signal in order to model a signal source. Where linear prediction is performed on a time domain signal to model spectral envelopes, 'Frequency Domain Linear Prediction' (FDLP) works on frames of the discrete cosine transform of a signal in order to provide an estimate of the time domain envelope of a signal, specifically the square of the signal's Hilbert Envelope (Athineos and Ellis, 2003). FDLP requires that the length of the window on which the algorithm operates be user defined, and also the polling rate, or number of data points per window the resulting envelope is comprised of. The fit of the resulting envelope is affected by these parameters, which makes the algorithm non-automatic and therefore not suitable for my use.

2.4.7 True Amplitude Envelope

True Amplitude Envelope (TAE) is a method proposed by Caetano and Rodet (2011) and similar to FDLP is adapted from a method for spectral envelope estimation. TAE is based on True Envelope Estimation, a method for estimating the spectal envelope of a signal. True Envelope Estimation is comprised of a number of steps. Firstly the 'real cepstrum' is calculated. The authors define this as the inverse Fourier transform of the log magnitude spectrum of a signal. The coefficients of the resulting signal are considered to be a measure of energy in different bands of the signal. The cepstrum is then

smoothed using a low pass filter to produce a smooth spectral envelope. True envelope uses an iterative method for finding the best envelope in terms of peak matching.

True Amplitude Envelope uses this process but applies it to a modified time domain signal in order to obtain an estimate of the time domain envelope. The time domain signal is rectified, which removes the negative components of the signal, then the signal is zero padded to the nearest power of two, a reversed copy of the original signal is added, and the signal is exponentiated. These steps are required so that the True Envelope Estimation algorithm will function correctly on a time domain signal.

Caetano and Rodet state: "Ideally, the amplitude envelope should be a curve that outlines the waveform, following its general shape without representing information about the harmonic structure". As our perception of change in volume is not exactly tied to the variation in amplitude of the wave and the goal of their method is to produce an envelope that outlines the waveform, this method would not be suitable.

2.4.8 Empirical Envelope Estimation Algorithm

The Empirical Envelope Estimation Algorithm (EEEA) (Meng *et al.*, 2013) is an envelope extraction technique based on the idea of 'prominent peaks'. These peaks are not well defined but can be thought of as maxima that should be included in the ideal envelope. The process outlined in the paper follows a sequence of steps: setting four initial parameters, adding noise to the signal, iteratively searching for maxima according to their algorithm, removing the noise, processing the end points, and interpolating between the obtained maxima for the envelope approximation. An example of EEEA is shown in figure 2.4.

The noise is added so that the envelope is not adversely affected in regions of the waveform where maxima are sparse, and then removed so they are not included in the obtained envelope. Similarly, the end points of the input signal are processed separately in order to avoid them negatively affecting the resulting envelope.

In relation to the cubic spline interpolation used in the paper, Meng *et al.* state "The overshoot and undershoot disadvantages of this interpolation method can be overcome by carefully choosing time scale parameter.". Having parameters that require manual adjustment disqualifies this method from being suitable for my use, but, in particular, having to fine tune the time scale to ensure the interpolation works correctly violates my criterion that the envelope extraction be automatic.

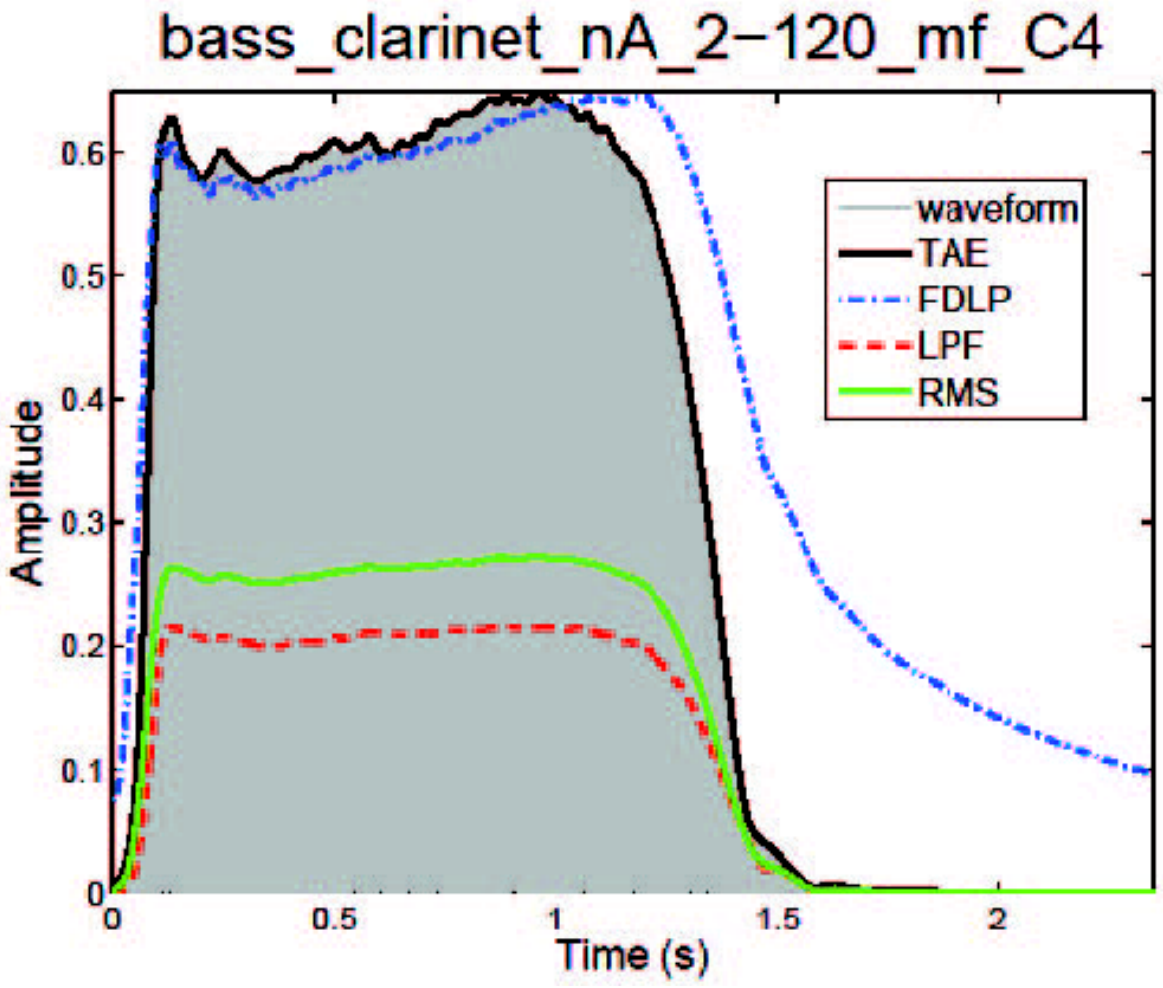


Figure 2.3: A plot containing examples of LPF, RMS, FDLP, and TAE from Caetano and Rodet (2011)

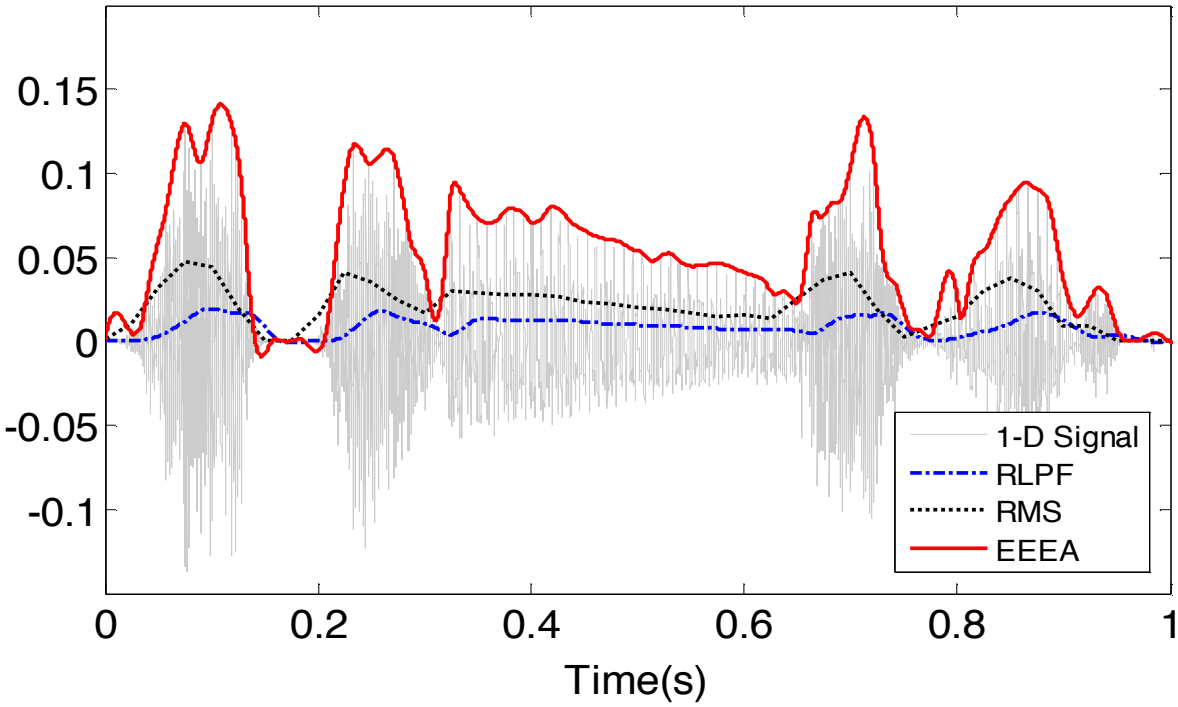


Figure 2.4: An example of EEEA extracted from Meng *et al.* (2013)

Chapter 3

Envelopes or Attack Decay Curves

3.1 Envelopes

The commonly held description of an envelope is a curve that represents the way the amplitude of a signal rises and falls. The importance of envelopes to this research is that they are known to affect the way we perceive an instrument's timbre. Envelopes are most often seen in musical synthesis in the form of the ADSR model. ADSR stands for attack, decay, sustain, and release, with each term representing a section of the overall envelope. The attack is the initial part of the note where the amplitude rises from nothing to its maximum value. After reaching maximum amplitude the signal decreases in strength slightly; this is the decay segment of the envelope. After decaying, the signal plateaus keeping a consistent or 'sustained' amplitude which typically is the longest part of the envelope in time. Finally the signal dies away to nothing and this is typically very quick compared with the sustained segment.

Use of the ADSR model is an important part of creating synthesised instrument sounds that mimic natural instrument sounds by rising and falling in volume in a similar way. Real sound waves lose energy over time and the structure of that energy gain and loss is approximated by the ADSR model where you can control the gradient and length of each segment, allowing a user to have control over the amplitude of a note over time.

One of the primary reasons for being interested in envelopes in relation to timbre analysis, besides using the overall shape or part of it as some kind of timbre descriptor, is that known aural phenomena occur at different times or with varying strength during a note. As an example, two such phenomena are transients and noise frequencies. Transients are short, high amplitude sounds that occur close to the onset of the note

or, alternatively, they are present in the attack of the note, and noise frequencies exist throughout the note and we hear these as the more percussive or non-harmonic elements in the sound, such as the sound of the bow against the strings of a violin. If we can detect the envelope of a given note then we can segment that envelope to isolate then use that information to refine our analysis.

As with most things in science the reality is more complex than the model. The rise and fall in the amplitude of real signals does not follow a sequence of straight lines as in the typical ADSR model. When attempting to detect an envelope in a real audio signal there are many difficulties that simply aren't present in synthesis. So there are a number of known techniques for detecting envelopes and these work to varying degrees. But a large part of the problem is that the question of envelope detection is ill-posed. There is a lack of agreement on a precise definition for an envelope. As a result of this lack of definition, the way I judged the results of a given algorithm developed over time.

Below I discuss some criteria that other researchers deem important for envelope extraction algorithms and further elucidate the problem of envelope extraction. Then experimental restrictions are discussed before the implementation and results of a number of my own algorithms are presented.

3.2 Attack Decay Curves

We know that envelope detection is an ill-posed problem. This is partly because different authors have different reasons for wanting to detect envelopes, some of which are discussed in Chapter 2. Possibly because of these different motivations the criteria that defines what an envelope is and what a detection algorithm should involve varies from paper to paper. One criterion for which there is agreement is that the envelope varies slowly over time, relative to the rapid vertical displacement seen in the main body of the wave. This content of the wave is sometimes defined in the literature as temporal fine structure or TFS (Moore, 2008). This criterion conveys the idea that the envelope does not dip whenever the wave does, that the envelope represents change over a longer period of time.

Another proposed criterion is the automatic adaptation to the signal (Turner and Sahani, 2011; Meng *et al.*, 2013) which I find itself to be ill defined. Any algorithm that attempts to detect the envelope of a signal must have some degree of adaptive ability. This criterion seems to be a little at odds with another criterion which is steerable

smoothness or user steerability (Meng *et al.*, 2013; Turner and Sahani, 2011). Some researchers believe that envelope detection algorithms should adapt automatically to a given signal but the user should maintain some control over the smoothness of the extracted envelope. This control is often done through the use of parameters and all of the methods described below have some parametrisation. It is my view that the use of too many separate parameters has an impact on whether or not you could claim your algorithm automatically adapts to the signal.

Another criterion for an envelope, that appears in several papers, is that the envelope should pass smoothly through the prominent peaks of the waveform. This raises the important question of how you define the prominent peaks and also how you isolate them once you have a precise definition. When viewing a waveform it is easy for a human to make judgements about which peaks would be considered prominent and therefore a human could easily annotate a waveform with a given envelope. But it can also be easy for a human to detect difference in timbre. Automatic detection requires a definition for what you are looking for and it appears that prominent peak labelling is also an ill-posed problem. Roebel and Rodet (2005) describe their notion of prominent peaks as follows, "For an harmonic signal the prominent spectral peaks are generally the harmonics, however, if some of the harmonics are missing or weak (clarinet) the spectral envelope should not pass through these". Meng *et al.* (2013) make no attempt to define what a prominent peak is other than to cite Roebel and Rodet.

As my interest in detecting envelopes is based upon the fact the rise and fall in volume of a signal is known to affect our perception of timbre, a curve through all the prominent peaks is not suitable. Consider figure 3.1 which shows a plot of formula 3.1 on the left and a plot of formula 3.2 on the right.

$$cos(x) + 0.25cos(2x) \tag{3.1}$$

.

$$-(cos(x) + 0.25cos(2x)) \tag{3.2}$$

.

If we were to use an algorithm based on prominent peak detection on each of these signals, the resulting envelopes would be different. And if interpreting the envelope as a description of the rise and fall in volume of the signal then the results would tell us that the volume of each of these signals is different. However, this is not the case as these signals are the same wave, except one is flipped upside down, and therefore

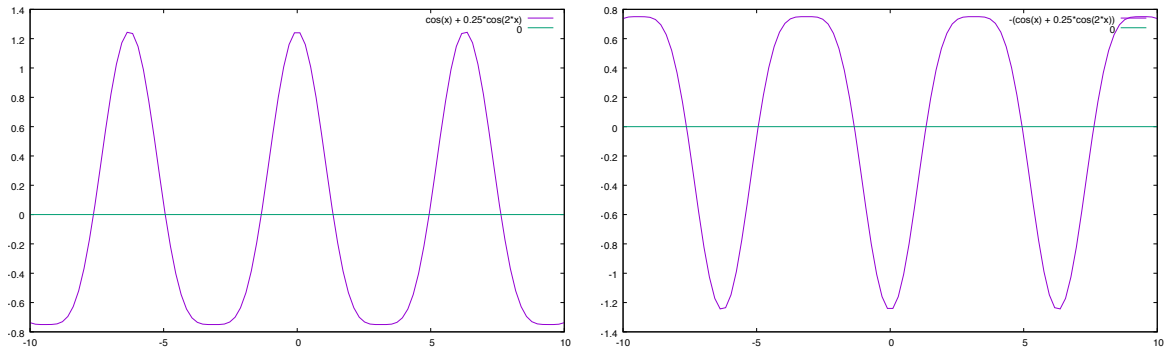


Figure 3.1: A mix of two cos waves presented upright and flipped upside down

they have the same volume. So any envelope detection algorithm that is based on the concept of prominent peaks or attempts to 'outline' the waveform, cannot be used to measure the rise and fall of volume over time.

Related to the idea of prominent peaks is the idea of wave intersection. Many of the techniques used for envelope extraction produce a curve that goes through some of the peaks of the wave and 'sits on top' of the wave, i.e. the envelope curve doesn't intersect the wave at any point. While a number of envelope extraction techniques produce envelopes that do not intersect the wave this is not necessarily a requirement.

To distinguish my goal from those most prevalent in the literature I think the term 'attack-decay curve' is a more accurate label. While the concept of an envelope is ill-posed and lacks an exact physical meaning, the ideal attack-decay curve is a curve that characterises the rise and fall of the waveform as they relate to change in perceived volume. It is important to note that my concept of the attack-decay curve is distinct from the terms used in the ADSR model.

3.3 Experimental Restrictions and Background

3.3.1 Isolated Tone

For each of my envelope detection methods the contents of the waveform supplied was restricted to a single isolated musical note recorded from a real acoustic instrument with a sample rate of 44.1KHz. This is because it is the most simple case to work with. Notes played consecutively when played fast enough will 'bleed' into each other meaning that that for each note the full variation in amplitude from the onset of the note to the end of its decay is not demonstrated. So to avoid that complication the

input to each of the detection algorithms is a file containing a recording of a single note.

3.3.2 Single Voice

All of my research is concerned only with monophonic music; music consisting only of a single voice. The reason for this is that the waveform produced by recording multiple instruments simultaneously is vastly more complicated than the waveform produced by a single instrument. When viewing the visual representation of a waveform of a single instrument a human can easily see the way the amplitude rises and falls, in most cases, approximately in accordance with the rise and fall in volume of the recording. When presented with a similar display of a polyphonic recording, you cannot correlate the graph with the sound of a single instrument within the mix.

In order to apply envelope or attack decay curve detection to a recording with multiple instruments, the first requirement would be to isolate how each instrument has contributed to the overall shape of the wave. This could possibly be achieved through segmentation based on Fourier analysis where you filter for approximate harmonic series but as instruments have overlapping frequency ranges this would be difficult or perhaps impossible, it is hard to say. In essence the task of detecting envelopes for timbre classification is already complicated, and so detecting envelopes in polyphonic music is outside of the scope of our research.

3.3.3 Preprocessing and Filtering

In sound files each sample value represents a deviation from average air pressure at the location of the detector. As sound waves cause variation in air pressure the detector records these variations as numbers greater or less than zero. So each recorded signal is comprised of both positive and negative elements. Because signals most commonly are somewhat symmetrical this causes complications when processing them. A typical process applied to signals, musical or otherwise, is filtering. Filtering involves the removal of some aspect of the signal which most commonly means removing unwanted frequencies. One example of why this is done is as a means of removing noise from a signal. The application of filtering to this research is the use of smoothing filters and the use of these involves a process called ‘convolution’, which, in general terms, uses one function to modify another. Given a function, in this case our musical waveform, and a filter function, the filter is applied to the waveform as a sliding window. In

each iteration the corresponding values of the waveform and the filter function are multiplied, summed, and then divided by the number of values in the filter window. The resulting value of one of these iterations is the value of the convolution that is aligned with the middle of the filter. In simple terms the filter function scales the values of the waveform within the window and then averages them to give a single result. When this process is performed for the entire waveform, a modified version of that waveform is produced. Because the values of the modified waveform are the average over a section of the waveform, the result is a smoothing of the original wave. Different filter functions produce different results and examples of this process appear in section 3.4.3.

Because a raw waveform contains both positive and negative values a smoothing filter applied to the raw signal will not produce a useful result as these components cancel each other out. In order to make the result meaningful, we need to perform a preprocessing step to remove the negative elements and for this task there are a few options, some of which are described below.

Firstly you can square the wave. This is achieved simply by multiplying each sample by itself. As each sample obviously has the same sign as itself you end up with a signal that has no negative components. This has the side effect of changing the frequencies present in the wave and also the overall shape of the wave. The change in shape is caused by two things, firstly that samples that were previously in the negative part of the waveform are now in the positive part, this is also the cause of the change of frequencies in the wave. Secondly, by squaring the samples the wave is essentially scaled non-uniformly.

An alternative would be to operate on a signal that has been half-wave rectified. A half-wave rectified signal is achieved by setting every sample less than zero to zero. This also affects the harmonic content of the wave but does not change the overall shape of the positive part of the wave.

A similar option is to perform full-wave rectification which essentially means taking the absolute value of each sample. As with squaring the wave, and half-wave rectification, full-wave rectification affects the harmonic content of the wave. Each of these options achieves the desired result of removing the negative components of the wave to facilitate filtering. However, the resulting attack-decay curve can differ depending on which pre-processing option is chosen. Besides variation in the shape of the resulting curve, the most prominent difference is seen when operating on the squared or full-wave rectified wave. In the final algorithm presented in this chapter the frequency increase

caused by these steps can, under the right circumstances, result in a 'wobble' in the calculated attack-decay curve that is not present when using half-wave rectification. In comparison, instead of the wobble, the curve obtained from the half-wave rectified wave can have plateaus due to the segments of the wave with zero values. These differences only occur under specific conditions, but they serve as an example to illustrate that the combination of a particular pre-processing step with and a particular extraction algorithm can give particular results. Therefore the pre-processing step is decided case by case.

3.3.4 Upper and Lower Envelopes

While waves are typically close to symmetrical this is not always the case and in some cases, a waveform can be significantly more positive than negative or vice versa. This asymmetry will exist in recorded instrument sounds only for a short time. This is because the waveform of a recording is representing the sound pressure, which has to average zero by definition, as it is the deviation from the average pressure at the location of the detector. In cases where the wave is asymmetrical, the resulting envelope might not be particularly representative of the signal. The typical process for envelope extraction is a preprocessing step, as described above, followed by the particular algorithm being used. However, these algorithms are typically only concerned with the 'upper envelope' meaning an envelope obtained from the positive elements of the wave. Of course this distinction only makes sense when the preprocessing step is half-wave rectification as in that case there is a clear separation between the positive and negative elements of the wave. To clarify, half-wave rectification could be performed by taking only the positive or only the negative elements of the signal, but in squaring or performing full-wave rectification of the wave there can only be one result. So when presented with an asymmetrical wave one envelope could be obtained if you square the wave but it is possible to obtain two separate envelopes, both an upper and lower envelope, if half-wave rectification is used. This is dependent on the algorithm being used to detect the envelope and some modification could be required.

