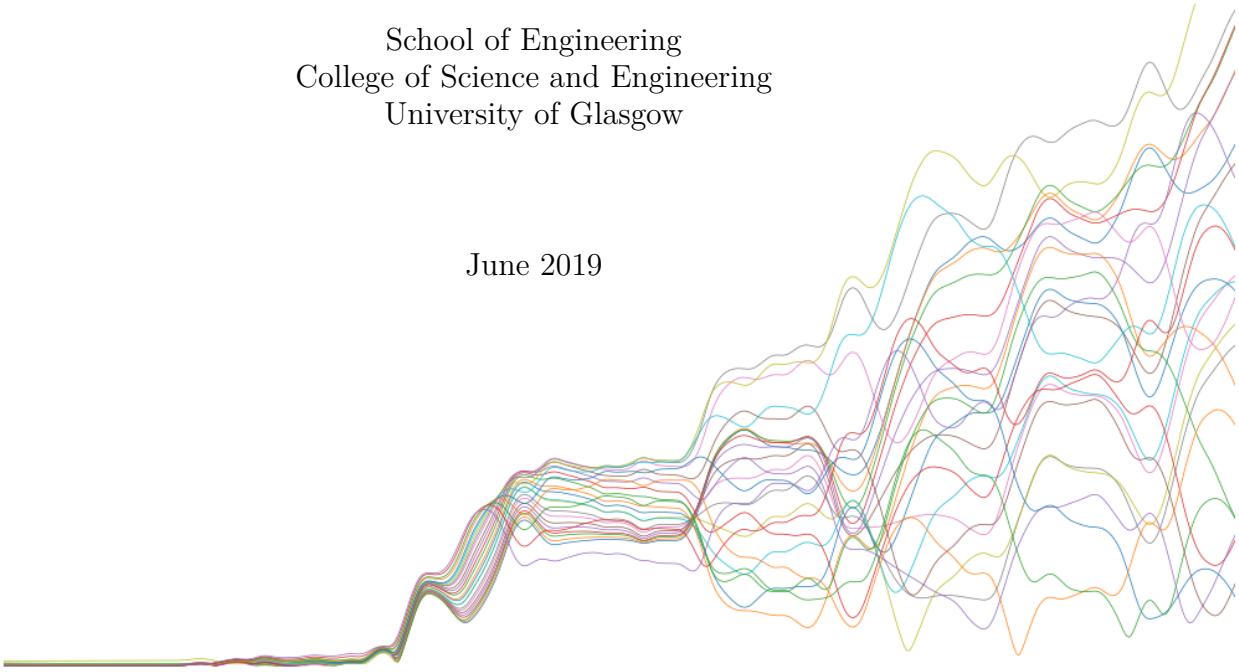
A Novel Technique for High-Resolution Frequency Discriminators and Their Application to Pitch and Onset Detection in Empirical Musicology

Keziah Milligan

Submitted in the fulfilment of the requirements for the degree of Doctor of Philosophy

School of Engineering
College of Science and Engineering
University of Glasgow

June 2019



Abstract

This thesis presents and evaluates software for simultaneous, high-resolution time–frequency discrimination. Whilst this is a problem that arises in many areas of engineering, the software here is developed to assist musicological investigations. In order to analyse musical performances, we must first know what is happening and when; that is, at what time each note begins to sound (the note onset) and what frequencies are present (the pitch). The work presented here focusses on onset detection, although the representation of data used for this task could also be used to track the pitch. A potential method of determining pitch on a sample-to-sample basis is given in the final chapter.

Extant software for onset detection uses standard signal processing techniques to search for changes in features like the spectrum or phase. These methods struggle somewhat, as they are constrained by the uncertainty principle, which states that, as time resolution is increased, frequency resolution must decrease and vice versa.

However, we can hear changes in frequency to a far greater time resolution than the uncertainty principle would suggest is possible. There is an active process in the inner ear which adds energy and enables this perceptual acuity. The mathematical expression which describes this system is known as the Hopf bifurcation.

By building a bank of tuned resonators in software, each of which operates at a Hopf bifurcation, and driving it with audio, changes in frequency

can be detected in times that defy the uncertainty relation, as we are not seeking to directly measure the time–frequency features of a system, rather it is used to drive a system. Time and frequency information is then available from the internal state variables of the system.

The characteristics of this bank of resonators — called a 'DetectorBank' — are investigated thoroughly. The bandwidth of each resonator ('detector') can be as narrow as 0.922 Hz and the system bandwidth is extended to the Nyquist frequency. A nonlinear system may be expected to respond poorly when presented with multiple simultaneous input frequencies; however, the DetectorBank performs well under these circumstances.

The data generated by the DetectorBank is then analysed by an OnsetDetector. Both the development and testing of this OnsetDetector are detailed. It is tested using a repository of recordings of individual notes played on a variety of instruments, with promising results. These results are discussed, problems with the current implementation are identified and potential solutions presented.

This OnsetDetector can then be combined with a PitchTracker to create a NoteDetector, capable of detecting not only a single note onset time and pitch, but information about changes that occur within a note.