When considering that characterising the wave is the goal it might well be useful to obtain both the upper and lower envelopes. This is because if the waveform exhibits significant asymmetry, then the obtained envelopes will provide different characterisation from one another, which in turn serves to distinguish a given waveform to a finer degree of accuracy. In this chapter I present three detection methods. For the first two I present only the results of the upper envelope with the knowledge that computing

the lower envelope as well is a trivial exercise. For the final method, the preprocessing step is full-wave rectification and so there is only one possible result.

3.3.5 Monophonic Recording

Audio files can come in many formats. All of the files used in following experiments are of the WAVE (.wav) format. WAVE is a popular file format used on all major operating systems for uncompressed audio. The audio being uncompressed is hugely important in musical signal processing as some compression algorithms alter the content of the stored waveforms in order to decrease their file sizes. While some types of compression can compress the audio in a way that doesn't result in an obvious perceivable difference, the underlying data is still changed and therefore not suitable for analysis. An exception is lossless compression formats that allow the complete reconstruction of the original signal, but this is an extra step in computation for no added benefit as file size was never a concern during this research.

Within each audio file, and depending on the specifics of the file format, there can be multiple separate 'channels'. Some files will only have a single channel these files are 'monophonic' or 'mono' for short. A file with two or more channels is 'stereophonic' or 'stereo'. The use of multiple channels is primarily for playback using speaker systems of various configurations. If using a speaker system with two speakers to play an audio file that has two channels then each speaker can be used to play only a single channel. From the perspective of the listener this allows for, among other things, clearer separation of different voices within the music and a sense of the location of those voices. These aspects make the recording more accurate compared with experiencing the live performance.

The manner in which a sound is recorded can make a large difference to the resulting audio file and these differences can in turn impact the analysis of the audio. For example, recording a musical performance with two microphones results in an audio file with two separate channels. Each channel is a separate waveform and yet both waveforms are recordings of the same event. So what are the differences between the two, what causes these difference and how does it impact the analysis of the recording? The waveforms seen in figure 3.2 are from a stereo recording of a segment of a cello performance. The top waveform is the recording from one microphone and the bottom waveform is from the other. At first glance it is difficult to see the how these two waveforms are related to each other. While some of the peaks in each waveform seem to line up with each other horizontally, meaning they take place at the same

point in time, the heights of these peaks are different and the overall shape of the wave is distinct between the two waveforms. Differences in wave shape in this case means that there could be differences in both amplitude and the harmonic content of the wave for the same moment in time. These differences exist despite the fact that both microphones are recording the same event. They exist due to the placement of the microphones relative to the cello. The overall volume of a given channel is dependent on how far away the instrument is from the microphone. Of course, the closer something is the louder it is perceived and so will have a greater effect on the detector in the microphone, resulting in a higher amplitude signal overall. The distance between the microphone and the instrument also has an effect on the wave shape due to the attenuation of frequencies. The sound wave produced by the cello consists of various frequencies including the fundamental frequency, the overtones, and noise frequencies. These frequencies attenuate differently with distance and the result is that the distance from the cello will affect the perceived timbre and therefore the shape of the recorded wave. The direction of the microphones relative to the cello also affects the recorded signal.

In terms of our analysis, having two waveforms for the same event poses a problem. If there are two waveforms, then there are two envelopes and for the reasons discussed above they will likely be different. But the differences are due to the recording process rather than something inherent to the instrument or the technique of the individual playing the instrument and so the differences are irrelevant. But there are still two separate waveforms. To solve this problem there are two possible solutions. I could either take only one channel from the file and ignore the other, or I could mix both channels together to create a single mono audio file. In the first case the question is how could I be sure that the chosen channel best represented the real audio? And in the second case the question is how representative of the real audio is the mixed audio? The mixed audio could introduce a degree of error into the envelope detection process. Determining the answers to these questions is outside the scope of this thesis and so for all experiments only mono audio is used.

3.3.6 Energy Supply

It is important to discuss the fact that some instruments can produce more complex envelopes or attack-decay curves due to the control the musician has over the instrument. For example, for a single note played on a guitar or piano the musician cannot continuously supply and control the energy provided to the instrument due to the in-

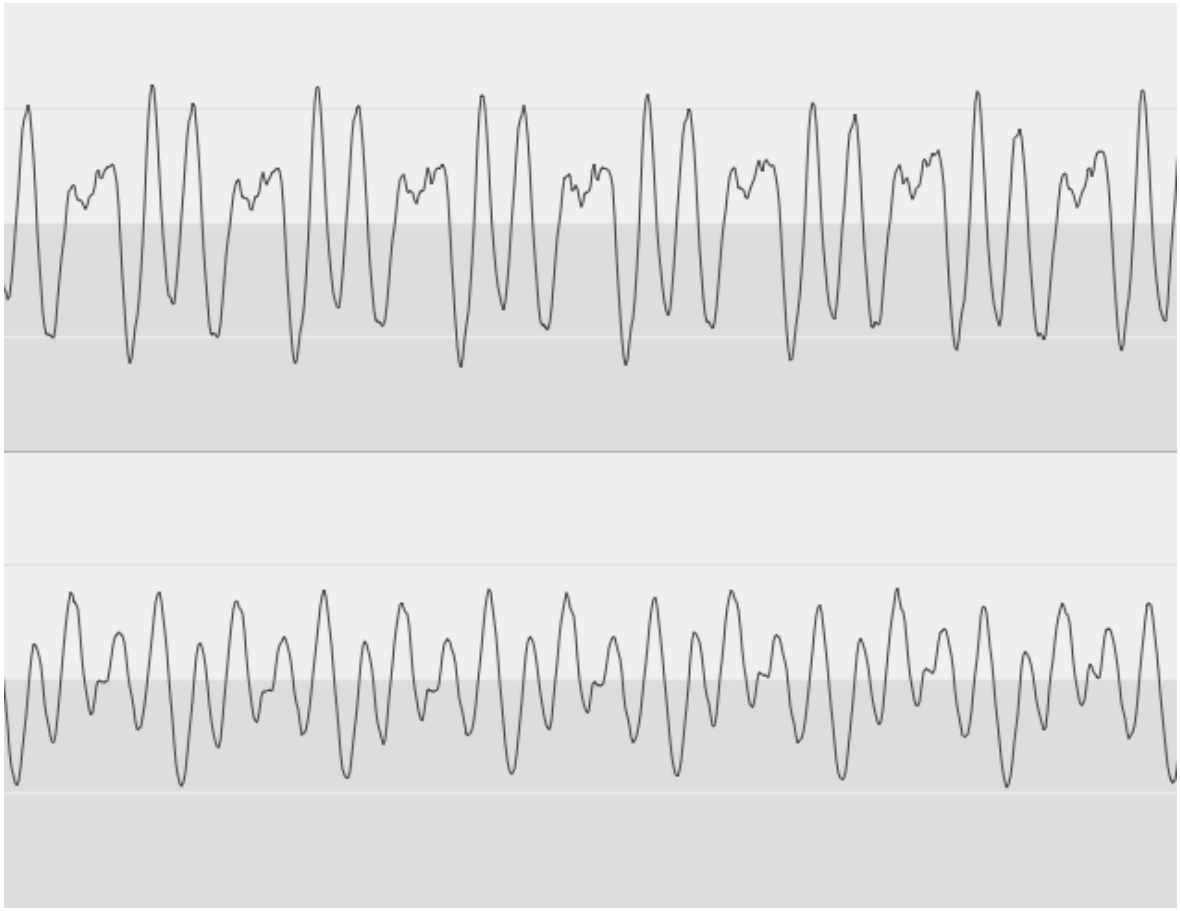


Figure 3.2: The waveforms of a stereo recording of a cello

strument's interface and mechanisms for sound generation. For these instruments, once a string has been plucked, or a key has been pressed, there is no way to increase the amount of energy to that note. In the case of bowed or wind instruments the musician can increase or decrease the energy supplied to the instrument over the course of what would be interpreted as a single note, by varying their breath or bowing technique.

3.4 Detection Methods

The first two methods described below served as intermediate steps for envelope extraction on the way to the final algorithm that I am presenting. I believe it useful in this instance to provide details on the intermediate steps for two reasons. Firstly because envelopes are ill-defined, the path my research would take was not clear from the outset. This process, in my opinion, is worth documenting as it helps to convey the difficulty of the problem itself and of research into timbre in general. Secondly, record-

ing intermediate steps might expedite the progress of future researchers by limiting the time spent on incomplete solutions.

The algorithm I developed for extracting the attack-decay curve is not restricted by the idea of prominent peaks or not intersecting the wave. The goal is simply to detect the curve that describes the overall behaviour of the wave in a way that is more closely correlated with our perception of volume. A commonly held idea is that the envelope should be a smooth curve and this is important for an attack-decay curve too.

3.4.1 Line Segment Algorithm

The Line Segment Algorithm is a geometry based algorithm which works by linearly interpolating between a set of points from the input waveform. These points are chosen by projecting lines of a given maximum length from a starting point to subsequent samples in the wave, with the restriction that the lines cannot cross the wave except at the endpoints. When a projected line does not cross the wave then the point through which it was projected is added to the set of points for linear interpolation. This restriction on crossing makes the resulting curve less likely to dip beneath the peaks of the wave.

Implementation

The algorithm takes a waveform and a maximum line length as input. The first step is to calculate the gradient between the first sample and the next sample. A line with the calculated gradient and with the given length is then projected from the starting point. At each sample spacing along the projected line the value of each sample is compared to the height of the line at that point. If at any point the value of the sample is larger than the height of the line at that point then the line is intersecting the wave and so this line segment is rejected. If that happens the next step is to calculate a new gradient using the starting point and the next sample along from the previously used sample. This process repeats until a non-intersecting line is found, in which case the index of the point through which the line was projected is saved into an array and this point is used as the new starting point. Or, if the maximum line length is reached and no non-intersecting line was found, then the point equal to the starting point plus the maximum line length is saved into the array and used as the new starting point, although the longer the line length the less often this occurs. Once the end of the waveform has been reached the array of saved indices is used to plot a

curve using linear interpolation. In my experimentation I used the maximum value of the waveform as the starting point and then did a forwards and backwards pass from that point. A single pass is described formally in algorithm 1.

A high level understanding of the algorithm can be gained by thinking about the conditions in which a projected line would intersect the wave. Consider the algorithm with a starting point on the left side of a peak in the wave. Subsequent samples are higher in value than the starting point and so the resulting line is projected upwards away from the wave resulting in no intersection and a new point added to the array for later interpolation. This continues with successive points being added to the array as the algorithm approaches the local maximum. The closer the algorithm gets to the local maximum the less steep the projected lines become until the local maximum is reached and subsequent lines are projected towards the wave. When this happens, provided the maximum line length is long enough relative to the waveform, the projected lines will continue intersecting the wave until the line is projected through the next peak where there is unlikely to be an intersection. In this way the algorithm repeatedly finds local maxima or points close to them. Over all this results in a curve that will climb to the peaks of the wave but not dip too far below them. An example of this behaviour is shown in figure 3.3

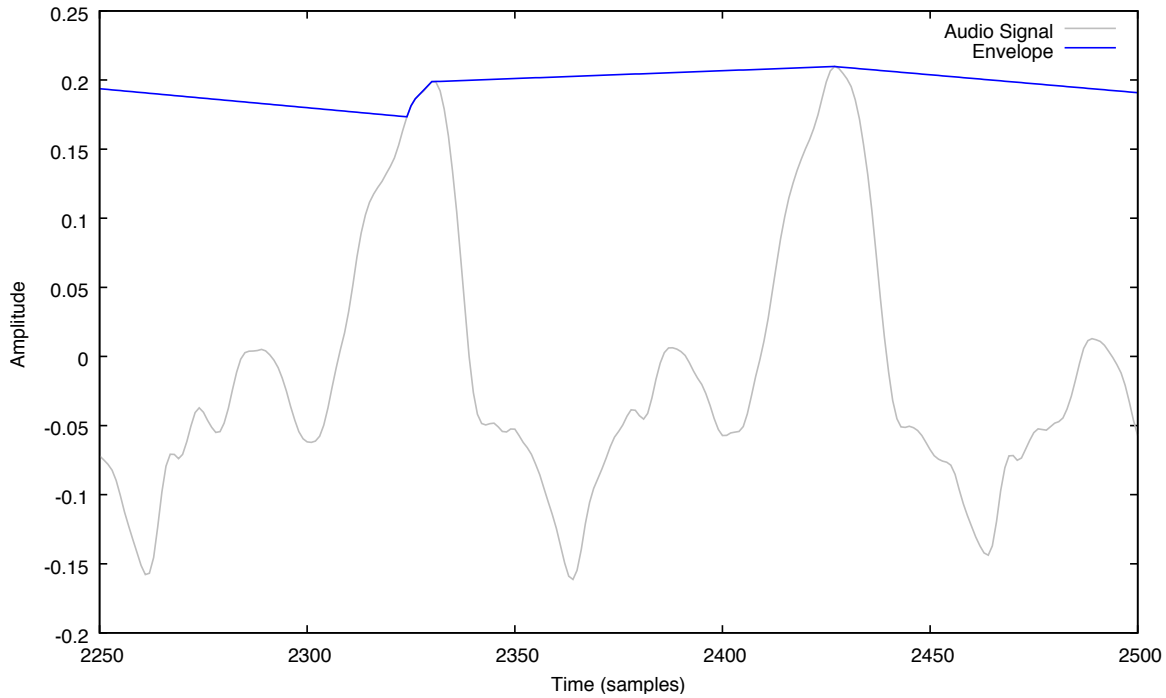


Figure 3.3: The line segment envelope curve as it climbs a peak

Algorithm 1 Line Segment Algorithm

```
1: procedure SinglePass(AudioBuffer ab, LineLength lineL, StartingPoint index)
2:     [ ] indices
3:     while index < length of ab - lineL do
4:         bool intersect
5:         for k ← index + 1 to index + lineL do
6:             grad ← gradient between ab[k] and ab[index]
7:             intersect ← false
8:             for l ← k + 1 to l ← index + lineL do
9:                 if line with gradient grad from index to l intersects wave  then
10:                    intersect ← true
11:                    break
12:            if !intersect then
13:                index ← k
14:                indices ← k
15:                break
16:        if intersect then
17:            index ← index + lineL
18:            indices ← index
```

Results and Discussion

The line segment algorithm is more of an algorithm for envelope extraction as opposed to attack-decay curve extraction. The reasoning behind that statement is that the algorithm works by drawing line segments that rest on top of the wave. The result of this behaviour is that, depending on the chosen line length, the extracted curve will approximately follow the peaks of the wave. As previously discussed, a change in the amplitude of the wave is not necessarily perceived as a change in volume. Therefore the line segment algorithm is likely incapable of adequately characterising change in volume.

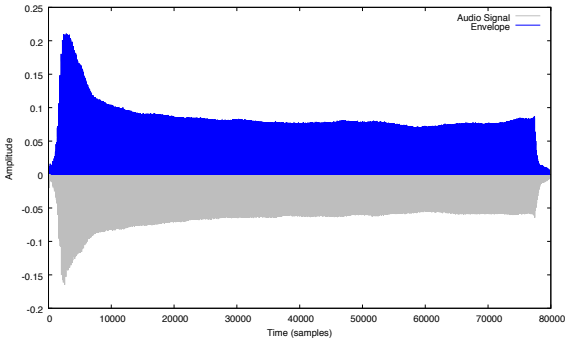
However, the algorithm as an envelope extraction algorithm performs quite well. Here I present a number of figures representing different instruments and different maximum line lengths. Figure 3.4 shows the result of applying the line segment algorithm to an a440 alto saxophone recording with maximum line length set to 50, 100, and then 500 samples. It can be difficult to determine differences between the extracted envelopes when viewing the whole wave so for each line length setting I have included a view of the first 10,000 samples.

As the line length increases the resulting envelope becomes less jagged. This is due to the longer projected lines being more likely to cross the wave after the algorithm has ascended a peak and therefore making the algorithm more likely to find a local maxima. Determining the maximum line length that will give the best envelope is an unsolved problem. Theoretically, the ideal line length is equal to the distance between adjacent peaks which might be approximated using pitch information, but as musical instruments do not produce repeating waveforms, that distance is not constant anyway. It seems likely that the the use of Fourier analysis could lead to automatic line length adjustment but I did not investigate this.

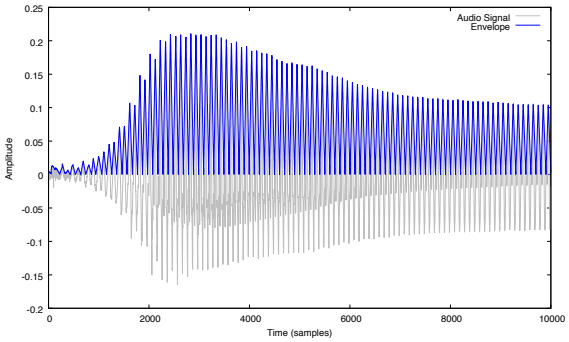
Figure 3.5 shows the results of applying the algorithm with a maximum line length of 500 to a guitar note at A220 and a violin note at A440. It seems that a line length of 500 produces good results with a variety of instruments but more testing is necessary.

3.4.2 Exponential Curve Fitting

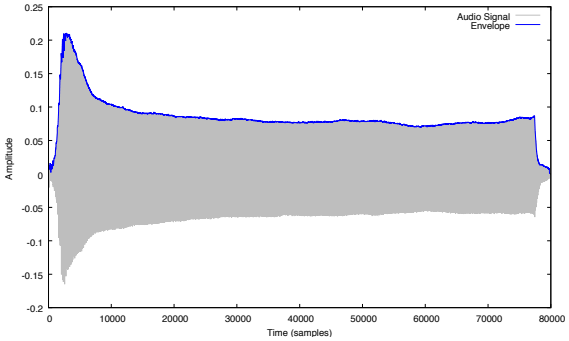
The goal of this detection method was to fit a pair of exponential curves to a given waveform. This method was aimed at capturing the attack decay curve rather than the typical idea of an envelope. With that fact in mind I attempted to fit one exponential for the attack and one for the decay of the signal. In this case the decay can be thought



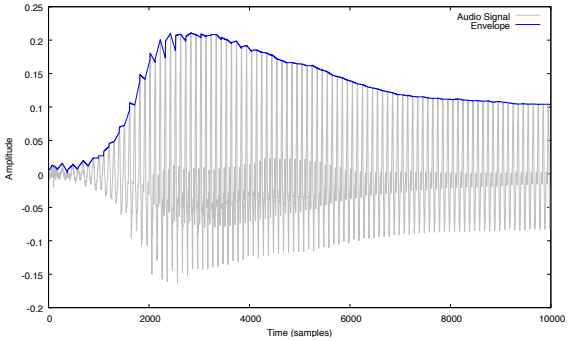
(a) Line Length 50—Full Wave



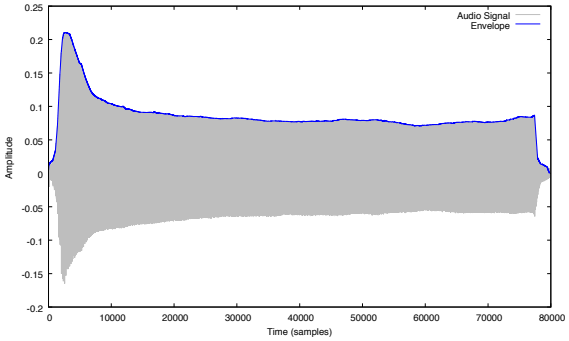
(b) First 10,000 Samples



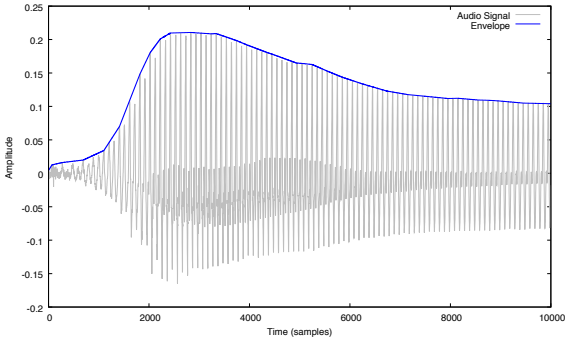
(c) Line Length 100—Full Wave



(d) First 10,000 Samples



(e) Line Length 500—Full Wave



(f) First 10,000 Samples

Figure 3.4: The results of different line lengths on an alto saxophone waveform

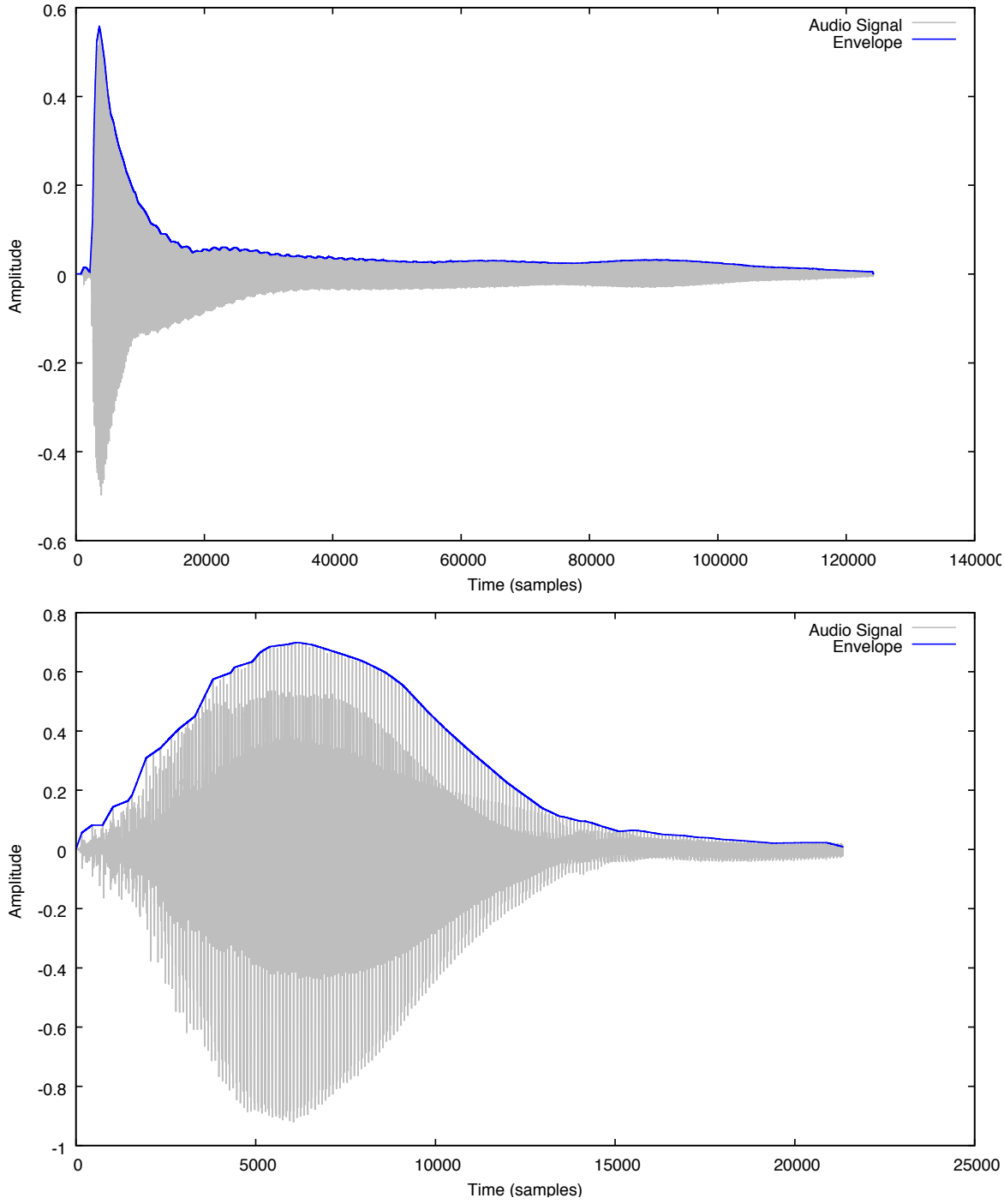


Figure 3.5: A guitar and a violin waveform and envelope with line length = 500

of as the remainder of the signal once the peak amplitude has been reached or, more simply, the rest of the signal after the attack. The idea behind this detection method was attempting to approximate the gain and loss of energy in the signal. The vibration of some musical instruments, which is responsible for producing the sounds that we hear, is known to reduce in intensity, or decay, in an approximately exponential way (Giordano, Gould, and Tobochnik, 1998). As these vibrations are responsible for the production of the sound we hear, the vibration is also responsible for the waveform that is produced when the sound is recorded. Therefore as the vibration in the instrument decays in an exponential fashion, so too does the amplitude of the recorded waveform and we should be able to fit an exponential curve to the decay of the wave. The physical justification for attack being modelled with an exponential is less clear but it appears to work at least as well as the decay of the wave.

It is important to realise how significantly the instrument on which a note is recorded affects the resulting curve. Some instruments such as wind or bowed instruments allow the musician to continuously supply energy to the instrument. While other instruments such as guitar or piano only allow the musician to supply energy at the start of the note. In the case of an isolated note on a guitar or a piano, the attack and decay are can be approximately modelled with exponentials but on wind or bowed instruments that is not necessarily the case. Because the musician has control over the energy supplied to the instrument, when playing a wind or bowed instrument they are able to sustain the note as they wish between the attack and decay, with the amplitude of the resulting waveform accordingly being a function of the musician's choice.

Broadly speaking this algorithm works by locating the maximum value in the waveform and fitting an exponential in each direction from that point. This assumes that there is no sustained portion between the attack and decay and, therefore, would not be expected to perform well on instruments where the sound can be sustained by the musician and the results reflect this. In the context of this thesis attempting to model this simple case seemed an appropriate place to start.

Implementation

The first step in the process is to normalise the input audio file, i.e. divide the sample values in the wave by the maximum value of the waveform. This is important because this detection method compares the processed input audio to exponential functions of the form

$$\exp(kt)$$

where t is time. At $t = 0$ these functions evaluate to 1 and by normalising the input audio the maximum value of the audio is also 1. Making the input audio the same scale as the exponential functions makes comparing the two straightforward. Following this the audio needs to be half-wave rectified in order to remove the negative components from the wave.

The next step is to create an approximation to the envelope by linearly interpolating between the maxima of the waveform. Initially this was done simply by manually choosing a window size and finding the maximum sample in each window. However, obtaining a decent approximation to the envelope with this method is very dependent on the window size matching the audio in some sense. As there is no way to know what the best window size would be ahead of time I devised an algorithm that is less reliant on manual intervention.

Firstly, traverse the samples of the normalised waveform and find all the maxima. This is done by checking each sample compared to its neighbouring samples; if a sample has a higher value than the previous and following samples then it is a local maximum and the index of the sample in the array should be stored. Once the array of maxima has been populated the next step is to remove maxima that will cause the resulting envelope to be erratic. In this case by 'erratic' I mean an envelope that dips too far below the prominent peaks of the waveform, the envelope should be close to the peaks of the wave and not change value significantly over a short time.

Removal of the maxima that will cause a less representative envelope is important and is performed by using a sliding window. The first step is to traverse the array of maxima. Starting at the beginning of the array calculate the average value from the values of the maxima in the current window. Then, iterate over the same window again and find the maxima with values less than the average and store them in an array. Repeat this process each time sliding the window along by 1 sample until the end of the array of maxima is reached. As the windows are overlapping it is important to be careful that duplicates are not added to the list of maxima with values less than the average. Additionally, once the array of maxima with values less than the average is populated, it is important to be aware that they might not be in order due to varying averages across different windows.

As maxima are most commonly the prominent peaks removing maxima that have values less than the average of a given window is a good way to remove the maxima that appear within the wave. The size of the sliding window was determined empirically and set to 10, but changing this value does not have a large affect on the resulting

envelope. As long as the value is small enough so that peaks from very different areas of the waveform are not included in any given averaging operation then the resulting envelope should be adequate for this purpose. This process is formally defined in algorithm 2.

Algorithm 2 Exponential Curve Fitting - Envelope Approximation

```
1:  procedure MAXIMA PROCESSING(AudioBuffer ab)
2:      windowSize ← 10
3:      [] indices                          ▷ Vector of the indices of maxima in ab
4:      [] removedIndices                   ▷ Vector of the indices of rejected maxima in ab
5:      for each value in sample in ab do
6:          if value is a maximum then
7:              indices ← value
8:      for each window of size windowSize do
9:          average ← average of the maxima in the window
10:         for each maxima in the window do
11:             if maxima < average and maxima is not already present then
12:                 removedIndices ← maxima
13:     Remove each maxima in removedIndices from indices
```

Remove the problematic maxima from the array and then form the envelope by linearly interpolating between the remaining maxima. For a given pair of maxima the linear interpolation is performed by calculating the distance in samples between the location of each maximum in the original audio. Then calculating the distance in sample value between the maxima and dividing that by the number of samples between the maxima gives the increment for each element of the envelope between the two maxima. Repeat this process for each pair of maxima; recording the incremented values between each pair. The process discussed so far produces an envelope like that seen in figure 3.6. The grey is the half rectified waveform of an A440 played on an alto saxophone, and the blue line that goes through the peaks of the wave is the obtained envelope.

Once the maxima based envelope has been obtained the next step is to fit a pair of exponential functions using the maxima based envelope as the basis for comparison. One exponential needs to be fitted for the attack of the audio and one for the decay. The process of fitting the exponential functions to the envelope is a two step process. The first step is to calculate the first candidate exponential function according to an

Algorithm 3 Exponential Curve Fitting

```
1: procedure CALCULATE EXPONENTIAL(Samples s, Coefficient k)
2:     range ← 2
3:     [ ] exponential
4:     incr ← range/s
5:     count ← 0
6:     for i ← 1 − s to 0 do
7:         t ← (incr ∗ i)
8:         exponential [count] ← exp (k ∗ t)
9:         count ← count + 1
10:    return exponential
11: procedure CALCULATE ENVELOPE(MaximaEnvelope maxEnv)
12:    error ← MaxDouble
13:    prevError ← 0
14:    k ← 1000                ▷ Unrealistically high value chosen as starting point
15:    topK ← 0
16:    bottomK ← 0
17:    nextMoveUp ← true
18:    [ ] function
19:    while difference between error and prevError > errorThreshold do
20:        prevError ← error
21:        error ← 0
22:        function ← CalculateExponential(length of maxEnv, k)
23:        for each sample i in maxEnv do
24:            error ← error+ abs(function [i] − maxEnv [i])
25:        if ! (error < prevError) then
26:            nextMoveUp ←!nextMoveUp
27:        if nextMoveUp then
28:            bottomK ← k
29:            k ← k/2
30:        else
31:            topK ← k
32:            k ← topK + ((bottomK − topK) /2)
33:    return function
```

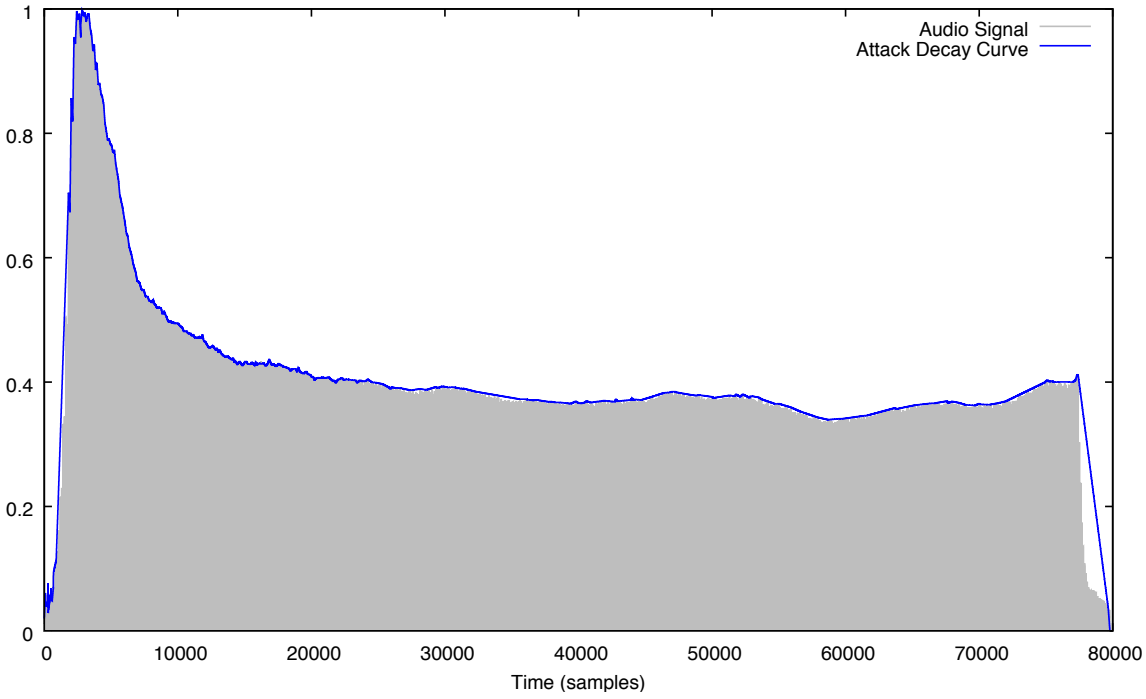


Figure 3.6: The maxima envelope of an A440 alto saxophone note

initial maximum value of k, a range for sampling the function, and the number of values to sample.

The initial maximum value of k is an upper limit and defines how sharply the function curves upwards. At $k = 0$ the function is a horizontal line but as k increases the exponential curves upwards more sharply. It is most sensible to start at a higher value of k and come down as the observed attack and decay are typically less steep than $\exp(kt)$ with high values for k.

The range for sampling the function in my implementation is a floating point number that defines how far below 0 the function is sampled. I have set the range in my implementation to 2 which was an empirically based decision as the function between $t = -2$ and $t = 0$ for a range of values for k provides shapes that are subjectively approximate matches to some observed attack and decay rates in my test data. A more robust solution would be to determine the sampling range dynamically but as I did not pursue this algorithm further I did not investigate this.

The number of values to sample is the number of values in either the attack or decay of the maxima envelope, depending on which part the algorithm is fitting.

The second part of fitting the exponential functions is to search the parameter space to find a good value for k. For each corresponding pair of indices between the

sampled exponential function and the maxima envelope section the absolute difference is computed and then added to a total. This total is a measure of error as the closer the exponential is to matching the maxima envelope the lower the total will be. The error measure is saved and then compared to the previous error measure in order to facilitate a binary search for a value of k that approaches the closest match between the maxima envelope and the exponential function. If the error is less than the previous error then the value of k used to calculate the exponential function is halved but the previous value of k is kept. On the next iteration if the error resulting from the new value of k is not less than the previous error then the new k value is calculated to be halfway between the previous two k values. This process repeats, but at the beginning of each iteration the current error is compared to the previous error. If the difference in errors is below a chosen threshold then the exponential is unlikely to get closer to the maxima envelope and the process is complete. In my implementation the error threshold was 0.001 but making this value smaller didn't noticeably improve the fit of the exponential function. Fitting an exponential curve is described more formally in algorithm 3.

After fitting an exponential for both the attack and decay these must be combined to form a single attack-decay curve. Depending on the details of the implementation it is important to reverse the decay to ensure that the curve follows the audio waveform. The remaining problem is that the goal of this process was to find the attack-decay curve for the input audio but the curve was fitted to the maxima envelope that itself was obtained from the half rectified input audio so the attack-decay curve must be further processed to make it match the original audio as closely as possible. The solution is to normalise the attack-decay curve by dividing each element by the maximum value in the curve. Then multiply each element by the maximum positive value of the original input audio. This brings the attack-decay curve to the same scale as the input audio.

Results and Discussion

It is important to discuss the maxima envelope detection process and the motivation for including it as part of the attack-decay curve extraction. When developing the idea of fitting a pair of exponential curves to a waveform it was obvious that there needed to be some way to measure how good a particular fit was in order to automate the process. The most straightforward idea is to do an element-wise comparison between the half-wave rectified audio and the candidate exponential. However, this does not work as shape of the half-wave rectified audio waveform is nothing like an ideal attack-decay

curve. The result of this comparison is shown in figure 3.7.

The issue is that many of the values in the audio are much smaller than the same position in the ideal attack-decay curve and particularly in the case of half wave rectification approximately half of the values are 0. So the error calculation makes the resulting attack-decay curve have many values very close to 0.

If we had an envelope based on prominent peaks for comparison, then at least it would prevent the attack-decay curve from being mostly flat and therefore useless. This was the motivation behind the development of the maxima envelope detection scheme. But this leaves the problem of how to determine which peaks are prominent. The method detailed above for removing peaks is to calculate a local average and remove the peaks that are less than that average. This is useful in that it removes most of the maxima that are within the wave, i.e. the maxima that are below the level of the majority of the peaks. Without peak removal the maxima envelope might look something like figure 3.8. Figure 3.9 provides a closer view.

While useful, this method for removing peaks is not a robust solution and will sometimes fail to remove peaks that would not be considered prominent or remove peaks that are. Figure 3.10 is an example of this. Part of the attack of the envelope sits above peaks that would likely be considered prominent, and I suspect this is because the large valued peaks at the top of the attack were included in the average calculation for these excluded peaks. A possible solution to this problem is to consider each peak for removal only once and to have the average calculated in an area centred around the peak being considered for removal. This might prevent the peaks at the top of the attack from causing the removal of other prominent peaks in the attack. It is also the case that the removal criterion of a peak having a lower value than the local average might be improved upon. However, the maxima envelopes obtained with the current method supported the production of an attack-decay curve that allowed me to judge how viable the exponential fitting algorithm was, and so the maxima envelope algorithm was sufficient for its purpose.

The best results from the exponential fitting algorithm were achieved when the input was a guitar note at A220. Consider figure 3.11. The attack of the resulting curve cuts across the wave slightly but provides a decent approximation. The decay curve is approximately the right shape to match the decay of the waveform and this serves to partially validate the choice of using exponentials to model the attack-decay curve. However, this particular exponential is not steep enough and the reason for this is the way the candidate exponentials are evaluated.

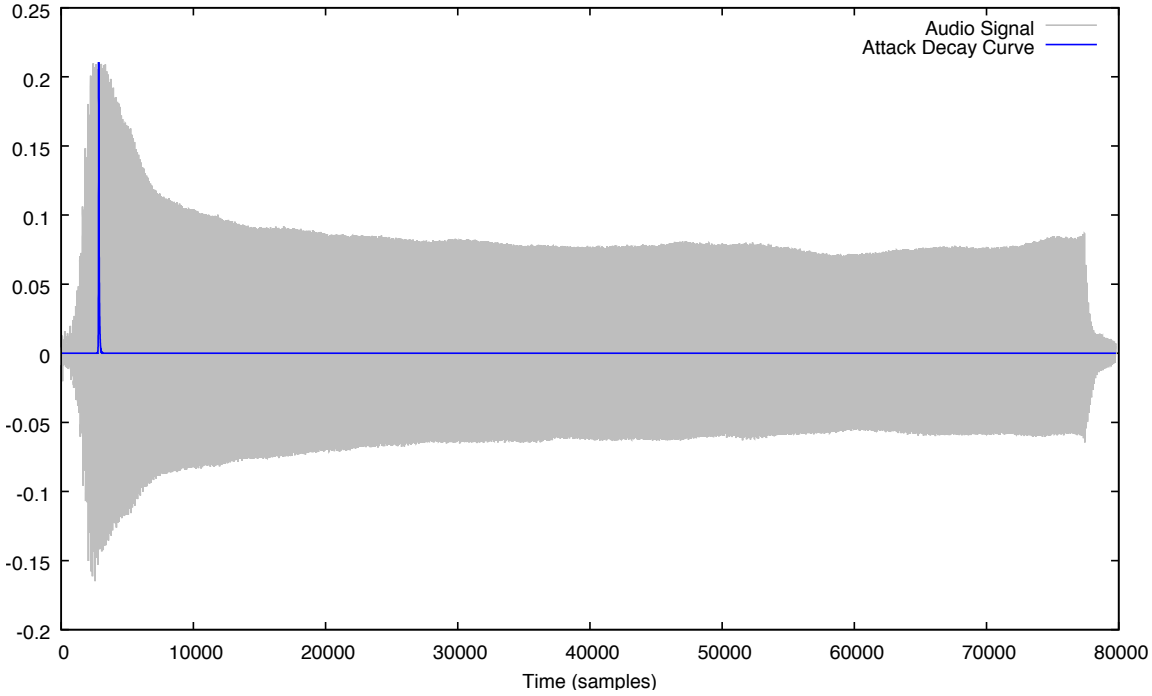


Figure 3.7: The result of comparing the exponential to the half-wave rectified wave

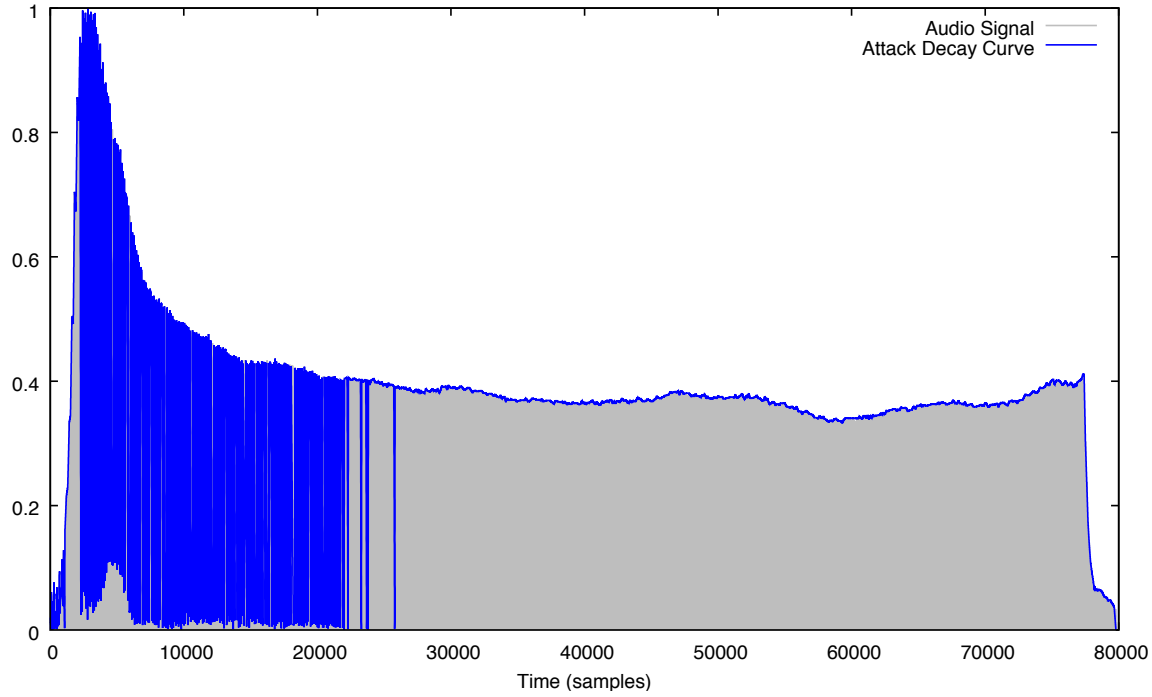


Figure 3.8: The maxima envelope without peak removal

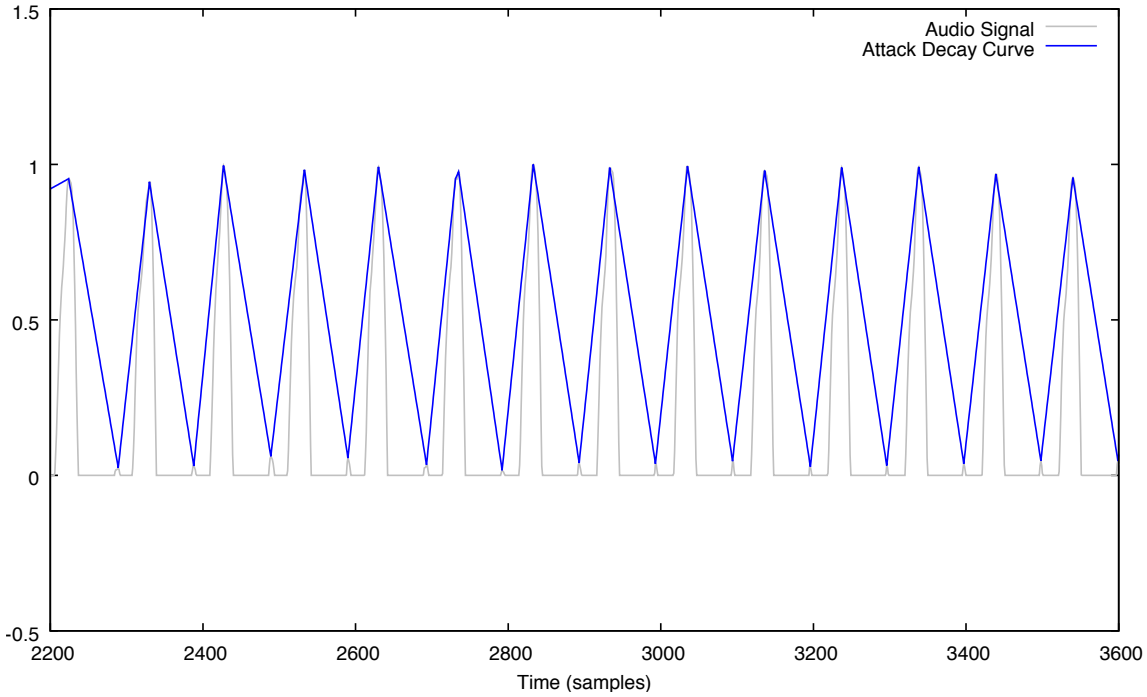


Figure 3.9: A close up of the maxima envelope without peak removal

In the current implementation of the algorithm every point on the curve is compared to the corresponding point in the maxima envelope. The guitar waveform exhibits a steady state after the initial decrease in amplitude. The values in that steady state part of the maxima envelope influence the resulting exponential as much as the values in the initial decay causing the exponential to be a worse fit for the initial decay. What this tells us is that the model in its current form is inadequate even for instruments that do not allow the musician to continually supply energy. A similar result is shown in figure 3.12. A model that could separate the steady state part of the wave from the attack and decay would likely produce better results. Figure 3.13 shows the expected result of applying the algorithm to a saxophone note.