Musical notes are not static entities: they contain much variation. Both the performer's intonation and the characteristics of the instrument itself have an effect on the frequency present, as well as features like vibrato. Knowledge of these frequency components, and how they appear or disappear over the course of the note, is valuable information and the software presented here enables the collection of this data.

Acknowledgements

I would like to thank my supervisors, Nick Bailey and Jon Weaver, as well as Graham Hair and John Williamson for their support and insight during this project, the EPSRC for funding it and the various other members of staff at the Univerity of Glasgow who provided help and assistance.

I would also like to thank my friends and family who not only encouraged, supported and/or pretended to be interested, but also put up with me moaning about how much of this thesis is simply the word 'the' (apparently "that's just how sentences work"), amongst other things.

There is a general rule by means of which the musician can obtain the symphony from the score, and which makes it possible to derive the symphony from the groove on the gramophone record, and, using the first rule, to derive the score again. That is what constitutes the inner similarity between these things which seem to be constructed in such entirely different ways. And that rule is the law of projection which projects the symphony into the language of musical notation. It is the rule for translating this language into the language of gramophone records.

— Wittgenstein, *Tractatus Logico-Philosophicus*

Contents

List of Figures

List of Tables

List of Code Extracts

Chapter 1

Introduction

1.1 What is onset detection?

This thesis details an investigation into a novel approach to automatic note onset detection in musical audio.

A musical note can be defined in either a technical context or a cultural one. Technically, a note is characterised by a fundamental frequency and an amplitude; it commences at its onset time. It's fine-grained spectral characteristics and its envelope are related to the instrument of its production.

The fundamental frequency is associated with pitch by cultural convention. The pitch remains constant throughout the note duration, subject to the culturally accepted norms for variation, for example vibrato.

Culturally, the note onset is defined by a simultaneity with the action that produced the note, for example a string being struck or bowed or air being blown into a wind instrument.

Identifying a note onset in a technical context requires the definition of a note boundary. Considering the audio signal from a monophonic instrument, before the onset, the signal contains no energy relating to the note. The onset marks the appearance of energy in the spectrum. Initially, there is a transient phase of the note during which the spectrum is changing rapidly;

after this, the spectrum becomes steady. The duration of the transient depends on the instrument and the technique used to excite it. In the steady state, the signal is not completely stationary; there will always be some variation over the course of the note, as different harmonics decay at different rates.

Onset detection refers to the task identifying a single moment corresponding to the beginning of the note, or, in more general terms, locating the times at which spectral components appear in a signal.

This project focusses on developing software to perform onset detection on monophonic audio signals. Although potential algorithms for this task are evaluated on these terms, it is important to consider that further work may include extending it to incorporate pitch tracking and analysis of polyphonic signals. It may also be desirable to operate in real time. As such, these traits, whilst not central, should not be rendered impossible by the design of the algorithm.

Detecting the onset of a given frequency component in a wideband signal is a problem that arises in many areas of engineering; for example, fault detection in various mechanical systems relies on detecting changes in the spectrum of a signal which correspond to failures in gear teeth or motor components (Benbouzid 2000, Staszewski et al. 1997, Feng et al. 2013, Ghasemloonia & Khadem 2011).

It is also relevant when using empirical methods to analyse musical performance and perception. Fields which undertake such study — empirical musicology and music information retrieval — have to acquire data about many aspects of music (Clarke & Cook 2004). Onset detection provides useful data in studies of musical perception or performance; for example, for calculating the interonset interval, a standard method of measuring rhythm (London 2012). In any experiment to measure a subject's perception of rhythm or meter in music performed by a person (as opposed to generated

by a computer), it is necessary to know the actual onset times with which the subject's responses can be compared. Similarly, we may wish to measure a performer's timing decisions in order to provide insight into interpretation of a score.

The quotation from Wittgenstein (2014) which opens this thesis uses the relationship between music as notated, performed and recorded as an analogy for the relationship between linguistic or mathematical representations of the world and the world itself. They are superficially different representations of the same entity, what Wittgenstein refers to as an "inner similarity" between them. This passage is interesting from the point of view of the central question of this thesis. A trained musician can indeed derive the symphony from the score; the groove on a record can be translated into the sound of the symphony with the correct configuration of mechanical and electrical parts or even from an image of the groove (Li et al. 2009, Räisänen 2017). However, reversing this process and deriving the score from the waveform is not a case of simply "using the [same] rule to derive the score again". There is no simple translation from audio to score. In order to create the audio, a musician first had to interpret the notation, then perform it on an instrument. This introduces a number of complications. Perception is a necessary part of the process: both the performer's perception of the score, which influences their timing and intonation choices, and the audience's perception of the sound.

For example, when listening to a piece of music and trying to work out the time signature, you must first discern whether you are listening to simple or compound time, then decide the number of beats in the bar, which depends on where emphasis is placed. This emphasis comes not only from the rhythm, but also the harmony and melodic shape. Not to mention the additional difficulty of features like polyrhythms or mixed metres, for example, "America", from *West Side Story*, which feels like the time signature is

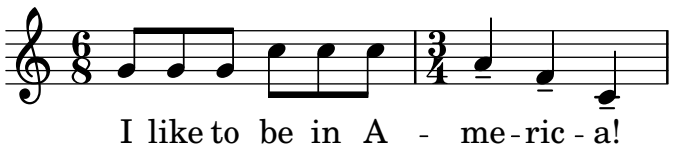


Figure 1.1: Extract from "America", from *West Side Story*, with the shifting meter explicitly marked.

changing from $\frac{6}{8}$ to $\frac{3}{4}$ every bar (see Figure 1.1).