For instruments that do not allow the performer to sustain the sound and with dynamic calculation of sampling range and separate processing for the steady state portion of the waveforms, then the exponential fitting algorithm might provide an adequate characterisation of rise and fall in volume of a given wave. However, as that would still be an incomplete solution further investigation into the exponential fitting algorithm is not warranted.

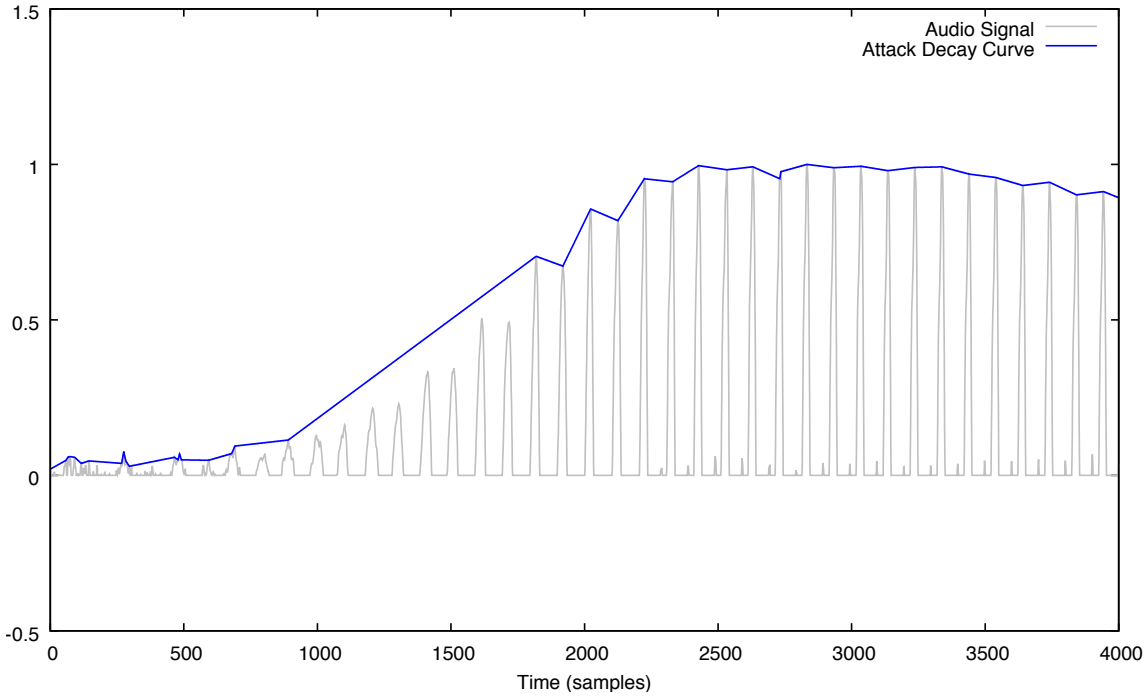


Figure 3.10: The start of a maxima envelope that misses some prominent peaks

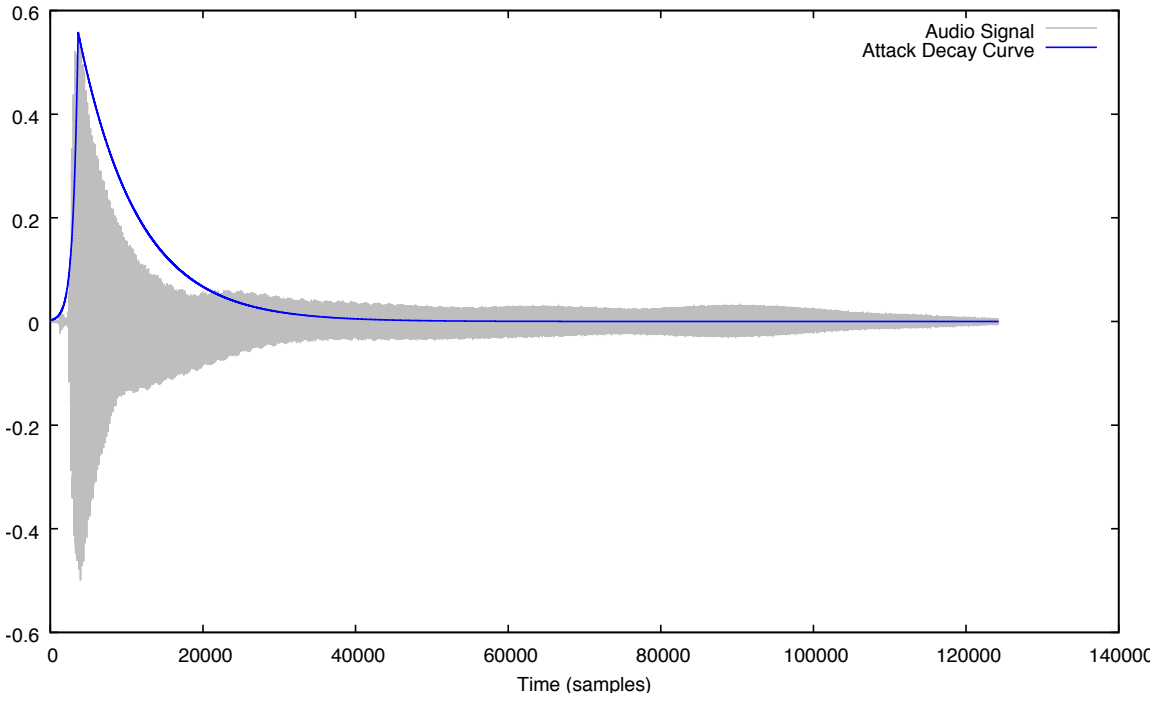


Figure 3.11: The exponential fitting algorithm applied to a guitar note

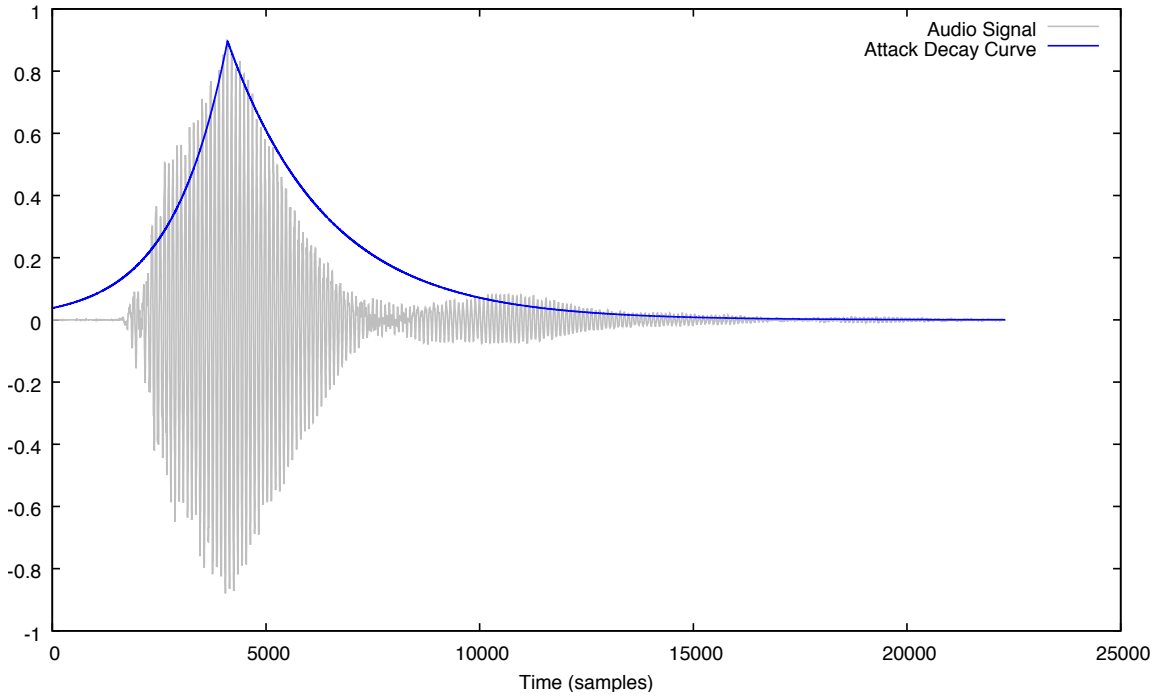


Figure 3.12: The exponential fitting algorithm applied to a non-sustained piano note

3.4.3 Pitch Based Convolution

This detection method produces an attack-decay curve by convolving smoothing filters with the given waveform. The concept of smoothing filters are introduced in section 3.3.3 but basically the process results in a curve that can be thought of as the average amplitude over time, but the values that are averaged are scaled by the given filter function.

The problem with using a smoothing filter on a musical signal is the question of filter size. How do you determine the ideal number of samples that should be included for each averaging operation? The filter needs to be wide enough to actually smooth the wave but narrow enough to capture a fast onset. The attack of a note in particular is known to be important to our perception of timbre so we want to minimise corrupting it when performing smoothing.

Initially I experimented with smoothing using an empirical approach. I experimented with different filter sizes and examined which of these produced a better envelope but I did not have a method for automatically determining the filter size. I moved on to other methods, until I had the idea that incorporating knowledge of musical pitch might prove useful.

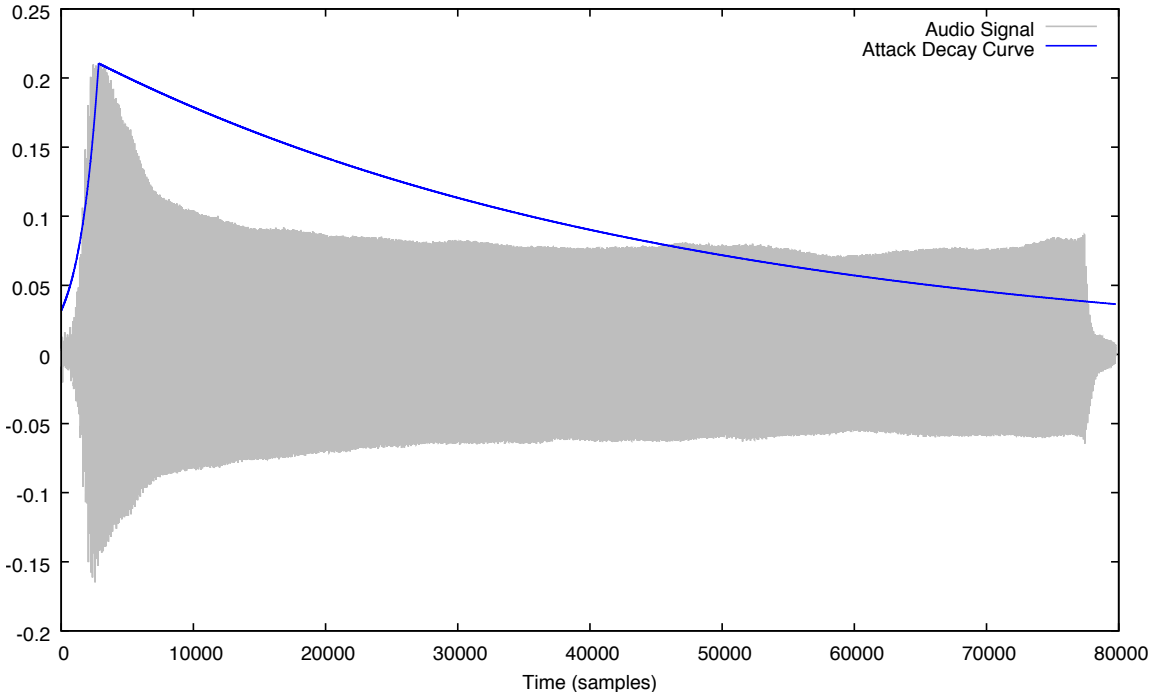


Figure 3.13: The exponential fitting algorithm applied to an alto saxophone note

There is a relationship between pitch and sample rate that allows for the calculation of a window size that captures a whole number of cycles of the waveform. Within each cycle each frequency component of the waveform is only present once and this is a crucial fact in understanding why a window the size of a whole number of cycles is important. It is important because the distribution of energy in a single cycle of waveform is not uniform. To illustrate, think of a window size that captures one and half cycles centred on a position in the wave where it captures two peaks representing the fundamental frequency, and remember that the fundamental frequency typically has the highest amplitude or largest amount of energy in the signal. If we shift this window half a cycle it is focused on an area very close to its original position but now the window is only capturing one of these peaks. So we end up with two window locations very close to each other but having a significantly different average amplitude. This significant difference in recorded average amplitude is not reflected in the way we perceive a rise in volume and so the window width of one and half cycles is clearly not fit for purpose. By having a filter size be a whole number of cycles we avoid this problem.

Another important concept is the specific functions used for the filter as this has

a significant effect on the resulting attack-decay curve. I implemented three filter functions, box, triangle, and a Gaussian-like function. A box filter has every value equal to one. When a box filter is applied to a window of a signal each value in the window contributes with equal importance to the resulting value as the values are unscaled. A box filter is the same as taking the average of the values. A triangle filter is typically defined as in formula 3.3.

$$Tri(t) = \begin{cases} 1-|t| & \text{if } |t| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The Gaussian-like filter I use is seen in formula 3.4

$$(1 - t^2)^3 \tag{3.4}$$

where t represents time. A plot of this function is shown in figure 3.14. Obviously we do not want to scale our sample values by negative amounts so we truncate the function at $t = 1$ and $t = -1$.

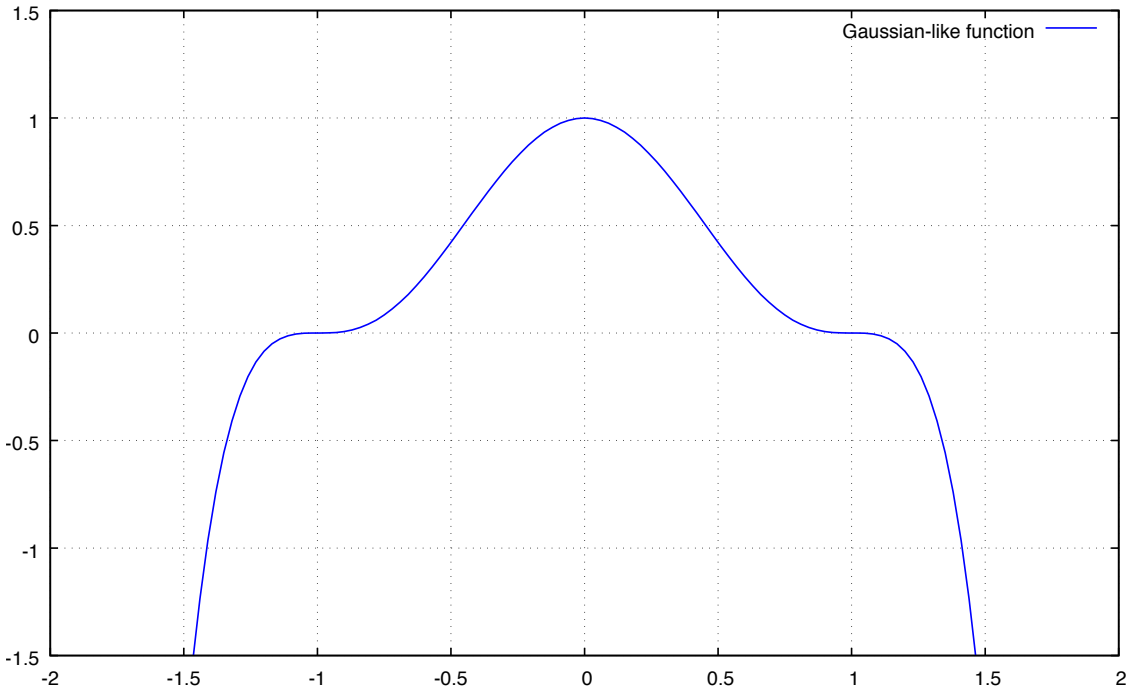


Figure 3.14: The Gaussian-like function

Implementation

Start with an audio file, the sample rate of that audio file, and the pitch of the note contained within. The number of samples that make up a whole cycle is determined by dividing the sample rate by the pitch of the note. For example, with a sample rate of 44.1KHz and a pitch of 440Hz the resulting cycle size is approximately 100 samples. The filter size is then some user defined multiple of the cycle size rounded to the nearest odd number. The reason that the filter size should be odd is because convolving a filter with a window of the signal produces a single value that that is aligned with the centre of the filter. If the filter has an even size then this value would be out of alignment with the audio signal by a half sample, which likely would not cause a problem but an odd filter size is standard. Given that the cycle size is determined according to the pitch this rounding of the filter size could introduce some error.

As the pitch increases the number of samples that represent a single cycle decreases, so with a high pitch note and a low sample rate the difference of 1 sample could be significant. The answer to this is to use a higher sampling rate. In my research I have used a sample rate of 44.1KHz but there are multiple standard higher sampling rates. The higher the sampling rate used the less significant a single sample is to the algorithm. However, I have not quantified this effect exactly.

The next step is to calculate the values for the chosen filter function. The box filter is simple as it is the same as averaging the values within the filter window, so there are no filter values to compute. To calculate the values for the triangle and Gaussian-like filters the respective functions must be uniformly sampled. Both functions are symmetrical so we need only compute half of the values but in my implementation I store the full filter as a matter of convenience.

Before the filter is used the negative elements of the audio must be processed. As the goal is to extract a curve that characterises the rise and fall of volume in a signal and the volume is a function of the entire waveform and not just the positive elements, the only sensible choice for this is full-wave rectification. Next, convolve the audio signal with the filter to get the the attack-decay curve. Typically the result of convolution should be a signal that is the size of the original plus the size of the filter. This is because convolution of the filter and a single window gives a value for the centre of the filter. Imagine the filter window as it begins to slide over the waveform, as the first value is covered by the filter the centre of the filter is half the filter width before the start of the waveform. So the result is a half filter width of values either side of the original signal.

As the half filter width either side of the original signal is perhaps not that useful anyway I start the convolution a half filter width into the waveform, i.e., the left most value of the filter is aligned with the first sample of the wave. I then pad the front of the attack-decay curve with a number of zeroes equal to the filter size. This was an empirically-based decision and results in an attack-decay curve that is more closely aligned with the waveform. A formal description of how a filter is applied to the waveform is given in algorithm 4.

Algorithm 4 Pitch Based Convolution

```
1:  procedure APPLY FILTER(AudioBuffer ab, WindowSize windSize, Filter filter)
2:      halfWindow ← windSize/2
3:      [] curve
4:      for i ← 0 to windSize do
5:          curve ← 0
6:      for i ← halfWindow to length of ab − halfWindow − 1 do
7:          sum ← 0
8:          filterIndex ← 0
9:          for j ← i to i + windSize do
10:             sum ← sum + ab[j] ∗ filter[filterIndex]
11:             filterIndex ← filterIndex + 1
12:         curve ← sum/windSize
13:     return curve
```

At this point there is a curve. But when compared to the waveform of the original audio it is difficult to tell how well the curve is representing the rise and fall of the wave. This is due to the obtained attack-decay curve appearing at a different position on the vertical axis when compared to the peaks of the original wave. This is because the values of the attack-decay curve are averages of scaled segments of the wave. As the sample values of the wave vary between 0 and the values of the peaks in the wave it follows that the average will not be near the peaks of the wave. The vertical displacement between the attack-decay curve and the peaks of the wave make it difficult to judge how representative the attack-decay curve is of the rise and fall of the audio. A human can easily follow how a waveform rises and falls by observing the peaks in the wave, and so by approximately vertically aligning the attack-decay curve with the waveform of the original audio it is easier to judge how well the curve represents the rise and fall of the wave.

To align the obtained attack-decay curve more closely with the peaks of the wave, allowing for easier comparison, is a simple process. Firstly normalise the attack-decay curve. This means dividing each element in the curve by the maximum value of the attack-decay curve so that each value is between 0 and 1. This is useful as each value can then be thought of as a ratio of the maximum value. The next step is find the maximum value in the original audio. The final step is to multiply each value in the attack-decay curve by the obtained maximum value of the original audio. This process brings the attack-decay curve to the same scale as the original audio thereby bringing the attack-decay curve closer to the peaks of the original audio.