We also must consider the complexity of musical signals; they are rapidly changing and comprise many frequency components alongside the fundamental. It is not controversial to say that computer generated audio of a performance is never as good as a performance by musicians: there is always some qualitative difference.

Sound production by an instrument or voice can be broken down in to discrete stages for the purposes of study; however, in reality there are additional intricacies, which — although small — have a profound effect on our perception of the sound.

For example, the relationship between the tension in a string, the length and the linear mass density tells us the frequency at which it will vibrate when excited. However, if attempting to synthesise the sound of a string, there are many more parameters to consider, like the effect of the finger stopping the string and the force with which it is played. In the case of bowed strings, we must also consider the effect of continuous excitation, resulting in continuously changing parameters over time (Percival 2013).

Similarly, pitched vocal sounds arise when air from the lungs causes the vocal cords to vibrate at a certain frequency. The sound produced by this is then shaped by the vocal tract — the tongue and lips, etc. — as well as the resonant cavities of the throat, mouth and nose. The vocal tract allows different sounds to be articulated and the resonant cavities give rise

to formants in the spectrum (Reetz & Jongman 2009).

All this is to say that musical sounds are not simple, stationary signals. Sounds can generally be split into two categories: pitched or unpitched. However, even a note with a pitch may contain unpitched elements. For instance, we often speak of the transient portion of a note: this is the initial part of the note where the signal is changing rapidly and the pitch has not yet established. It should also be noted that unpitched sounds due to the note production, like a violin bow scraping on the string, can be audible — that is to say present in the audio — but not part of the note itself.

It is, therefore, apparent that a note is not a static entity: the fundamental frequency can vary, along with changes that affect the quality of the sound, rather than the perceived pitch of the note.

These features are absent from a score and occur due to the instrument itself and the musician's intonation. Therefore, data regarding this can only be gleaned by measuring the instrument or the performance. Although measuring instruments is not directly related to this thesis, the software presented here for measurement of performance may well be useful in organology.

The ability to measure performances could provide information that enable us to answer a number of questions that arise in musicology. For instance: What distinguishes different musicians' performances of the same piece? Are there underlying features of a particular musician that appear in many of their performances, like a musical fingerprint? Or does a particular instrument have certain characteristics which distinguish it from similar instruments? In a selection of performances which are all technically correct, what elevates some to great beauty, whilst others are 'merely' good?

Such endeavours should not be limited to music that falls into the western classical tradition. Music outwith this, especially that which comes from oral traditions or which uses different scales, may not be notated or may

be difficult to transcribe accurately within the system largely developed for the western classical tradition. However, audio recordings can be obtained which, at time of collection, are not subject to the interpretation and biases of the person collecting it, unlike transcription.

If we wish to pursue these questions, we must first be able to measure performances and retrieve quantifiable data about small changes in intonation and timing. As stated in Milligan et al. (2016): "What we want to know is not just *that* [musicians] diverge from the literal and mechanical [interpretation of the score], but *when*, *why*, *how* and more." These small deviations from the music as written in the score — or from what may be expected — may contain the key to understanding our perception of music. Indeed, Huron (2006) puts forward the theory that the psychology of expectation is key to understanding our emotional responses to music.

Attempting to analyse music using scientific methods is not a new idea. In his writings on the aesthetics of music, nineteenth century musicologist Eduard Hanslick (1986) wrote that "the striving for as objective as possible a scientific knowledge of things... must necessarily also have an impact upon the investigation of beauty". These writings precede the invention of the phonograph in 1877 (Edison 1878) — the first device capable of both recording and reproduction of sound — which subsequently enabled musicologists to gather with greater accuracy than allowed by transcription. Several composers in the first decades of the twentieth century — for example, Percy Grainger, Béla Bartók and Leoš Janáček — used the phonograph to collect folk music in various European countries, which then informed their own compositions (Ross 2008).

Further developments in sound recording, and the advent of computing, have allowed for more rigorous study. As computers can handle increasing volumes of data at faster rates, we have new opportunities to analyse larger and larger data sets at greater levels of detail.

In order to answer large-scale questions, like those given above, we must first be able to gather information about what is happening and when at the lowest level: the time a note begins to sound (its onset) and its pitch class, as well as what changes occur within the note and when they occur.

Identifying notes in audio by hand is a time-consuming and error-prone process, as it requires a very small subset of samples to be selected as the exact onsets. For example, five minutes of audio, recorded at a standard sample rate of 48 kHz, contains over 14 million samples, of which a few hundred may represent note onsets.

When manually marking-up onsets, there are two possible types of error that may occur. The time chosen may be too early — perhaps we reacted in anticipation of an onset — or too late — maybe the onset was masked by another sound. Generally, sounds will be perceived as simultaneous if they occur within 30 ms of each other (Moore 2012). This gives us the maximum acceptable error when identifying onsets.

An accurate method for automatic onset detection would be a significant gain for the musicologists' toolbox, as it data collection would be both much faster and much more reliable. The ability to analyse large data sets allows more robust conclusions to be drawn.

Before we begin to design software to identify note onsets, we must first consider the defining characteristics of a note.

1.2 What is a note?

A note is the fundamental unit of music; built up to form phrases, which in turn form themes, movements and whole works. However, the question "What is a note?" is deceptively simple, as the parameters of an answer to this question will change depending on the context: a note in a score has different defining characteristics from a note in audio. There are also edge cases which any definition must encompass.

When seen in a score, we may say a note is represented by a dot on the page, the vertical and horizontal location of which determines the pitch and time. Both pitch and time fall into a discrete set of possible values. In aural terms, we can provide a similarly rudimentary definition: a note is an individual sound produced by an instrument. There are infinite possible values for its pitch and time.

It is important to note that pitch and frequency are not the same thing. Frequency is the rate of repetition of a pattern in a signal; pitch is the perception of a note's position in a scale, across the whole range of the given instrument. Pitch class is the set of notes given the same name regardless of octave (Morehead & MacNeil 1992).

There are several features which appear in music like vibrato, glissandi and ornaments which create ambiguity in how the music is perceived: are these intended to be heard as multiple, discrete note events or one continuous note event? (Milligan et al. 2018)

Ornaments (or grace notes) do not change a melody, but add flourishes. In the Baroque era, ornaments were not included in scores — performers would be expected to add them — but are typically notated in more recent traditions. From the perspective of the score, it may be said that ornaments are not notes in their own right. They are attached to other notes and have no meaning on their own: a grace note can be removed from a note, but if the note itself is removed, the grace note must necessarily go with it. However, they do have pitches and times independent of the melody notes, which will be heard when the music is performed.

Vibrato is an example of a feature which occurs in audio, but very infrequently in notation and then only as a general direction. Very wide vibrato could sound like a trill, but only a trill would appear in the score.

In deciding what characteristics to use when defining a note, it may helpful to study existing computerised representations of notes in both audio

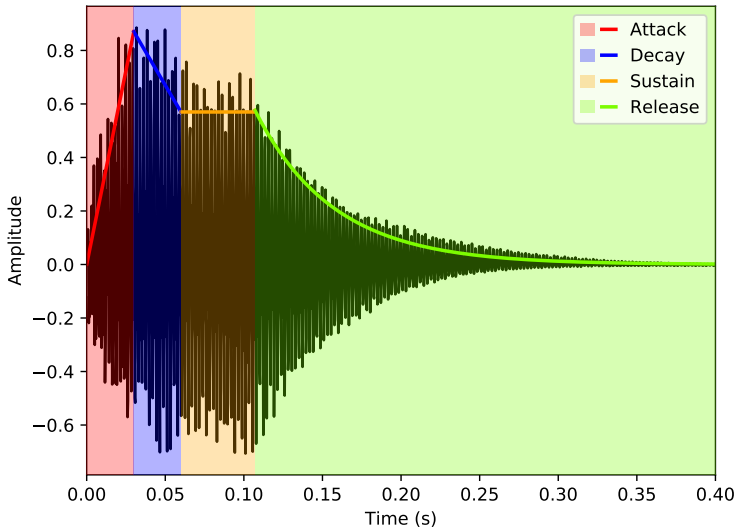


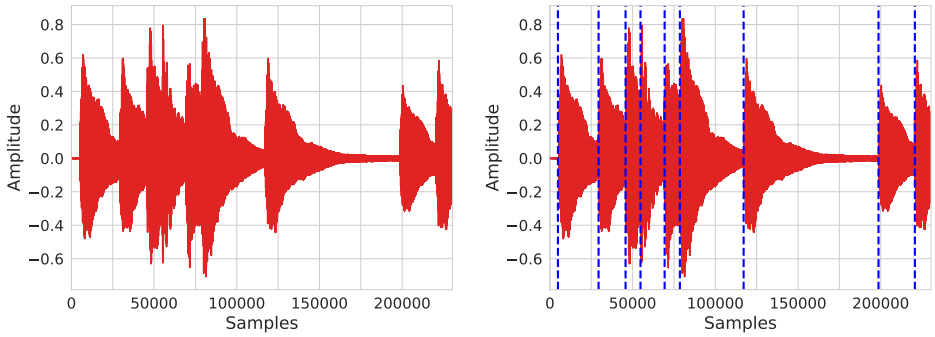
Figure 1.2: Waveform of a note, with ADSR envelope marked

and notation software.

1.2.1 Audio

When analysing recorded audio, the change in amplitude of the waveform a note over time is often described by a four stage envelope: the attack, decay, sustain and release (ADSR). The attack is the initial period when the amplitude of the waveform is rises to its maximum. This is followed by a short decay in amplitude to a steady level, which is sustained while the note continues to be played. Bowed strings, brass, woodwind, piano and voice have the capacity for an extended sustain region; plucked strings and percussive instruments have short sustains which cannot be prolonged. This sustain portion of the note is where we are likely to find significant variation in frequency characteristics. The release occurs as the note ends and the sound fades away. Figure 1.2 shows the waveform of a note, overlaid with the envelope outline.