This is done purely to aid visual comparison. As we are interested in how the curve characterises the rise and fall in volume of the signal the way to test how well it performs is through psychological testing.

Results and Discussion

I ran the algorithm on all combinations of filter type, with filter sizes from 1 to 5 cycles, and achieved some interesting results.

The combination of different filter types and sizes makes a significant difference. Consider figure 3.15. The box filter of size 1 produces a significantly better attack-decay curve than the Gaussian-like filter of size 1. But if we look at figure 3.16 we see that the results of the Gaussian-like filter are far better with a filter size of 5 but the box filter does not improve by nearly as much. The curves resulting from the all the filter types get smoother as the filter size is increased but at a certain point the curves become less representative than with smaller filter sizes as they have smoothed out too much detail.

The most interesting result comes from the triangle filter. Consider figures 3.17 and 3.18. Unlike the box and Gaussian-like filters that simply provide better curves as the filter size increases up to a point, the triangle filter performs better as the filter size increases on the whole but the results are notably worse when the filter size is an odd number. When the triangle filter size is an even number the resulting curves are the best results I have achieved. Figures 3.19 and 3.20 provide more proof of this. A closer comparison of the effect of the triangle filter of size 4 and the Gaussian-like filter of size 5 on the alto saxophone note can be seen in figures 3.21 and 3.22. The difference is slight but notice a wobble in the curve obtained with the Gaussian-like filter.

Part of my criteria for attack-decay curve detection algorithms was that they be automatic. My description of this pitch-based convolution technique requires that the

user provide a pitch and the size of the filter in cycles. However, a pitch detection algorithm could be used to automate this. Additionally, given that the triangle filter with an even size gives such good results, perhaps the filter size could be fixed at a value such as 4, pending more thorough testing.

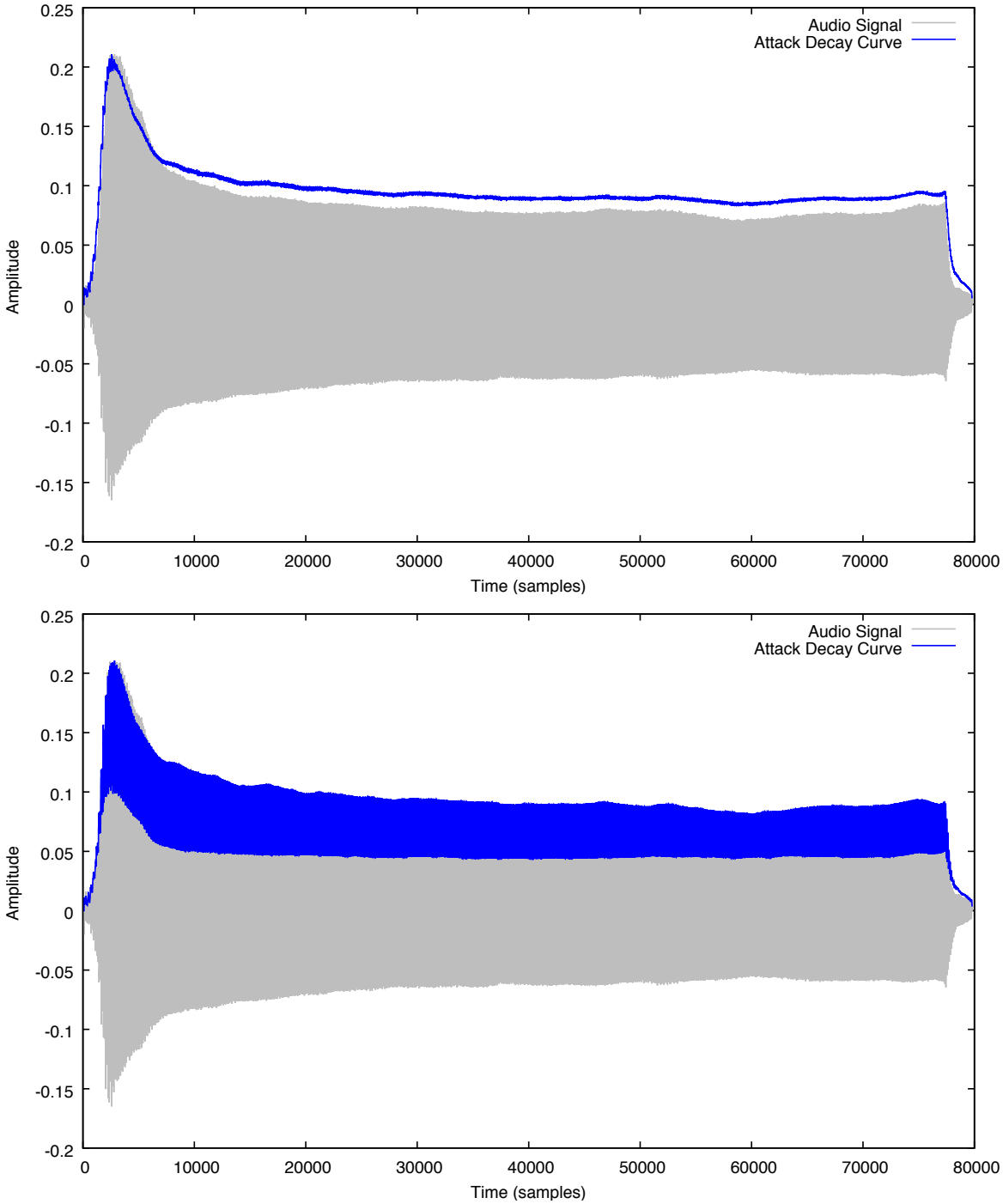


Figure 3.15: An A440 alto saxophone filtered with a box and Gaussian-like filter of size 1

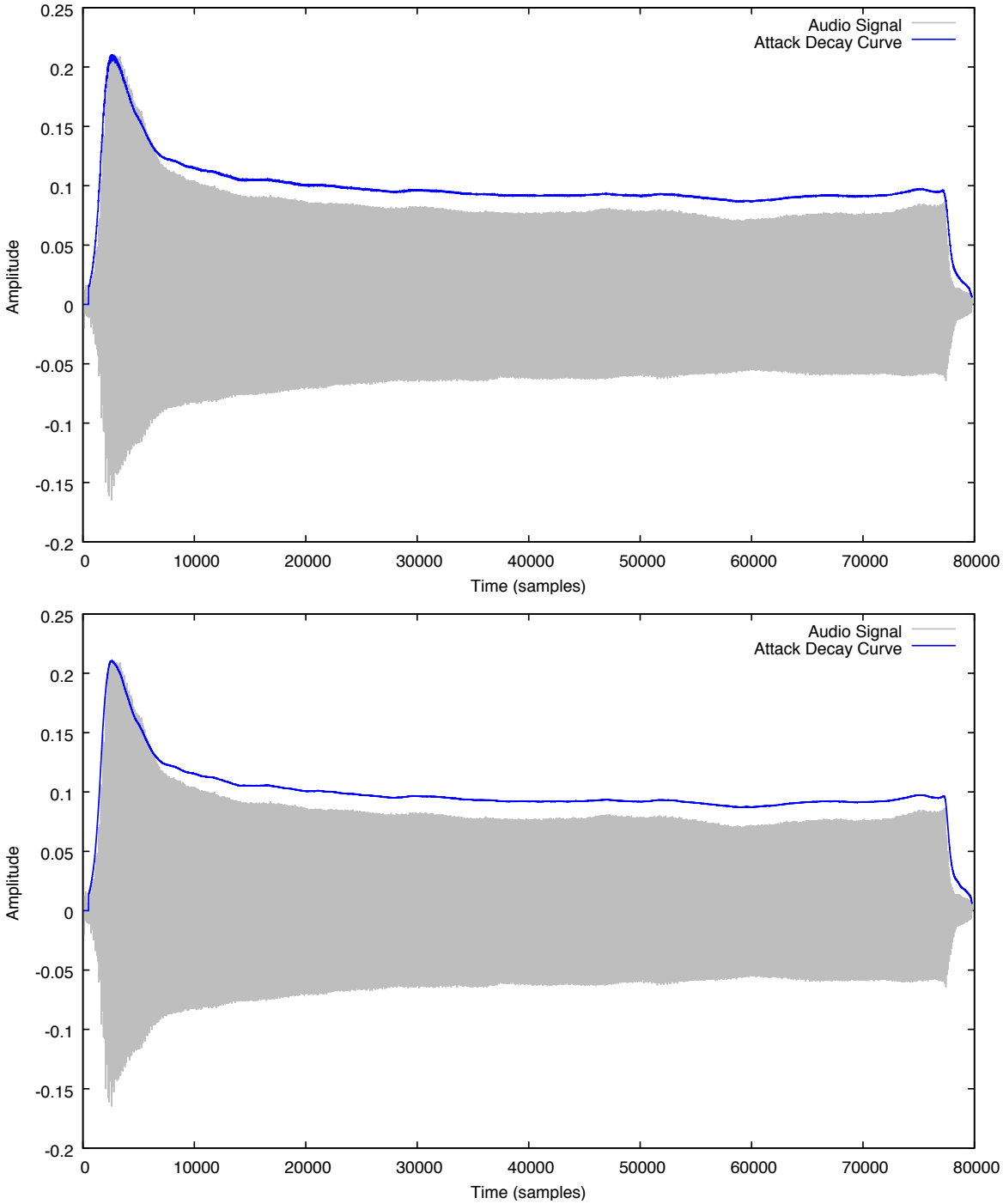


Figure 3.16: An A440 alto saxophone filtered with a box and Gaussian-like filter of size 5

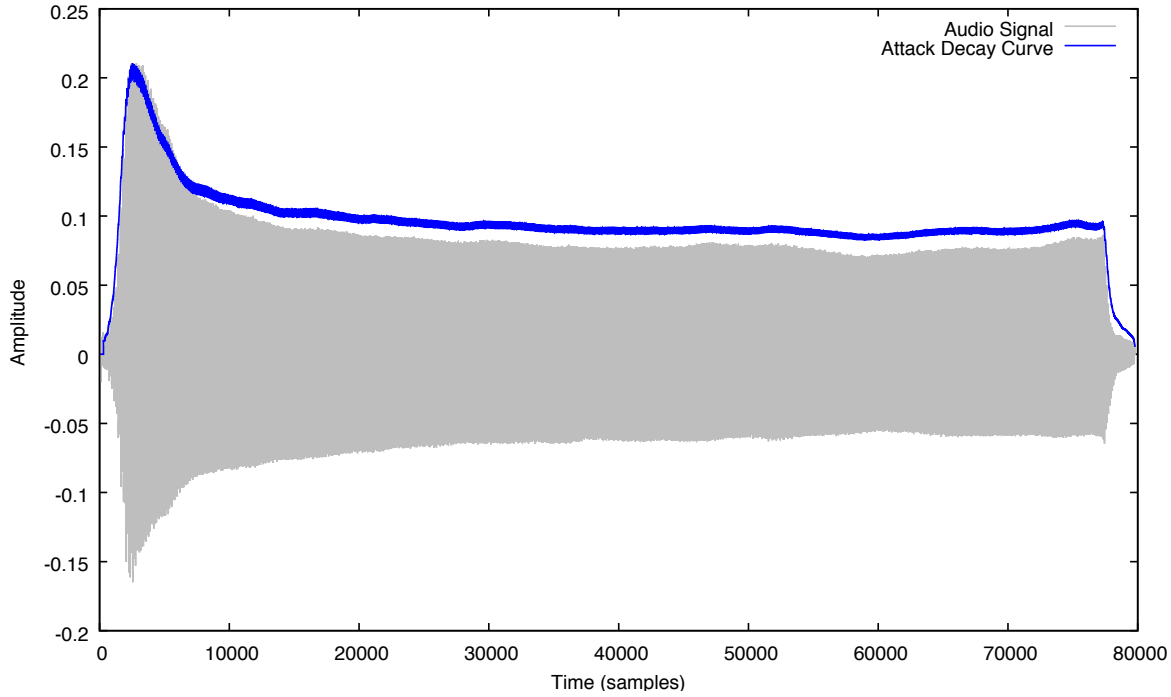


Figure 3.17: An A440 alto saxophone filtered with a triangle filter of size 3

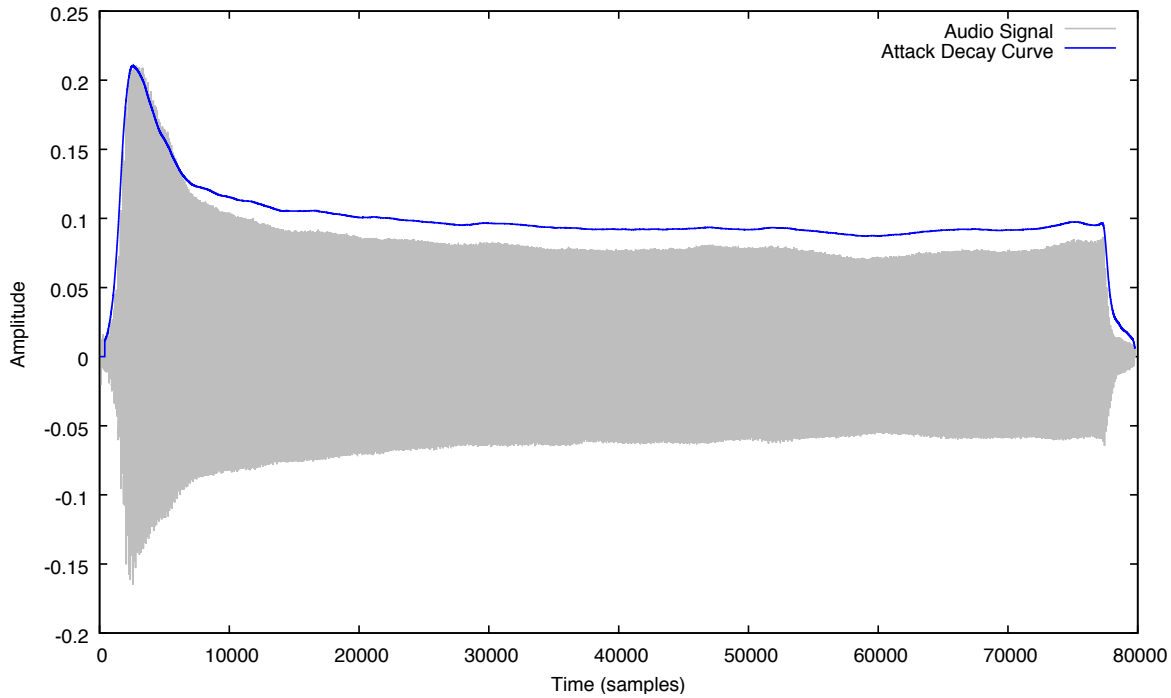


Figure 3.18: An A440 alto saxophone filtered with a triangle filter of size 4

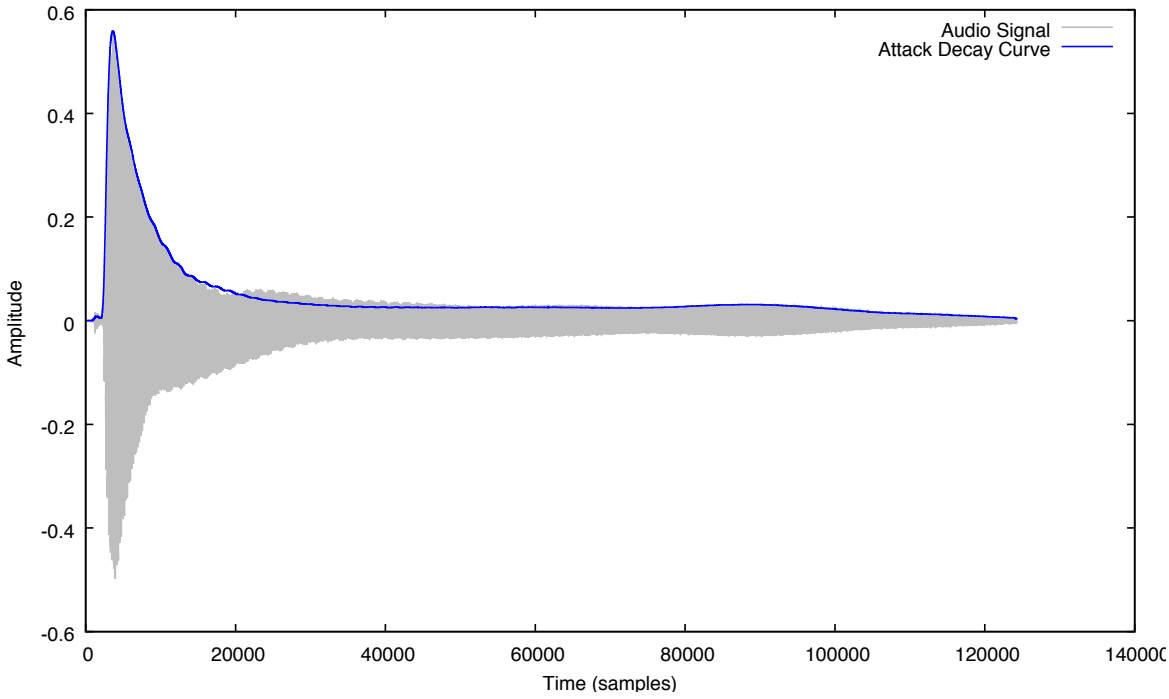


Figure 3.19: An A220 guitar filtered with a triangle filter of size 4

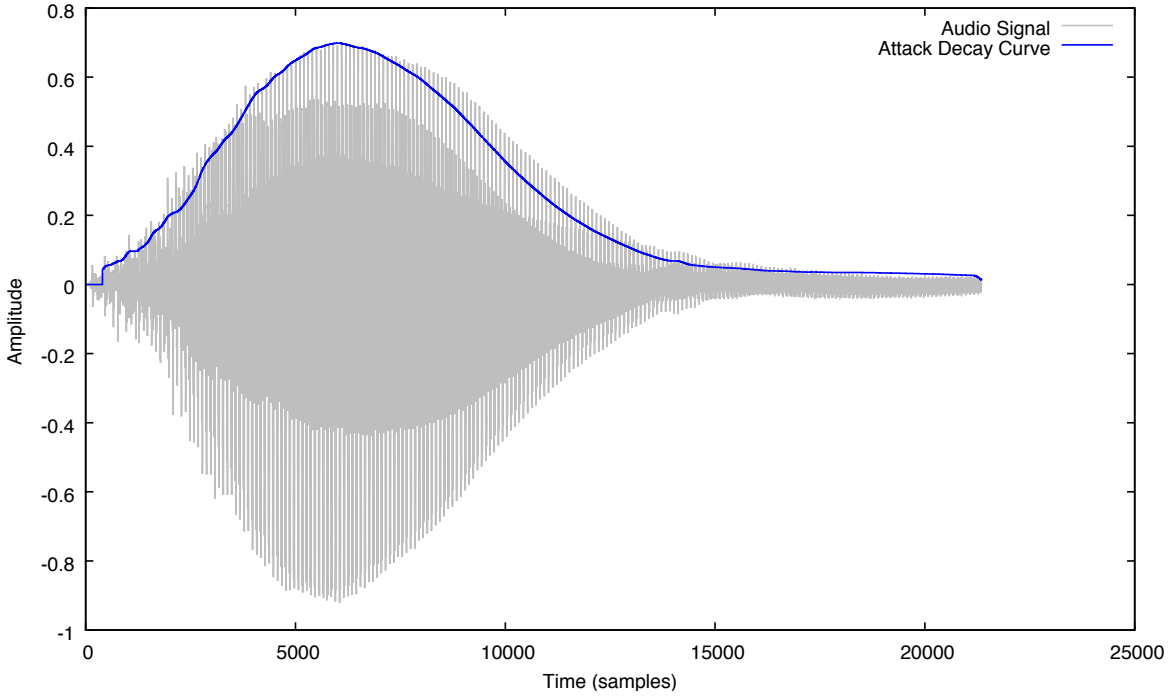


Figure 3.20: An A440 violin filtered with a triangle filter of size 4

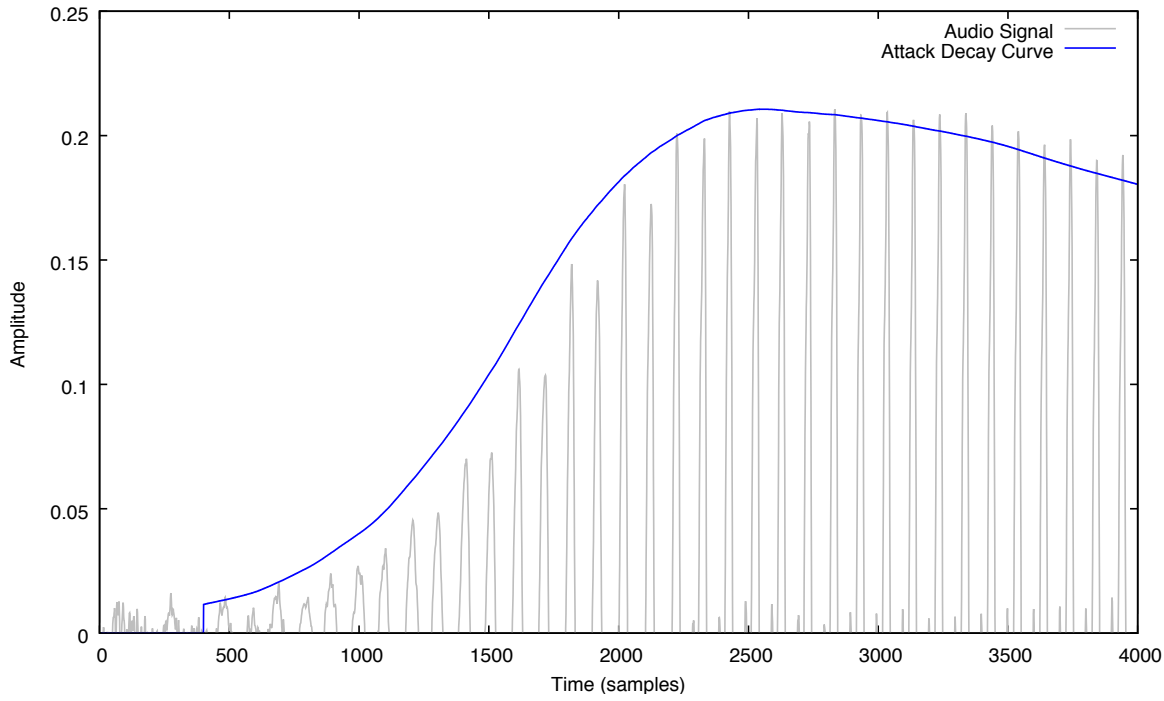


Figure 3.21: A close view of an A440 alto saxophone filtered with a triangle filter of size 4