Note that the attack is not synonymous with the transient. The attack



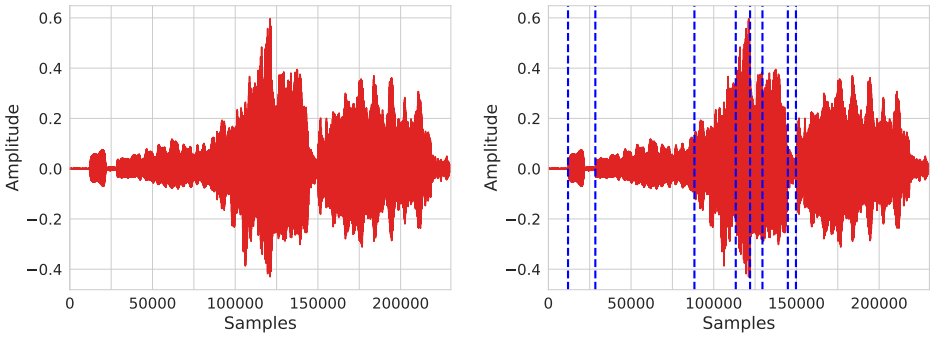
(a) (b)

Figure 1.3: Short piano excerpt, with the nine onsets present marked in (b) by blue dashed lines.

refers to the time in which the amplitude has not yet reached its maximum; the transient is the period before the signal is sufficiently steady for frequencies to occur.

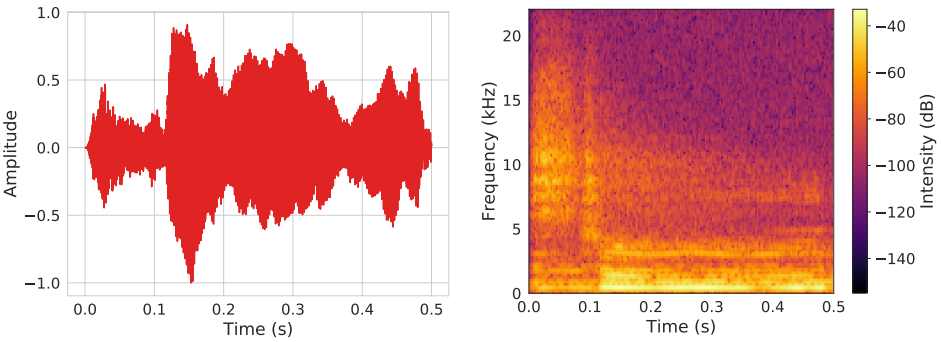
This project is primarily concerned with onsets. The onset time is a single point in time chosen to be the beginning of the note, i.e. the start of the attack. Identifying these can be a difficult task, particularly for instruments which do not have percussive onsets. For example, Figures 1.3 and 1.4 show short excerpts of piano and vocal melodies, respectively. The piano excerpt consists of monophonic staccato notes, therefore the approximate locations of the onsets (see Figure 1.3b) are quite easy to see by visual inspection; the vocal onsets (Figure 1.4b) cannot be identified in this manner. However, it should be noted that, even when the approximate onsets seem clear, as in Figure 1.3a, the process of selecting a single moment — in this case, a single sample — from the thousands that surround it is open to mistakes, as discussed in Section 1.1.

The highly changeable nature of a note, discussed in the previous section, is particularly apparent in onset detection in vocal music, as one note may comprise more than one speech sound. For example, the word ‘stood’, shown in Figure 1.5a sung at an A4 (440 Hz), starts with an unpitched,



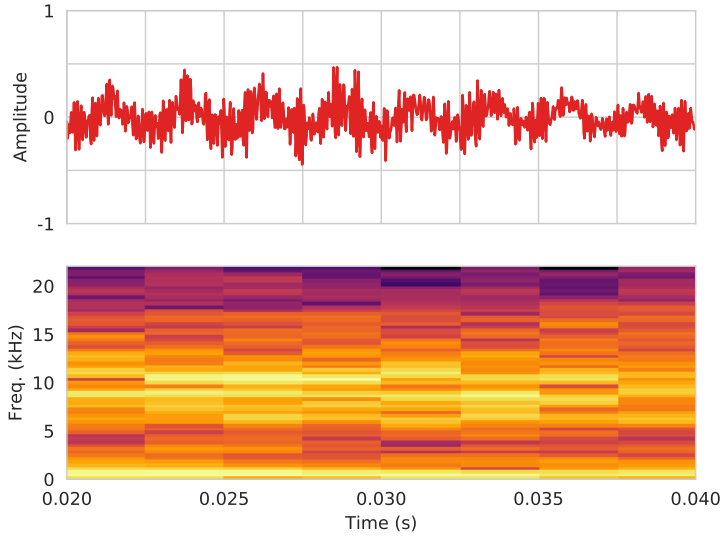
(a) (b)

Figure 1.4: Short vocal excerpt, with the eight onsets present marked in (b)