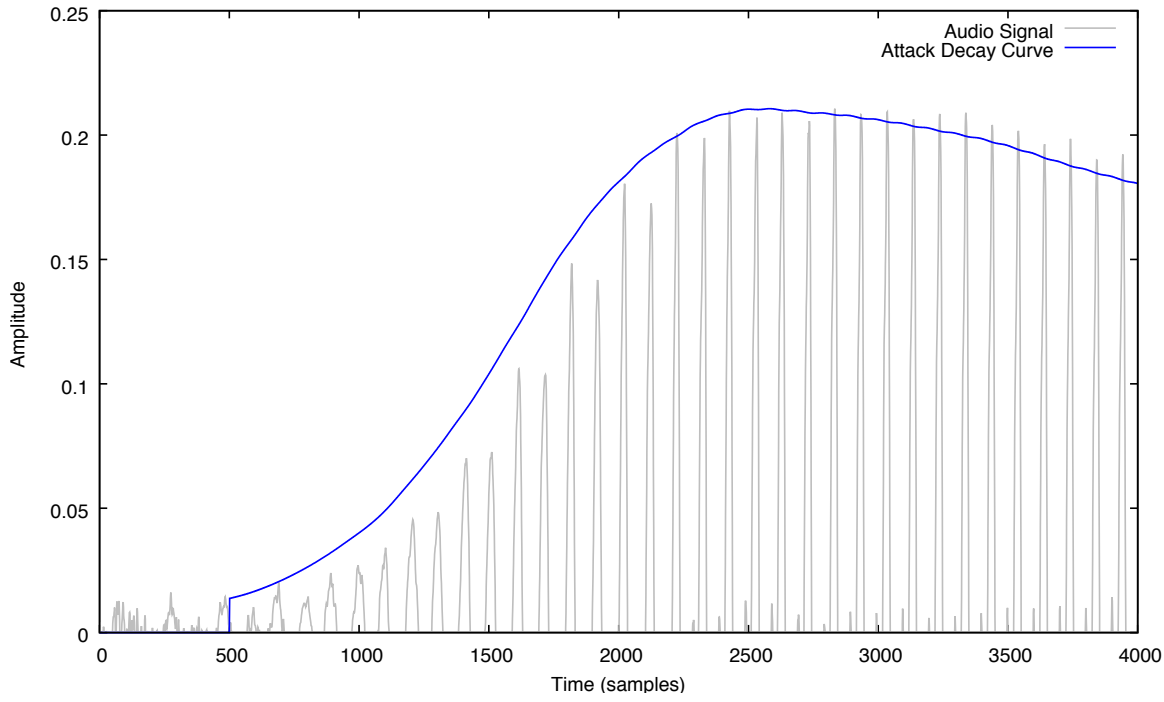


Figure 3.22: A close view of an A440 alto saxophone filtered with the Gaussian-like filter of size 5

Chapter 4

Wave Editor

The second area of my contribution was the research and development of a program with a graphical interface that allows a user to manipulate the amplitude and frequency of a synthetically produced waveform in order to match it to a recorded waveform. While there are existing tools for waveform editing they were not fit for purpose and the reasons for this are explored below.

While experimenting with envelope algorithms an idea developed similar to representing a signal in terms of a carrier component and modulator component. A wave could be thought of as being composed of two parts; a repeating signal of constant amplitude and an attack-decay function but this type of representation could also serve as the basis of a model for timbre. The model is explored through the development of the wave editor, and expanded upon by introducing the idea of frequency adjustment in order to more accurately reproduce a waveform.

4.1 Motivation

The motivation for developing a tool that allows a user to match a synthetic wave to a reference wave, is to explore what the modified synthetic wave sounds like when compared to the real audio reference wave. This serves as a means of testing the hypothesis described in section 1.3.2. The synthetic wave is a produced by repeating a single cycle obtained from the steady state portion of the audio wave. This is motivated by our idea of a wave being composed of a repeating signal of constant amplitude and an attack decay function that characterises the rise and fall of the wave. The difference is that here the attack decay function is applied manually. Once the synthetic wave has been matched as accurately as possible, the resulting waveform can be played back.

If the waveform sounds close to the original audio then this vindicates our decision in choosing to produce the synthetic wave from a single repeated cycle and also our decision to model a wave in terms of a repeating cycle of constant amplitude and an attack decay function.

In the course of this research I found that the beginning of notes played on real instruments are often sharp, meaning that the pitch at the start of the note is higher than the musician intended. Or, more technically, the attack of a note is often a higher pitch than when the wave is stable. This is not due to a fault on the musician's part but it is an inherent characteristic of many instruments. This means that in order to match the wave as accurately as possible there needs to be a method for modifying the frequency of the synthetic waveform. As the frequency is different in different parts of the wave that frequency control needs to be fine grained.

There are many music and waveform editing tools already in existence that provide a large number of features, primarily for the purpose of music production. Examples include Audacity, Pro Tools, Apple Logic, and Ableton, but there are many more. These digital audio workstations (DAWs) are highly useful for their purpose and provide many different audio effects and methods for applying them. This, of course, includes allowing the user to modify the frequency and amplitude of waveforms. Amplitude modification is fairly straightforward and DAWs even allow for complex amplitude adjustment according to different types of curves. The problem is the frequency modification and more specifically the method by which you alter the frequency of the wave. DAWs typically allow a user to select a segment of the waveform and 'pitch shift' it. As the increase or decrease in frequency results in the compression or expansion of the wave in time the DAW will often require input from the user as to how they want the waveform to change in time. For example, does the user want the pitch shifted wave segment to compress or expand in time proportionally to the increase or decrease in frequency? Or does the user want the pitch shifted wave segment to take up same amount of time as the original wave segment? Or does the user want to define a new amount of time for the segment to occur over? In the second and third case the DAW has to determine how to fill the amount of time the user has specified and some DAWs offer a number of different pitch shifting and time stretching algorithms for this task. If the pitch shifted wave segment needs to take up more time than the amount proportional to the change in frequency, then the DAW needs to synthesise more of the wave segment in order to fill up that time. This can lead to strange waveform interpolation that results in significant changes in the harmonic content of the wave. The issue here is that the

wave segment is unlikely to be a single cycle, which would be ideal, or even a whole number of cycles and so synthesising more of the wave is not a simple task. If the pitch shifted wave segment needs to take up less time than the amount proportional to the change in frequency then the wave segment needs to be truncated somehow which could result in a discontinuity in the waveform. In the worst case the DAW would not allow a user to elect to have the time change be proportional to the frequency shift. This means that the user would have to manually enter the parameters to allow for this and that would become difficult when altering the frequency by small amounts. It is also possible that the DAW would not allow the user to modify the frequency by the small amounts that we will need to.

All of these failings could perhaps be accounted for with careful editing of the wave. But this is far from the ideal scenario. To summarise, the problems with using existing wave editing software or DAWs are: the method by which the user controls the frequency modification, the degree of control the user has on frequency shifting, and the errors introduced by how the frequency shifting is implemented. These problems can be separated into two groups. The method for controlling frequency modification and the granularity of the frequency controls are problems relating to the software's interface. The error introduced by frequency shifting is a problem with the underlying implementation(s). Because we require a very fine level of control over the frequency at any point in the wave, selecting segments of wave and providing a percentage for time and frequency shift is inadequate. We need to be able to match the synthetic wave to the real wave in terms of peak by peak accuracy. If using the previously mentioned method the user would be required to guess the percentages that would align the synthetic wave to the real wave. While a user might get better at this over time it is far from ideal. I needed a system that was easy to use so that wave matching could take place in a reasonable time or better yet with immediate accuracy.

The remaining problem of how the frequency shifting is implemented is not a simple one. Some DAWs allow the use of several different pitch shifting and time stretching operations either via the use of plugins or through native support. In either case it is difficult to ascertain precisely how the pitch shifting is implemented, which is hugely important as, in our case, we need to be sure that the harmonic structure of the waveform is not being changed.

So what is needed for the frequency adjustment? Firstly the user needs to be able to alter the frequency by very small amounts at arbitrary locations along the waveform. This is necessary as the difference between certain parts of the reference

and the synthetic wave could be fairly small and we need to allow the user to match the waves as accurately as they care to. Secondly, the interface for modifying frequency needs to be simple to understand and quick to operate. Matching the synthetic wave to the reference wave might require many adjustments and so having each individual adjustment be simple to initiate and quick to resolve is important. Thirdly, as we require the ability to match the waves peak to peak, and modification by possibly very small amounts, the method for controlling the frequency modification should be direct and real-time. By direct I mean that the control method should involve interacting with the waveform directly as opposed to indirect control via parameters in a dialogue box. The problems introduced by DAWs when performing pitch shifting and time stretching cannot be part of our solution to the wave matching problem.

As the issues discussed above are present in existing software the remaining options were to develop an entirely new application or to extend one of those existing pieces of software. There are a number of open source DAWs and applications designed to aid the user in interacting with audio files. These include 'Ardour', 'Audacity', and 'Sonic Visualiser'. Sonic Visualiser is distinct in that its primary application is musical analysis rather than music production like DAWs. It allows easy extension through the use of the 'Vamp' plugin system, these plugins do not require modifying the application's source code in order to implement new functions, however they seem to be used exclusively for analysis and annotation of the wave rather than transformation of the raw wave data. In addition, the interface for the plugins is restricted to a dialogue box similar to that seen in Audacity and other DAWs. So in order to match the frequency of one waveform to another the user would be required to perform the same trial and error process that is required for pitch shifting in DAWs. So utilising vamp plugins as a means of achieving our goal has been ruled out as it violates our criteria.

The alternative to using plugins to implement wave matching is to modify the source code of one of these applications; the motivation being that working with an existing code base might increase the speed of development. In order to implement wave matching in a way that suits the criteria described above I would have have to locate and modify the source code responsible for a number of different things. It would require the modification of mouse and keyboard handling functions in order to allow direct interaction with, and modification of, the wave. In applications like audio editing software where the graphical user interface is complex, the logic behind handling mouse and keyboard events could be fairly complicated, increasing the likelihood that

any modification I make could interfere with pre-existing mouse or keyboard handling. Once the mouse and keyboard handling is in place, I would need to know how to transform the position of my mouse within the waveform window into the coordinate system in which the waveform data is represented. This is because the mouse pointer location relative to the waveform would be important to the logic for waveform manipulation. Additionally, I would need to write code that defines how the wave data should change in response to the behaviour of the mouse and keyboard, and then have to ensure that the changes are consistent with the internal representation of the waveform, and are then represented when the waveform is rendered. All of these pieces of logic would likely be in different places in a significantly large code base. Any additional rendering such as control points for amplitude and frequency, or wave annotations would also have to be integrated into the already complex code base of the audio editing software. Essentially, the possibility is too high that understanding and modifying an existing code base would hinder progress more than it would help.

This leaves only one option remaining; to write an entirely new application. By writing a new application I would spend a fair amount of time in order to get the application running but that the time would be balanced out by the speed at which I could introduce new features due to my familiarity with the code and my ability to manipulate it in its entirety. Also as DAWs are primarily concerned with music production and not musical analysis this application can serve as a base for timbre related analysis.

In relation to the previously discussed criteria the development of a new application makes a lot of sense as we have full control over the granularity of frequency shifting and the mechanism by which the frequency shifting is controlled. This means that frequency manipulation can be quick, easy, real-time, and direct according to our wishes. Also, because we know that our synthetic wave is produced from a single repeated cycle, and we are fully in control of how frequency modification is controlled and implemented, we can ensure that the change in harmonic content or discontinuities in the wave introduced by frequency shifting and time stretching algorithms in DAWs are mitigated completely.

4.2 What does it do?

In its current form the wave editor enables the user to load in an audio file and an isolated cycle from that waveform from the command line. It then duplicates that

isolated cycle into a full waveform and renders both of these to the screen. At this stage the user will see a view like Figure 4.1. The blue wave is the waveform of the input audio file and the red wave is the waveform produced by repeating the isolated cycle. On the y-axis is sample value and on the x-axis is time.

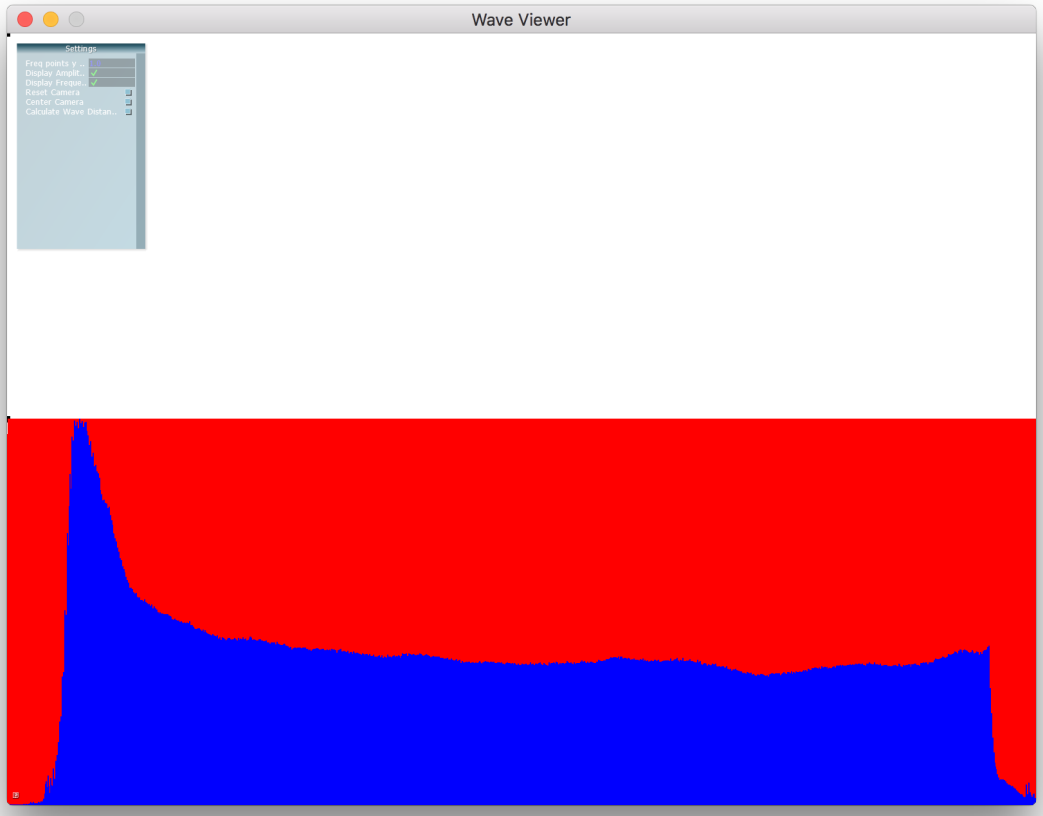
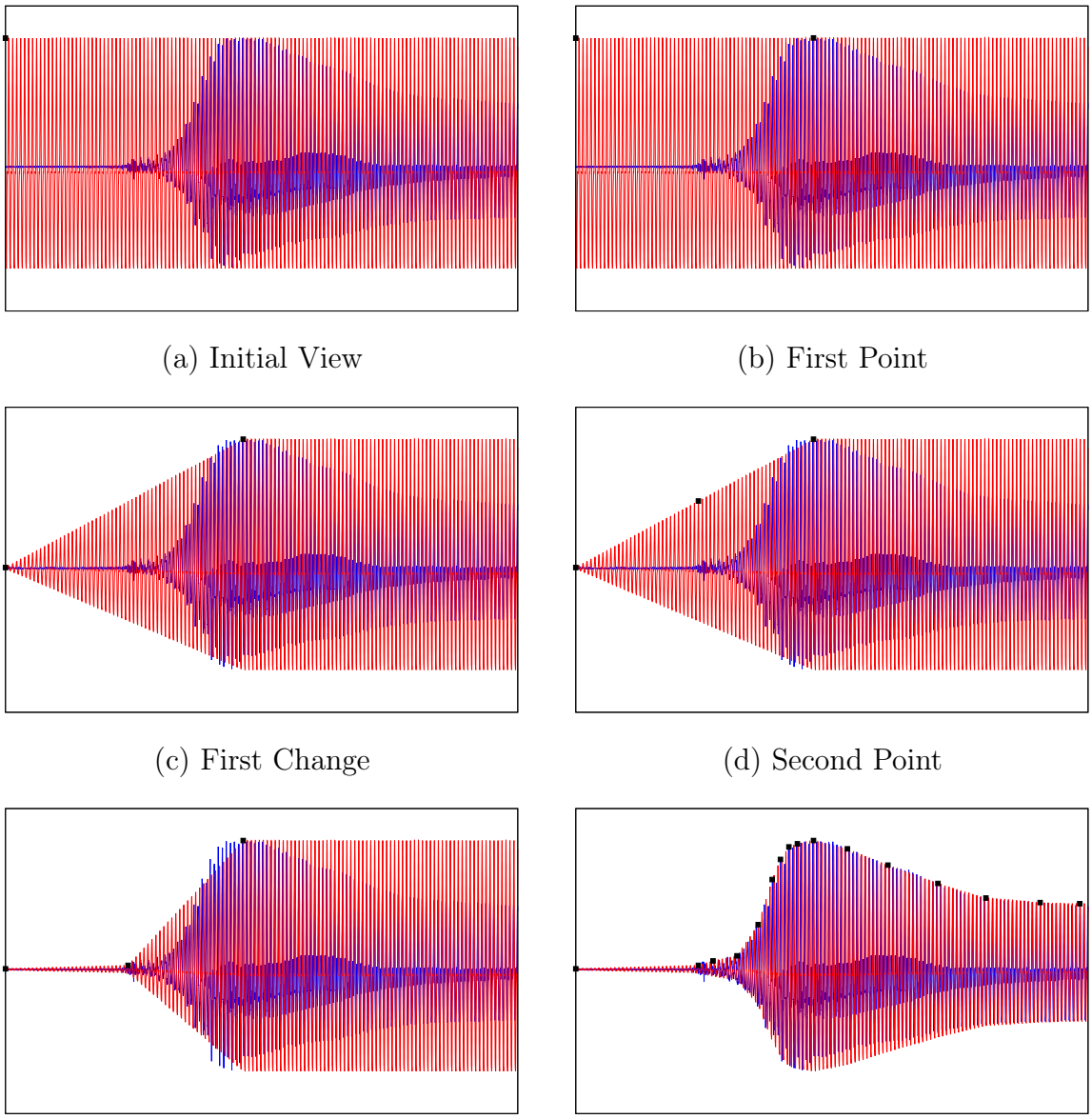


Figure 4.1: The Wave Editor on start up

Using the keyboard and mouse the user can add, remove, and manipulate two separate sets of control points. The first set of control points are responsible for controlling the amplitude of the synthetic wave. These amplitude control points can be placed anywhere on the wave and moved up and down. Every time the control points are modified in any way a curve between them is calculated using linear interpolation and that curve is then used to scale the wave. A demonstration of this process and the resulting changes to the waveform can be seen in Figure 4.2.

In image (a) a zoomed in view of the wave can be seen that is unmodified from start up. The black square on the left of the image is the first default amplitude control point and there is a corresponding amplitude control point out of view at the end of



(a) Initial View

(b) First Point

(c) First Change

(d) Second Point

(e) Second Change

(f) Matched Wave

Figure 4.2: Applying an attack-decay curve through amplitude control points

the wave. Image (b) shows the same view but with a newly added amplitude control point. In image(c) the default control point has been moved downward approximately to the level of the blue input audio wave. The equation for the line that connects the two control points is calculated and used to generate a value for each element in the section of the synthetic wave between the two control points. These values are then used to scale the section of the synthetic wave that is between these two control points and this change is reflected in the red waveform in image (c). In images (d) and (e) a second additional control point is added and then adjusted to bring the red synthetic wave closer to matching the blue audio waveform. Notice that the red waveform is only altered between the control points on either side of the selected control point. This is due to the way the control points scale the wave according to a linearly interpolated curve. Each point only affects the curve between the two control points either side and this behaviour is reflected when the scaling is applied to the synthetic waveform. Finally in image (f) we see the fully matched wave section. At this point the wave is approximately as close as it will get to the input audio wave through the manipulation of amplitude control points. Despite the amplitude control points only being placed on the upper part of the waveform, the entire wave is being scaled by these points. Waveforms are typically close to symmetrical so for this first version of the wave editor I chose to implement only one set of amplitude control points. To improve the fit, and accommodate less symmetrical waveforms, using one set of amplitude control points for the upper waveform and one set for the lower waveform could be trivially implemented. Waveform symmetry is discussed in section 3.3.4.

If we look closer at the wave as in Figure 4.3 we see that the synthetic wave is not well aligned with the audio wave. The isolated cycle that the synthetic wave is generated from is taken from the steady portion of the audio wave so why then are the two waves so misaligned? One of the reasons and one that formed part of the motivation for developing the wave editor is that the attack of a note is often sharp. The harmonics of the wave being slightly out of tune also has an effect on the alignment of the waveforms and this is discussed in section 4.3.1. An instrument's attack being sharp means that the frequency of the recorded wave is higher during the attack than when the note has settled into its steady state.

The synthetically produced wave has a constant frequency because it strictly repeats a single cycle taken from the audio wave. This means that cycle to cycle the peaks of the synthetic wave will always be in the same place but the same is not true for the audio wave. Because of the varying frequency of the audio wave the individual

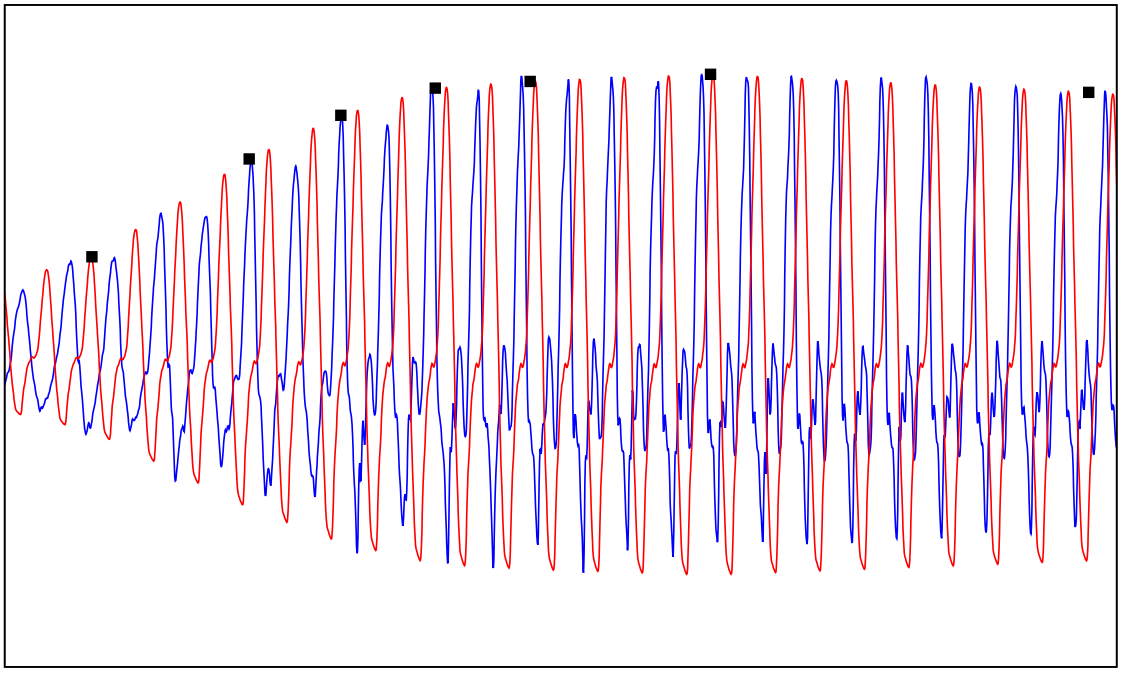
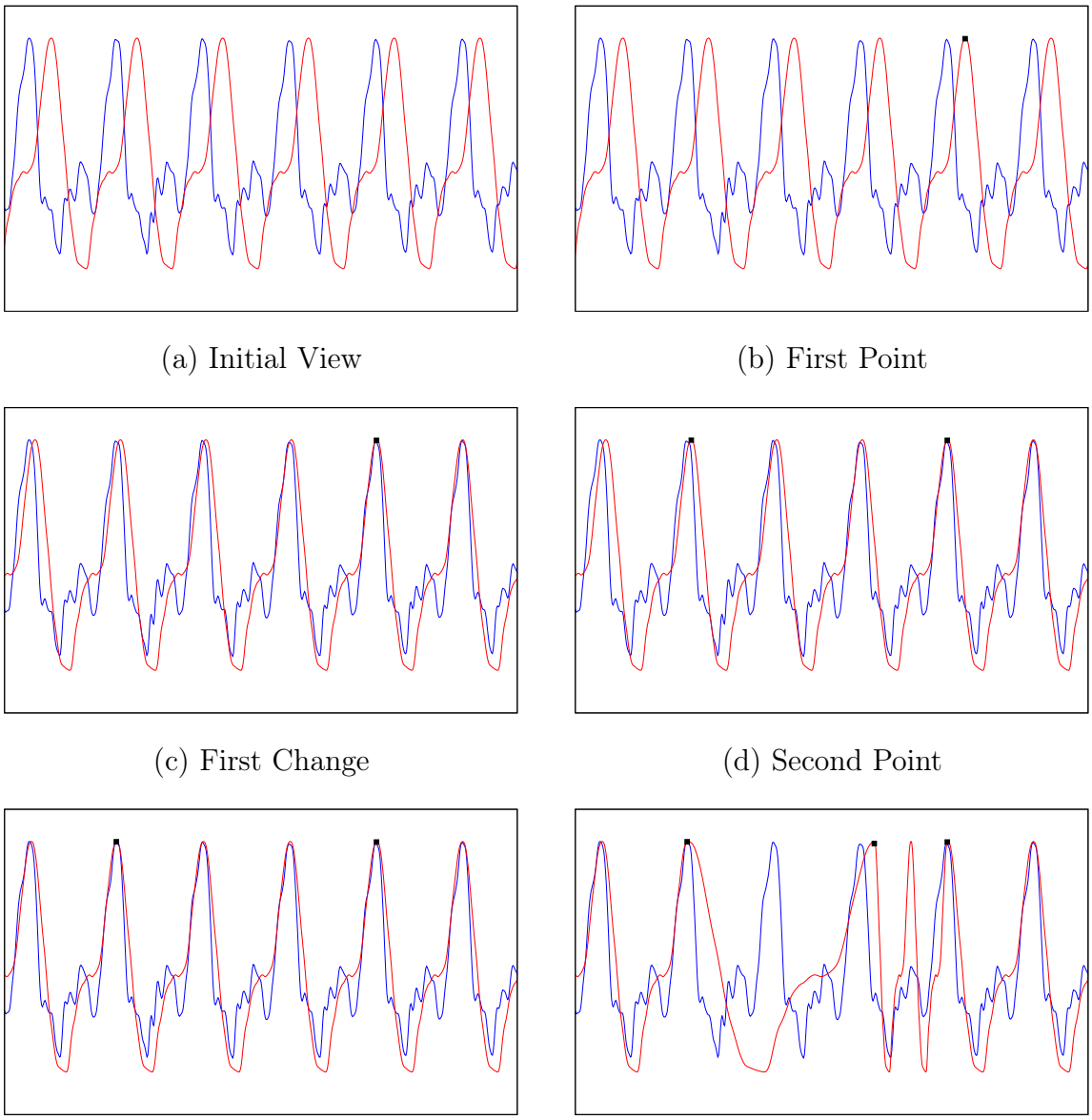


Figure 4.3: A close up view of the amplitude matched wave

cycles take varying amounts of time so the peaks occur in different places relative to the synthetic waveform. These differences are not constant as the frequency of the audio wave decreases as the note decays to its steady state where the frequency of the note is the one intended by the musician. Evidence of this can be seen in Figure 4.3. The blue and red peaks furthest to left are approximately a half cycle out of phase but as the wave continues to the right the peaks get closer together. The non constant frequency difference between the attack and the steady state, from where the isolated cycle is taken, means that matching the synthetic wave to the audio wave requires a number of small frequency adjustments to get the peaks aligned as closely as possible.

These adjustments are achieved through a set of frequency control points and these effectively compress or extend sections of the the wave in time. For example, when a point is pulled to the left then the section of wave to the left of the point will be 'squashed' or more accurately the wave section will be condensed in time which increases that wave segment's frequency. Conversely the section of wave to the right will be 'stretched' extending it in time and decreasing its frequency. For more detail consider Figure 4.4.

In image (a) there is a close up view of the blue audio wave and the red synthetically produced wave. The synthetic wave has been adjusted to approximately match the au-



(a) Initial View

(b) First Point

(c) First Change

(d) Second Point

(e) Second Change

(f) Third Change

Figure 4.4: Demonstrating the effect of frequency control point adjustment

dio wave in amplitude which is evident by the matching heights of the prominent peaks in both waves. But the waves are misaligned primarily due to frequency differences. A new frequency point can be added and manipulated with the mouse and doing so will produce a view similar to image (b). This new point can be pulled to the left or right to adjust the frequency either side of the control point. Dragging this new point to the blue peak nearest to the left produces image (c). The process thus far has been simply to click a red peak, or anywhere on the red wave and drag it to the nearest blue peak and release the mouse.

Notice that some of the peaks are now well aligned but others require further adjustment. This is due to the difference in frequency between the attack and steady state being non constant, meaning that aligning one peak and having the rest of the wave move proportionally is insufficient. So to improve the match between the audio and synthetic waves more frequency adjustment is required.

In image (d) a new control point is added towards the left of the window and then in image (e) that point is shifted slightly to the left. The alignment between the peaks is now very close to being as accurate as possible. The reader might notice that some peaks do not seem as closely matched as the others. This difference is due to difference in the shapes of the peaks in the audio wave. Looking at the difference between the audio and the synthetic waves on the right side of the peaks you will see that the area between the red and blue waves is fairly consistent between the peaks. This is evidence that the matching process has been performed to the same degree of accuracy but yet the differences in peak matching still exist. The difference in shape of the peaks of the audio wave is because the overtones of the audio not being perfectly in tune. So for each cycle of the fundamental frequency the magnitude of the contribution of each overtone varies, and this causes the differences in wave shape over time. These differences are not present in the synthetic wave as it is produced from a single cycle of the audio wave. This is discussed in more detail in section 4.3.1.

In image (f) a third control point has been added and dragged to the right which has increased the frequency of the segment to the right and decreased it to the left. This makes the wave alignment significantly worse and was done in order to illustrate how the frequency control points work in isolation. When that middle control point is moved, only the section of wave between the point's neighbouring control points is affected and this can be seen in the difference between image (e) and (f). Outside of those neighbour control points the wave is the same in both image (e) and (f) so the frequency control points can be thought of 'anchoring' the wave. Having changes to one

frequency control point cascade and change the alignment further along the wave made matching the waves difficult so anchoring the wave at the control points is necessary. Relevant implementation details are discussed in appendix B.

4.3 Results and Discussion

By using the wave editor to apply an attack-decay curve and to perform frequency adjustments, a user can create a fairly close visual match between the waveforms of a recorded instrument note and a synthetic wave generated from a single cycle of that recorded signal. Consider Figure 4.5.

The top image shows the audio and synthetic waves in the audio wave's steady state. The match here is fairly close to perfect. The bottom image shows the match during part of the attack of the audio wave. The quality of the match during the attack, and to a lesser extent in some other parts of the wave, is worse than the match for the steady state of the wave. If we have adjusted for the amplitude and frequency then what is the problem?

4.3.1 Wave Shape Differences

The difference between the steady state and attack of the audio wave is that the wave at these points is a different shape. The steady state match in the top image is so close because the single cycle from which the synthetic wave is generated, is taken from the steady state of the audio wave and so the synthetic wave's shape is the exact same shape as most of the steady state of the audio wave. This raises two questions: why do we generate the synthetic wave from a steady state cycle? And what causes the difference in wave shape over time? By explaining some of the cause of the changing wave shape it will become clear why we generate the synthetic wave from a steady state cycle.

Noise Frequencies

There are several factors that have an effect on the differences in wave shape over time. Firstly the presence of noise frequencies in the audio is important to consider. The audio in our case has a fundamental frequency and a series of overtones and what remains are the "noise frequencies". These are the frequencies not associated with the harmonic content of the sound and they have a significant effect on timbre. Examples

include: the click of a plectrum on a guitar string, the hammer of a piano hitting the strings, a bow rubbing against the strings of a violin, or the breath of the musician playing a saxophone. Noise frequencies are particular to a given instrument, they define part of the instruments character but like harmonic content they are not uniform for the duration of the note. As the noise frequencies are different over time, the wave at any point has a varying contribution from the different noise frequencies and this is part of the cause of the wave changing shape over time.

A single cycle from the steady state is supposed to capture the characteristic tone of an instrument. We want to generate our synthetic wave from a cycle that is most representative of the timbre of the instrument. So because noise frequencies are particularly prominent during the attack of a note, taking our single cycle from the attack is a poor choice. Additionally, as I've demonstrated in the figures above, the frequency and amplitude is most variable during the attack of the wave which further supports the decision to not take our cycle from the attack. We take the single cycle from the part of the wave that isn't affected by the variability in amplitude, frequency, and noise that is present in other parts of the wave. This is why we refer to it as the steady state of the wave; it is the most constant part of the signal in all respects.

Overtones

While the steady state part of the wave is the most constant, like the rest of the wave, it is not identical cycle to cycle. These differences are due to the overtones of the instrument being slightly out of tune. When a note is played at a given pitch instruments produce a set of related frequencies. These frequencies begin with the fundamental frequency, which is the pitch that we perceive when listening, and continue in integer multiples of that fundamental frequency. In reality these secondary frequencies, or overtones, are often not quite in tune. They are not exact multiples of the fundamental frequency. So in theory for every cycle of the fundamental frequency there should be two cycles of the first overtone, three cycles for the second overtone and so on. If these overtones are perfectly in tune with the fundamental frequency, then every time a new cycle of the fundamental begins so too will new cycles of the overtones, in other words, they remain synchronised. In reality an instruments overtones are slightly out of tune and this results in the overtones shifting compared to each other and the fundamental frequency. Consider a first overtone that is slightly flat. For every cycle of the fundamental this overtone in theory should have two full cycles, but because it is flat these two cycles take slightly longer and so fall out of phase with the fundamental

frequency. This phase difference compounds overtime and with each overtone being slightly of tune by a varying amounts different phases are exhibited by each overtone. So for any given cycle of the fundamental the set of overtones contribute with varying strength and phases differences and this results in the wave shape changing cycle to cycle.

Limitations of the Single Cycle

The shifting overtones are why it is not possible to exactly match our synthetic wave to the audio wave even when the audio wave it is its steady state. The single cycle is an approximation to the character of an instrument. In reality the timbre of an instrument is encoded in its attack-decay curve, frequency shifting over time, the frequencies of its overtones, its noise frequencies, and possibly more. All of these features of timbre cannot be captured in a single cycle.

4.3.2 Automation

The wave editor in its current form requires the user to manually apply the attack-decay curve and perform frequency adjustment as well as to provide a .wav file containing a single cycle from the steady state part of the wave. This differentiates the wave editor from the work in the previous chapter that lead to the development of the application as the attack-decay curve detection in Chapter 3 was automatic. The idea was to start with a tool that could reproduce the audio waveform as accurately as possible and it made the most sense to start by doing that manually.

It is possible to automate the placement and adjustment of the amplitude and frequency control points, as well as the extraction of the single cycle from which to generate the synthetic wave, but due to time constraints this is left for future work.

4.3.3 Outcomes and Future Work

By using the wave editor to match between the audio wave and the synthetic wave, a match can be achieved that results in a subjectively similar timbre when the synthetic wave is exported as audio and listened to.

This supports the hypothesis that an attack-decay curve and frequency adjustment applied to a single repeating cycle form at least a partial description of timbre. However, due to the subjective nature of timbre perception, to truly establish how well the

output of the wave editor recreates the timbre of the input audio, psychoacoustic experiments would need to be conducted. These experiments would likely involve presenting participants with original and reproduced audio and asking them to judge the similarity between the presented samples. It might be plausible to have a range of partially matched audio to present to participants as a way of judging the individuals sensitivity to differences in timbre. Once psychoacoustic experimentation has established how representative this model is, the attack-decay curve and frequency adjustments could be extracted and used as timbre descriptors in classification systems, other areas of music information retrieval, and research related to timbre.

This proposed description of timbre is not complete as it cannot yet accurately reproduce noise frequencies and the shifting of overtones that occurs in instrument audio. By capturing the noise frequencies and the shifting of the overtones we will be that much closer to a precise definition of timbre. The next steps for the wave editor are to try and capture these elements of timbre. By performing a Fourier transform on the waveform, we would obtain a description of the magnitude, frequency, and phase of each of the overtones. This information could be used to construct a synthetic wave that is more accurate to the recorded audio waveform and this would hopefully produce a final matched wave that is a closer match in timbre than the current output of the wave editor.

How to capture the noise frequencies present in the signal is a little more complex. Once the Fourier transform process is implemented, one possible avenue is to match the wave and then see what the differences are between the matched wave and the real audio. It is possible that the remaining audio is mostly composed of noise frequencies but this would require more experimentation to confirm and has been left for future work.

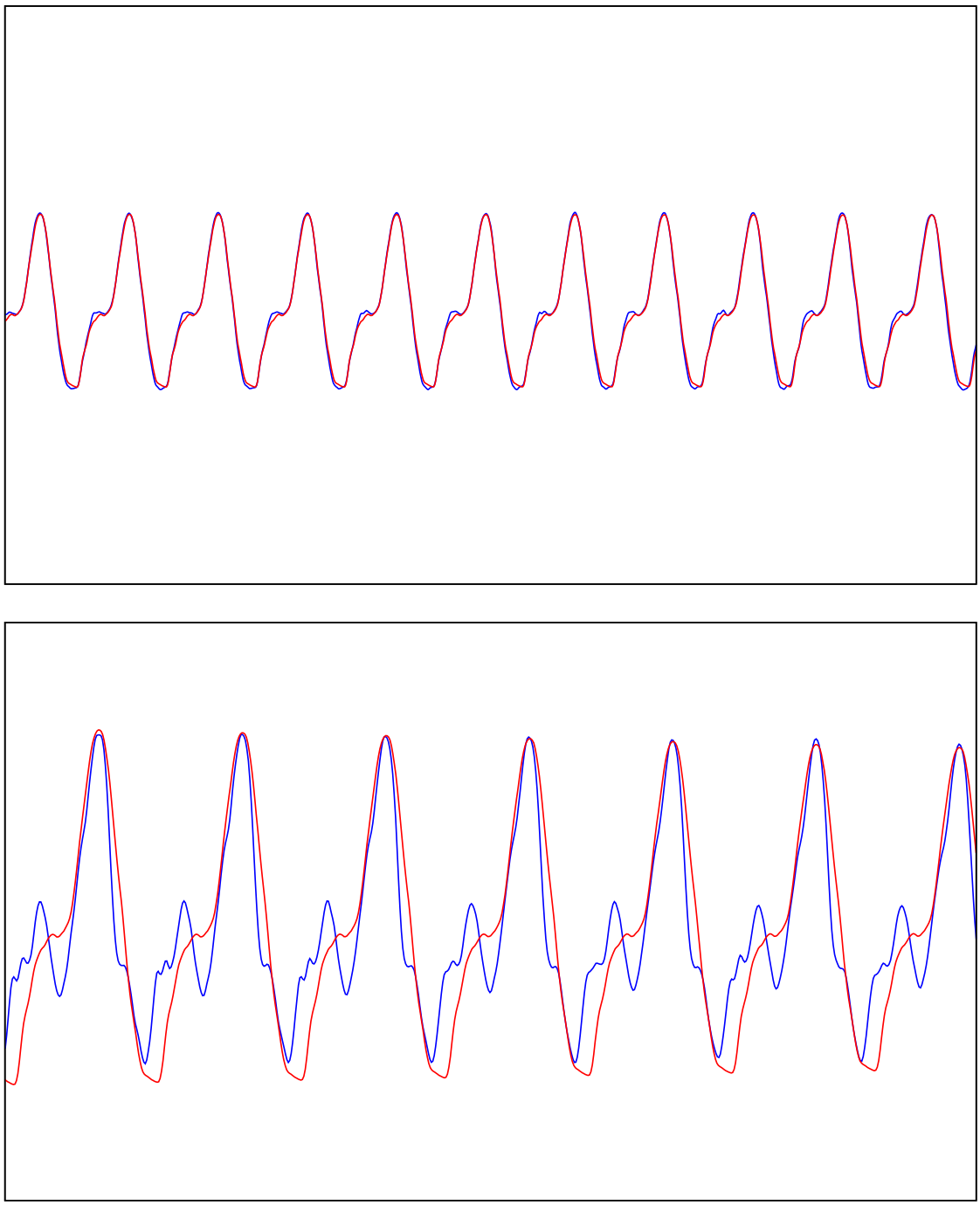


Figure 4.5: The steady state and attack of the matched waves

Chapter 5

Conclusion

The initial conception for this thesis was to develop a model for instrument recognition based on the perceptual quality, timbre. Typical examples of instrument classification techniques in music information retrieval take a machine learning or neural network based approach that is results focussed. While accurate instrument classification is a perfectly valid goal, no method is perfect and through the use of those methods we lose the opportunity to learn something new about timbre. My goal was to utilise knowledge of timbre perception to create an instrument classifier. The effectiveness of this classifier would serve to validate the model for timbre on which the classifier was built.

However, due to the complexities surrounding timbre my research was very exploratory. A key component of the initial model for timbre was the envelope, a curve describing the rise and fall in a sound's volume. Research on the existing techniques used for extracting this component from audio signals, and the development of my own algorithms lead to the beginnings of a new model for timbre. The basis for this new model was explored through the development of the wave editor application.

5.1 Summary of Envelopes

We know that the way the volume of a sound rises and falls affects the way we perceive the timbre of the sound. This served as the motivation to begin research into envelopes. An envelope is commonly described as a curve that varies slowly over time and is representative of the overall change in amplitude over that time. However, past that basic idea, there is little agreement upon the properties an envelope should have and what physical meaning should be attributed to it. Some researchers think that an

envelope is a curve that rests on top of the wave, and should pass through all its 'prominent peaks'. Others view a signal as the product of a rapidly varying carrier component and the slowly varying modulator component, which corresponds to the concept of an envelope, and they contend with the problem of deconstructing the signal into these separate parts.

Through my research I developed three separate algorithms, all of which model different but related phenomena and were based upon different ideas that were representative of direction of the research at the time. The first algorithm is a more conventional envelope extraction algorithm that is based on a geometric view of the signal and seeks to find a curve that covers the signal. The second algorithm is based on a simplified physical model of vibration in musical instruments and seeks to fit curves to the signal that are consistent with that model. The third and final algorithm is my proposed solution to modelling the rise and fall of volume in audio signals and uses smoothing filters to achieve this result. I describe the resulting algorithm as extracting the attack-decay curve, as a means of differentiating its purpose from those seen in the literature.

The first presented method consists of an algorithm that uses straight line segments and a restriction on wave intersection in order to approximate the envelope of the signal. As a conventional envelope extraction algorithm it can perform well. However, its performance is subject to the maximum line length parameter being chosen correctly.

The second method attempts to fit an exponential curve to the attack, and to the decay of the signal. The idea behind this method is that the vibration in musical instruments decays exponentially and therefore so does the resulting audio signal. This model assumes that the instruments producing the audio signals do not allow the performer to continue to supply energy to the instrument after the start of the note. When applied to instruments like a violin or saxophone the algorithm performs poorly as the energy supply to these instruments is under the performers control. When the algorithm is applied to a guitar the resulting curve is better than the violin or saxophone curve but it is still lacking due to the steady state of the note. In short, the use of two exponential curves is inadequate to model the rise and fall of a musical instrument signal.