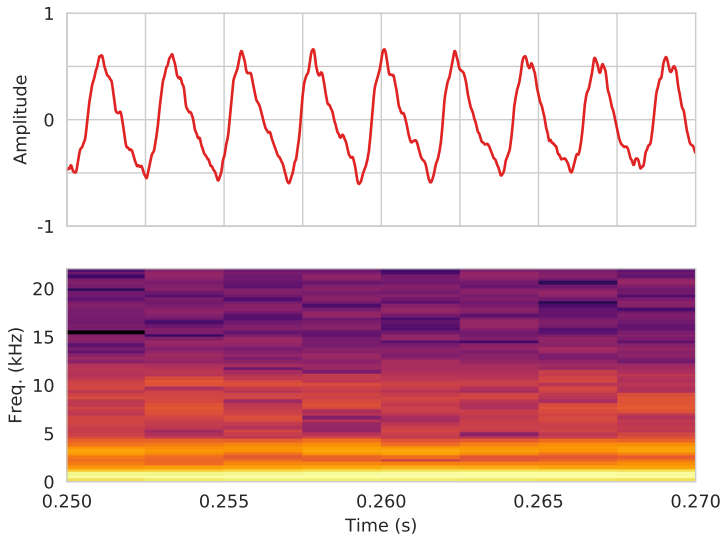


(a) (b)

Figure 1.5: (a) waveform and (b) spectrogram of ‘stoo’ sung at A4 (440Hz).



(a) ‘s’ sound



(b) ‘oo’ sound

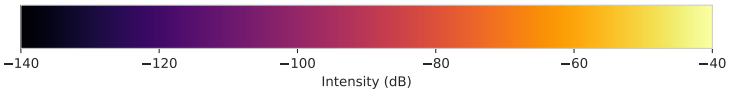


Figure 1.6: Extracts of the waveform and spectrogram in Figure 1.5. (a) shows part of the ‘s’ sound, with the energy spread across the frequency components in the audible range. (b) shows part of the ‘oo’ sound. The signal is periodic here, so most of the energy is concentrated in the low frequency region.

voiceless ‘s’ sound and contains two short, plosive sounds: ‘t’ and ‘d’, although the ‘d’ sound has been omitted by the singer in this example. The ‘oo’ sound is the only periodic, and therefore pitched, sound in the whole word (Roach 2001). This can be seen in the spectrogram, shown in Figure 1.5b. The energy is initially spread across the spectrum, as the ‘st’ sound is sung, before becoming concentrated at lower frequencies when ‘oo’ is sung. Figures 1.6a and 1.6b show the waveform and spectrum of the ‘ss’ and ‘oo’ sounds, respectively. However, the sound extracts are very short (20 ms), which restricts the frequency resolution available in these graphs.[1]

We also encounter the inverse of this problem: one speech sound may also be spread over several notes. In this case (known as melisma), notes may be missed due to lack of a pronounced onset.

We may also encounter problems due to features like reverb — either added as an effect or as a consequence of the environment in which the music is performed — which can partially mask note onsets.

1.2.2 Notation

We will here consider two widely-used, non-proprietary methods of storing and representing note data: MIDI and MusicXML.

MIDI

MIDI (Musical Instrument Digital Interface) (*The Complete MIDI 1.0 Detailed Specification* 2014) was created in the 1980s. Although intended primarily for performance, not notation, the information contained in the some of the messages is instructive. The MIDI messages NoteOn and NoteOff are comprised of three attributes: channel, note and velocity, where each of

[1] The scripts used to create these figures can be found at https://github.com/keziah55/ExtraThesisMaterial/tree/master/Introduction_figures. The parameters used to generate the spectrograms can be found in these files, and the relevant module documentation can be found at https://matplotlib.org/api/_as_gen/matplotlib.pyplot.specgram.html.

these can take integer value up to 127.[2] Whilst 'channel' does not provide useful information here, 'note' and 'velocity' record which note was played and the force with which the key was pressed. From these, and the timings of the NoteOn and NoteOff events, a basic profile of the note can be formed.

MIDI keyboards can be to investigate characteristics of a pianist's timing, as precise note onset times are provided. However, this is not practical for large-scale musicological studies for several reasons, the most obvious being that it can only be used to study piano performance. Additionally, data collection requires the pianist to perform on the keyboard; whilst there are repositories of MIDI performances — for instance, Yamaha have a publicly available collection of MIDI recordings for the Disklavier (*The web's signature MIDI collection* 2015) — the quantity available is slight compared with recorded audio. Lastly, and perhaps most importantly, the mechanical response of digital pianos is different from that of real pianos, a difference which is perceptible to players and so may affect their performance (Fontana et al. 2015).

MusicXML

In the early 2000s, MusicXML was created as a way to share scores (Good et al. 2001). Using XML confers a number of benefits: XML is widely-used, human readable and available to anyone and storing scores in XML databases also enables musicological analysis with tools such as XQuery (Ganseman et al. 2008).

As MusicXML elements relate either to how a note should sound or how it should appear in the score, MIDI renderings can easily be obtained from a MusicXML file.

MusicXML note elements contain pitch information, which itself is com-

[2] The document type definition (DTD) for MIDI messages can be accessed at https://www.midi.org/dtds/MIDIEvents10.dtd.html

Table 1.1: MusicXML elements comprising a `pitch` element

Element	Description
`step`	Diatonic name (A–G)
`alter`	−1 or 1, denoting flat or sharp
`octave`	Number (0–9), where 4 is the octave beginning at middle C

Table 1.2: MusicXML elements which specify intended and notated duration

Element	Description
`duration`	Number which multiplies the `divisions` attribute
`type`	Note type in American terminology (e.g. 'quarter', 'half')

prised of the elements given in Table 1.1.[3]

Changes in pitch over the course of a note can be represented with `glissando`, `slide` and `bend` elements, or ornaments like `tremolo` and `trill-mark`, all of which would also be notated. Aspects of intonation, like vibrato, are not included in MusicXML.

Grace notes do not contain duration information, but regular notes do. This is specified by two elements: `type` and `duration`, shown in Table 1.2. The former gives the symbol to be notated; the latter indicates the length of time (in divisions) that the note is intended to sound for. In some scenarios, the `duration` will differ from what may be expected from the `type`; for example, if the note is to be swung.

`divisions` is an attribute which is specified along with the time signature. It gives a number of divisions of a crotchet which is then multiplied by `duration`.

For example, if `divisions` is 4, the shortest note that can be represented — a note with a `duration` of 1 — is a semi-quaver. To represent a crotchet (or quarter note) in this instance, `duration` should then be 4, as seen in Code Extract 1.1.

`note` elements can also have attack and release attributes, given in terms

[3]The MusicXML note element DTD can be accessed at https://github.com/w3c/musicxml/blob/v3.1/schema/note.mod

```
<attributes>
    <divisions>4</divisions>
</attributes>
<note>
    <pitch>
        <step>C</step>
        <octave>4</octave>
    </pitch>
    <duration>4</duration>
    <type>quarter</type>
</note>
```

Code Extract 1.1: MusicXML extract, representing one note

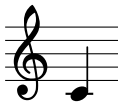


Figure 1.7: C4, as specified by Code Extract 1.1

of divisions, to further adjust the onset and offset times.

Unlike MIDI, onset times are not directly recorded by MusicXML and instead should be inferred from `duration` elements. The pitch information in MusicXML provides context not available in MIDI, which gives only a key number; for example, enharmonic notes are indistinguishable in MIDI, whereas MusicXML is guided by the key signature and needs to be told where in the stave to place the note.

1.2.3 So, what is a note, then?

When considering the question 'What is a note?' we must also think about what information the software could return. Given that we cannot know all possible applications of the software a priori, it should be designed to provide results that may be useful to various users in various scenarios. Potentially useful data should not be discarded because they do not fall into a simple set of categories corresponding to a note.

This section has shown a wide variety of characteristics that indicate or

represent notes in different contexts. For our purposes, despite the fact that we are working with audio, not scores, the typical representation of a note as found in notation software is more suited to our purposes.

The (deliberately loose) characterisation that will be used as a starting point when designing and building this software is:

Definition 1. *A note is an entity with an onset time and pitch, where the onset is a single moment in time which marks the beginning of the note.*

We must also bear in mind that the actual frequencies present can vary over the course of the note. Information identifying these changes — both how the frequency has changed and when — is also potentially useful data.

Any technology would struggle to deal with such complex problems, particularly new technology; however, good software should be able to deal with a range of inputs robustly and should be open to extension.

Software yielding results that adhere to the definition given above must be composed of several parts. The lowest level of this should identify events corresponding to any change in frequency. Classifying events in this data, for example, as one note containing vibrato or an acciaccatura and a note as two separate events, can then happen on a higher level: either by another program or a person. This may mean that, in practical terms, the definition of a note can be broadened to:

Definition 2. *A note is an entity with* at least *an onset time and pitch, where the onset is a single moment in time which marks the beginning of the note. It may also contain intra-note events corresponding to inflections within the note.*

One difficulty of the problem of onset detection is that it requires us to consider several levels of context simultaneously. It is necessary to 'zoom in' and consider a small number of samples to identify the precise onset times, but decisions must be informed by high-level considerations of how the event fits into a wider context.

As such, this project details the design and development of a number of software objects, each of which is specialised to a particular part of the note detection task.

1.3 Content and scope of this project

This thesis presents software for the automatic detection of note event onsets. The remainder of this chapter reviews existing methods of onset detection, before introducing the physiological and mathematical basis for the novel approach to frequency and time measurement at the centre of this thesis.

Chapter 2 discusses the design and implementation of a piece of software referred to as the 'DetectorBank', along with a detailed investigation of its response characteristics. Chapter 3 then details how note onset times can be found by analysing the output of the DetectorBank. A number of possible data extraction techniques are evaluated, the best of which is forms the foundation of an OnsetDetector. The full source code of all the software developed here is available at https://github.com/keziah55/DetectorBank.

The OnsetDetector is tested with a wide range of audio samples, results of which are presented in Chapter 4 and evaluated in Chapter 5. Section 5.3.2 describes how the software could be extended to include pitch data.