The final method seeks to approximate the attack-decay curve of an audio signal by convolving the signal with smoothing filters sized according to the pitch of the audio signal. I experimented with a box, triangle, and Gaussian-like filter sized according to a varying number of whole cycles of the fundamental frequency. The result from

each filter type improves as the filter size increases, up to a point where the curve's representative power begins to decline. With a single cycle filter size the box filter performs the best, however as the filter size is increased the results do not improve significantly. The Gaussian-like filter produces worse results than the box filter initially but improves more significantly as the filter size increases. It performs slightly worse than the triangle filter, which shows the best results but with the restriction that the filter is sized according to an even number of cycles. The curve resulting from the triangle filter is the smoothest and is most representative of the wave.

5.2 Summary of Wave Editor

The smoothing filter approach led to thinking about a musical signal in terms of a signal of constant amplitude and an attack-decay curve. This idea as the basis of a model for timbre was explored through the development of the wave editor application. The application takes two .wav files as input, one containing a recording of a musical instrument note, and the other containing a single cycle from the steady state portion of that note. The single cycle is then repeated so that it matches the length of the note from which it was taken and both signals are rendered to the screen.

The user can then add, remove, and manipulate amplitude and frequency control points in order to visually match the repeated waveform to the original. This repeated waveform can then be exported as audio and played back. A well matched wave sounds subjectively close to the original waveform.

Establishing the perceptual similarity between the original and synthesised audio would validate the proposed beginnings of the model for timbre. This model would then provide a base upon which to develop a more complete model.

5.3 Future Work

5.3.1 Envelopes

The pitch based convolution method in its current form requires the user to give the pitch of the note when running the program. The next step is to implement automatic pitch detection. This would serve to make the method easier to use, but could improve the results of the method as well as notes are often sharp at the beginning.

The results of the pitch based convolution method are the most promising for

measuring the rise and fall in volume of a given signal. However, volume is a subjective perceptual attribute, and therefore the only way to test if the curves resulting from pitch based convolution are representative of the perceived change in volume is to conduct psychological studies.

Additionally studying and quantifying the effect of envelope shape on volume and timbre perception would be interesting. For example, it could be that we are not capable of perceiving smooth changes in envelopes, which could lead to more simple envelope related algorithms.

5.3.2 Wave Editor and Timbre Modelling

In my opinion, a synthetic wave that has been carefully matched to the original using the wave editor sounds very similar to the original wave. The next step is to perform a psychological study to formally establish the degree of perceived similarity between sets of recorded audio and synthesised audio. If this similarity is established, the model of timbre upon which the wave editor is based can be extended according to the steps detailed in section 4.3.3.

References

Association, A. S. (1960). Acoustical Terminology SI. *1-1960, American Standards Association*.

Athineos, M. and Ellis, D. P. W. (2003). Frequency-domain linear prediction for temporal features. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*, 261–266.

Benetos, E., Kotti, M., and Kotropoulos, C. (2006). Musical Instrument Classification using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Volume 5, V–V.

Berger, K. W. (1964). Some Factors in the Recognition of Timbre. *The Journal of the Acoustical Society of America*, *36*(10), 1888–1891.

Caclin, A., McAdams, S., Smith, B. K., and Winsberg, S. (2005). Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. *The Journal of the Acoustical Society of America*, *118*(1), 471–482.

Caetano, M. and Rodet, X. (2011). Improved estimation of the amplitude envelope of time-domain signals using true envelope cepstral smoothing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4244–4247.

Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, *35*(3), 283–319.

Choi, S. and Jiang, Z. (2008). Comparison of envelope extraction algorithms for cardiac sound signal segmentation. *Expert Systems with Applications*, *34*(2), 1056–1069.

Deng, J. D., Simmermacher, C., and Cranefield, S. (2008). A Study on Feature Analysis for Musical Instrument Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(2), 429–438.

Elliott, T. M., Hamilton, L. S., and Theunissen, F. E. (2013). Acoustic structure of the five perceptual dimensions of timbre in orchestral instrument tones. *The Journal of the Acoustical Society of America*, *133*(1), 389–404.

Fu, Z., Lu, G., Ting, K. M., and Zhang, D. (2011). A Survey of Audio-Based Music Classification and Annotation. *IEEE Transactions on Multimedia*, *13*(2), 303–319.

Giordano, N., Gould, H., and Tobochnik, J. (1998). The physics of vibrating strings. *Computers in Physics*, *12*(2), 138.

Gouyon, F., Pachet, F., and Delerue, O. (2000). On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00.*

Gregory, A. H. (1994). Timbre and Auditory Streaming. *Music Perception: An Interdisciplinary Journal*, *12*(2), 161–174.

Grey, J. M. (1977). Multidimensional perceptual scaling of musical timbres. *The Journal of the Acoustical Society of America*, *61*(5), 1270–1277.

Grey, J. M. and Gordon, J. W. (1978). Perceptual effects of spectral modifications on musical timbres. *The Journal of the Acoustical Society of America*, *63*(5), 1493–1500.

Handel, S. and Erickson, M. L. (2001). A Rule of Thumb: The Bandwidth for Timbre Invariance Is One Octave. *Music Perception: An Interdisciplinary Journal*, *19*(1), 121–126.

Helmholtz, H. v. and Ellis, A. J. (1895). *On the sensations of tone as a physiological basis for the theory of music.* London, New York : Longmans, Green, and Co.

Iverson, P. and Krumhansl, C. L. (1993). Isolating the dynamic attributes of musical timbrea). *The Journal of the Acoustical Society of America*, *94*(5), 2595–2603.

Kaminskyj, I. and Czaszejko, T. (2005). Automatic Recognition of Isolated Monophonic Musical Instrument Sounds using kNNC. *Journal of Intelligent Information Systems*, *24*(2), 199–221.

Kendall, R. A. and Carterette, E. C. (1993). Verbal Attributes of Simultaneous Wind Instrument Timbres: I. von Bismarck's Adjectives. *Music Perception: An Interdisciplinary Journal*, *10*(4), 445–467.

Krumhansl, C. (1989). Why is Musical Timbre so hard to understand? 43–53. Excerpta Medica.

Lakatos, S. (2000). A common perceptual space for harmonic and percussive timbres. *Perception & Psychophysics*, *62*(7), 1426–1439.

Licklider, J. C. R. (1951). Basic correlates of the auditory stimulus. In *Handbook of experimental psychology*, 985–1039. Oxford, England: Wiley.

Loughlin, P. J. and Tacer, B. (1996). On the amplitude- and frequency-modulation decomposition of signals. *The Journal of the Acoustical Society of America*, *100*(3), 1594–1601.

Makhoul, J. (1975). Linear prediction: A tutorial review. *Proceedings of the IEEE*, *63*(4), 561–580.

Marozeau, J., de Cheveigné, A., McAdams, S., and Winsberg, S. (2003). The dependency of timbre on fundamental frequency. *The Journal of the Acoustical Society of America*, *114*(5), 2946–2957.

McAdams, S., Winsberg, S., Donnadieu, S., Soete, G. D., and Krimphoff, J. (1995). Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes. *Psychological Research*, *58*(3), 177–192.

Meng, Q., Yuan, M., Yang, Z., and Feng, H. (2013). An empirical envelope estimation algorithm. In *2013 6th International Congress on Image and Signal Processing (CISP)*, Volume 2, 1132–1136.

Miller, J. R. and Carterette, E. C. (1975). Perceptual space for musical structures. *The Journal of the Acoustical Society of America*, *58*(3), 711–720.

Moore, B. C. J. (2008). The Role of Temporal Fine Structure Processing in Pitch Perception, Masking, and Speech Perception for Normal-Hearing and Hearing-Impaired People. *Journal of the Association for Research in Otolaryngology*, *9*(4), 399–406.

Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., and McAdams, S. (2011). The Timbre Toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, *130*(5), 2902–2916.

Roebel, A. and Rodet, X. (2005). Efficient Spectral Envelope Estimation and its application to pitch shifting and envelope preservation. In *International Conference on Digital Audio Effects*, Madrid, Spain, 30–35.

Schubert, E. (2006). Does Timbral Brightness Scale with Frequency and Spectral Centroid? *ACTA ACUSTICA UNITED WITH ACUSTICA*, *92*, 6.

Siedenburg, K., Fujinaga, I., and McAdams, S. (2016). A Comparison of Approaches to Timbre Descriptors in Music Information Retrieval and Music Psychology. *Journal of New Music Research*, *45*(1), 27–41.

Smith, Z. M., Delgutte, B., and Oxenham, A. J. (2002). Chimaeric sounds reveal dichotomies in auditory perception. *Nature*, *416*(6876), 87–90.

Steele, K. M. and Williams, A. K. (2006). Is the Bandwidth for Timbre Invariance Only One Octave? *Music Perception: An Interdisciplinary Journal*, *23*(3), 215–220.

Sturm, B. L. (2012). Two Systems for Automatic Music Genre Recognition: What Are They Really Recognizing? In *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM '12, New York, NY, USA, 69–74. ACM.

Sturm, B. L. (2013). Classification accuracy is not enough. *Journal of Intelligent Information Systems*, *41*(3), 371–406.

Turner, R. E. and Sahani, M. (2007). Probabilistic Amplitude Demodulation. In *Independent Component Analysis and Signal Separation*, Lecture Notes in Computer Science, 544–551. Springer, Berlin, Heidelberg.

Turner, R. E. and Sahani, M. (2010). Statistical inference for single- and multi-band Probabilistic Amplitude Demodulation. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 5466–5469.

Turner, R. E. and Sahani, M. (2011). Demodulation as Probabilistic Inference. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(8), 2398–2411.

Vakman, D. (1996). On the analytic signal, the Teager-Kaiser energy algorithm, and other methods for defining amplitude and frequency. *IEEE Transactions on Signal Processing*, *44*(4), 791–797.

Vinet (2002). The CUIDADO Project.

von Bismarck, G. (1974). Timbre of Steady Sounds: A Factorial Investigation of its Verbal Attributes. *Acta Acustica united with Acustica*, *30*(3), 146–159.

Weinreich, G. (1977). Coupled piano strings. *The Journal of the Acoustical Society of America*, *62*(6), 1474–1484.

Wessel, D. L. (1979). Timbre Space as a Musical Control Structure. *Computer Music Journal*, *3*(2), 45–52.

Wilson, B. S. and Dorman, M. F. (2008). Cochlear implants: A remarkable past and a brilliant future. *Hearing Research*, *242*(1), 3–21.

Winsberg, S. and Carroll, J. D. (1989). A quasi-nonmetric method for multidimensional scaling VIA an extended euclidean model. *Psychometrika*, *54*(2), 217–229.

Winsberg, S. and Soete, G. D. Multidimensional scaling with constrained dimensions: CONSCAL. *British Journal of Mathematical and Statistical Psychology*, *50*(1), 55–72.

Winsberg, S. and Soete, G. D. (1993). A latent class approach to fitting the weighted Euclidean model, clascal. *Psychometrika*, *58*(2), 315–330.

Zacharakis, A. and Pastiadis, K. (2016). Revisiting the Luminance-Texture-Mass Model for Musical Timbre Semantics: A Confirmatory Approach and Perspectives of Extension. *Journal of the Audio Engineering Society*, *64*(9), 636–645.

Zacharakis, A., Pastiadis, K., and Reiss, J. D. (2014). An Interlanguage Study of Musical Timbre Semantic Dimensions and Their Acoustic Correlates. *Music Perception: An Interdisciplinary Journal*, *31*(4), 339–358.

Zacharakis, A., Pastiadis, K., and Reiss, J. D. (2015). An Interlanguage Unification of Musical Timbre: Bridging Semantic, Perceptual, and Acoustic Dimensions. *Music Perception: An Interdisciplinary Journal*, *32*(4), 394–412.

Zeng, F.-G., Nie, K., Liu, S., Stickney, G., Del Rio, E., Kong, Y.-Y., and Chen, H. (2004). On the dichotomy in auditory perception between temporal envelope and fine structure cues (L). *The Journal of the Acoustical Society of America*, *116*(3), 1351–1354.

Appendix A

Musical Terminology Glossary

- Amplitude - The amplitude of a wave is the maximum oscillation the wave exhibits measured from the vertical centre of the wave.
- Frequency - The number of cycles a wave exhibits per second. Measured in Hertz (Hz)
- Pitch - The perceived frequency of a note. In western music certain frequencies are labelled according to the chromatic scale consisting of twelve pitches.
- Fundamental Frequency - Typically the strongest and lowest frequency present when a musical note is played and one that corresponds to the note being played on the instrument.
- Harmonics - Musical instruments produce notes made up of a range of frequencies. These include the fundamental frequency and a series of pitches that are multiples of the that fundamental. A harmonic is any pitch in this range including the fundamental.
- Overtone - Typically used interchangeably with harmonic but doesn't include the fundamental frequency.
- A440 - Concert pitch. A440 is used as a standard pitch for tuning where the A above middle C is 440 Hz

Appendix B

Wave Editor Implementation Details

In this appendix I discuss implementation details of the parts of the wave editor that are key to understanding how the application works. Discussion of implementation outside of these elements will be kept to a minimum.

B.1 OpenGL

Everything in the wave editor is rendered using OpenGL. The samples in each wave are converted into vertices where the x-coordinate is their index in the wave and the y-coordinate is the value of the sample. These vertices are stored in arrays, one for each wave, where the first two positions are the x and y coordinates of the first vertex and the second two positions are the coordinates for the second vertex and so on. These arrays are loaded into OpenGL objects called vertex buffers and lines are rendered between each adjacent pair of vertices. These collections of lines form the visual representation of the audio and synthetic waves. The control points are rendered as small squares made up of two triangles.

B.2 Synthetic Wave Generation

The first thing to discuss in relation to the actual running of the application is the generation of the synthetic wave from the single cycle. In the current form of the wave editor the reference audio wave and the single cycle wave files must be supplied by the user. The single cycle must be manually extracted from the steady state of the reference audio wave using an external program. The reasons for this are discussed in section 4.3. Once the wave editor starts and the .wav files are parsed the program

generates a full waveform from the given single cycle. This generation is achieved by repeatedly looping over the array containing the samples of the single cycle and copying the samples to another array. This raises the question of how long to make the synthetic wave. Generating the synthetic wave to be the same size as the audio wave is insufficient as the frequency adjustments will cause the synthetic wave to become too short. This is due to the sharp attack present in recordings of many instruments. When a synthetic wave is generated to be the same size as the audio wave and then is matched to the audio wave the synthetic wave must have it's frequency increased in certain segments and the result is that the wave is compressed in time. This will make the match between the synthetic and audio wave poor as the synthetic wave will no longer be of matching length to the audio wave.

One alternative is to make the synthetic wave long enough that after a reasonable amount of frequency adjustment the synthetic wave won't be shorter than the audio wave and when the synthetic wave is output it is truncated to be the same length as the audio wave. In the current version of the wave editor I make the synthetic wave twice the length of the audio wave. In order to make the synthetic wave shorter than the audio wave the entire synthetic wave would need to have its frequency more than doubled. I highly doubt a correctly played note on any instrument would have an attack that is anywhere close to being more than twice the frequency of the fundamental so twice the length is a safe option. Further still the rest of the wave would be very close to being the same frequency as the single cycle that the synthetic wave is based on so it would never reasonably be a problem. However, this is a hack in order to make the program work.

A more elegant solution would be to have an essentially 'infinite' wave. The concept of having the frequency control points 'anchor' the wave was discussed in section 4.2 and it is important to know that the ends of the synthetic wave are anchored by default frequency control points. The result of this is that when the user adds and manipulates the first frequency control point the frequency is being shifted between those two default control points. The problem with that is that the user might want to start by shifting the whole wave to account for a difference in phase in order to get a closer initial match before performing more fine grained frequency adjustment. The smaller the synthetic wave is then the closer the initial anchor points are and therefore the more significant the effect of a single frequency adjustment. This is because the spacing between samples in the synthetic wave is changed uniformly, so the larger the segment of wave the smaller the frequency adjustment for the same displacement of

the given frequency control point.

By having an 'infinite' wave the problem of how long to make the synthetic wave is solved along with the added benefit of allowing the user to do an initial alignment of the wave without significantly affecting the frequency. To implement this infinite wave the locations of the beginning and end anchoring control points would be set to the minimum and maximum values, respectively, of the data type used to represent them. The actual values for the synthetic wave would only be generated and rendered in the range of time that matches the audio wave. But by having the initial anchoring points so far away from wherever the first control point is placed by the user the result is that any movement of that first point has no significant effect on the frequency of the visible segment of wave. This lack of significant frequency change would allow the user to perform an initial alignment of the synthetic wave that would make the subsequent frequency modification easier as the whole synthetic wave has been shifted to be closer to the audio wave before frequency adjustment begins. Once that initial alignment has been performed the user could then adjust the frequency by placing and manipulating new frequency control points which would have an effect on frequency due to these new points being close to the point used to align the wave. A plausible first adjustment might be to anchor the visible wave at each end in order to make frequency adjustments affect the wave on both sides of newly placed points. For example, the first frequency control point is placed and used to align the synthetic wave as closely as possible to the audio wave. If a new control point is placed the left of that first control point it will change the frequency between that first control point and the beginning default control point. But as the default control point is so far away no significant change in frequency would occur to the left of the newly placed control point. By manually anchoring the synthetic wave at each end after the initial alignment has taken place the user is ensuring that the frequency will be modified on each side of new control points. In my opinion the infinite wave is a more elegant way of representing the synthetic wave and it would be beneficial to the user as well.

B.3 Amplitude Adjustment

Once a synthetic wave has been generated the user can begin to match the wave to the audio wave. Frequency adjustment requires the user to horizontally align the wave so, in my opinion, it makes sense to perform amplitude adjustment first in order to bring the peaks of the synthetic wave to the same level as the peaks in the audio wave.

However, the user can perform amplitude and frequency adjustment at any point.

The amplitude of the synthetic wave is calculated firstly by linearly interpolating between the positions of the amplitude control points to create an amplitude curve. Then each of the samples in the wave are multiplied by the corresponding position in the amplitude curve. Initially the amplitude curve is a straight line as it is calculated from the default amplitude control points. These are at each end of the synthetic wave or more accurately positions (0, 1) and (wavesize - 1, 1).

When a new point is added the synthetic wave needs to be updated so the new amplitude curve is calculated and then multiplied by the synthetic wave. This is achieved using the equation of a straight line $y = mx + c$. So given a pair of amplitude control points the gradient, m, is calculated by dividing the vertical difference of the two control points by the horizontal difference between the two points. The y-intercept, c, is calculated by rearranging the equation for a straight line into the form $c = y - m * x$. Then we calculate the y-value from the x-coordinate of every vertex and save these y-values into an array. Subsequently, when the method to update the vertex buffer is called, each y-coordinate in the vertex buffer is multiplied by the corresponding y-value. This scales the wave according to the amplitude control points.

B.4 Frequency Adjustment

To understand how the frequency adjustment works it is important to understand: how the audio and single cycle are read into the program, how the data is represented, the correlation between the visual waveform and the underlying wave data, and then how the synthetic wave is converted into an audio file once it has been matched to the audio wave.

When the audio wave and the single cycle are read into the program the user must specify the sample rate of the files. The sample rate tells you how many samples there are per second of audio and is required by the library that parses the audio files. The waves are stored contiguously as floating point values in arrays where each element is a single sample. As they are stored in a contiguous array we can think of the 'spacing' between the samples as being 1 as it is not possible to have a fractional index. This representation is maintained when transferring the wave data into vertex buffers so that it can be rendered. For example, the first vertex has an x-coordinate of 0 and the second vertex has an x-coordinate of 1 and so on. These x-coordinates correspond to the sample's index in the array in which they are stored after being parsed. So the

waves in their unmodified state consist of a sequence of values with constant spacing between them.

So how does manipulating frequency control points affect the underlying wave data? Each frequency control point has a field for its position and the index of the closest x-value in the vertex array of the synthetic wave. When a new frequency control point is added by clicking the mouse its position is stored and the nearest x-value is found and its index is also stored. Then when that new point is moved its position is updated but its nearest index remains the same. This can be thought of as tying the frequency control point to a particular part of the wave. Since we know where the frequency control points are and we know which part of the wave they were originally placed at due to the closest index being stored we can calculate how much the wave should compress or extend horizontally in order to maintain the relationship between the frequency control point's position and its stored nearest index.

To illustrate, we start with the the application in its initial state and in the current version of the wave editor this means that the synthetic wave has a default frequency control point at each end. We place a new frequency control point on the wave, and move it to the left. Visually this results in the section of wave to the left of the new control point being compressed and the section of wave to the right of the new control point being extended. This is achieved by altering the spacing between the vertices and we calculate that new spacing according to the control points' positions and nearest indices. Consider the new control point and the default control point to the left. We calculate the number of samples in the section of wave between these control points by subtracting the stored closest index of the left control point from the stored closest index of the new control point. Once we have that number of samples we calculate the distance on the x-axis between the two points by subtracting the position of the left control point from the position of the new control point. Dividing this distance by the number of samples gives us the distance between each sample required in order to fit the segment of wave between the recorded nearest indices into the space between the current positions of the pair of frequency control points. We then update the corresponding x-values in the vertex array so that the distance between them matches the distance we calculated. The same process is performed for each adjacent pair of frequency control points starting at the second control point.

Remember that when the audio and single cycle files are parsed and transformed into vertex arrays the spacing between each sample is 1. This spacing means that, at the sample rate of 44.1KHz, the amount of time between each sample is 1/44100 of a

second. So when we calculate the distance between the positions of the control points we are essentially calculating the amount of time between the control points. And by then changing the spacing between the vertices according to that new amount of time we are changing the amount of time between each vertex. In other words the same segment of wave is taking place over a shorter amount of time. We have increased the number of cycles of the wave that occur in a given time meaning we have increased the frequency.

When exporting the synthetic wave as an audio file, so that it can be listened to, the function responsible takes an array holding the values of the synthetic wave and the sample rate and creates an audio file. This function treats each element in the array as a sample with a constant amount of time between each one that is dependent on the given sample rate. Because we have altered the spacing between each sample in order to adjust the frequency we need to calculate the value of the synthetic wave at each integer position. We do this by iterating through the vertices and finding x-values that are equal to or either side of each integer. If the x-value is already an integer we record the corresponding y-value and continue. If the x-values are either side of an integer we linearly interpolate to get the y-value. It is possible that the linear interpolation could result in a slightly incorrect sample value but the samples are all so close together it is unimportant.