Throughout the thesis, short code extracts are provided. The purpose of these is to illustrate algorithms and ideas described in the text. They are presented in simple Python, rather than pseudocode or diagram, to ensure consistency and concision, and to give readers the ability to execute examples directly. Any reader unfamiliar with basic programming language constructs may find Appendix A useful.

1.4 Evaluation of onset detection techniques

1.4.1 Signal processing techniques

There are a number of common signal processing techniques and terms that provide useful background information here. This section provides a brief summaries; detailed descriptions can be found in digital signal processing textbooks, for example Stein (2000).

Fourier transform

The Fourier transform, $X(\omega)$, given in Equation (1.1), is used to find the frequency-domain representation of a function, $x(t)$.

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{1.1}$$

To apply a Fourier transform to a discrete time series, the discrete time Fourier transform (DTFT), Equation (1.2) is used.

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \tag{1.2}$$

In digital systems, it is necessary for signals to be sampled in both the time and frequency domains, and for both input and output to be finite. This is achieved with the discrete Fourier transform (DFT), shown in Equation (1.3).

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{1.3}$$

Energy

The energy in a signal provides a measure of its capacity to influence a physical system. The total energy in a signal $x(t)$ can be found with

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt \tag{1.4}$$

Parseval's theorem gives the relation between energy in the time and frequency domains

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \tag{1.5}$$

From this, it can be seen that the total energy in the signal is the same whichever domain it is considered in.

For discrete signals, this becomes

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \tag{1.6}$$

Causal signals

The Fourier transform, given in Equation (1.1), is non-causal, as it requires integration from negative time. In practice, this is often managed by applying a time delay to the output. This delay is constant at all frequencies.

Another possible solution is to use the Laplace transform, shown in Equation (1.7), which can be used to transform causal signals from the time to frequency domain.

$$X(s) = \int_{0}^{\infty} x(t) e^{-st} dt, \ s \in \mathbb{C} \tag{1.7}$$

In the sampled domain, the z-transform, given in Equation (1.8), can be used to represent causal signals.

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n}, \ z \in \mathbb{C} \tag{1.8}$$

Bandwidth

The bandwidth of a signal is the difference between the minimum and maximum frequencies it contains. In more general terms, bandwidth refers to the range of any band of frequencies.

The term 'wideband noise', used in Section 2.2.9, refers to noise distributed across the range of frequencies. In uniform wideband noise, or white noise, the power is equal across the frequency components.

Sampling theorem

An essential step in digitizing a signal is sampling in the time domain. In order to completely represent a signal with a bandwidth of B Hz, samples must be taken at time steps no more than $1/2B$ seconds apart, i.e. at a rate of $2B$ Hz. This minimum sampling frequency is known as the Nyquist rate.

1.4.2 Review of software

These tests were originally presented and discussed in Milligan & Bailey (2015) and are reproduced here.

Methods

In their paper reviewing onset detection techniques, Bello et al. (2005) define the onset of a note as "a single instant chosen to mark the temporally extended transient", distinguishing it from the attack and the transient.

The output of an onset detector should, therefore, be a series of times identifying the exact moments where each note begins.

Broadly, the approaches to onset detection reviewed here can be split into the following categories:

Energy-based As the transient part of a sound is essentially a burst of white noise, onsets can be detected by looking for energy changes either in the high frequencies or across the whole spectrum (Bello et al. 2005, Brossier 2006).

Spectral-based A musical signal (and its spectrum), being largely comprised of periodic elements, does not vary greatly from one short-time

frame to another. Spectral-based detectors compare the spectra of two consecutive frames — a relatively large difference is taken to signify a new note onset, rather than continuation of the same note (Bello et al. 2005, Brossier 2006, Scheirer & Slaney 1997, Foote & Uchihashi 2001).

Another method involves analysing over frequency bands with a width of one semitone (Pertusa et al. 2005).

Phase-based Tracking how the phase changes with time can be used to identify onsets, as the change from steady-state to transient will be pronounced (Bello et al. 2005, Brossier 2006, Bello & Sandler 2003).

Complex domain Duxbury et al. (2003) propose a method which combines phase and energy techniques, with the aim that the benefits of each will offset the drawbacks of the other.

Sonic Visualiser (Cannam et al. 2010) is a program designed to aid musicological analysis of audio files. Several onset detector plug-ins are available for Sonic Visualiser. Additionally, Böck & Widmer (2013*a*,*b*) have developed two spectral-based algorithms and released them as command line tools.[4] These onset detectors were tested, and the results were then compared with manually marked-up onsets.

As discussed in Section 1.1, the process of manually marking note onsets in audio is susceptible to error. For this analysis, a tolerance of 50 ms was used; that is, if an onset was detected within 50 ms of the manually found time, it was regarded as correct. Recall that, in most circumstances, stimuli will be perceived as simultaneous if they occur with 30 ms of each other (Moore 2012). A wider tolerance was chosen here to account for any human error when manually marking-up the audio.

The onset detectors were tested using recordings of solo voice and solo

[4]These can be found at the following Git repositories: https://github.com/CPJKU/onset_detection and https://github.com/CPJKU/SuperFlux, and were downloaded for these tests on 8 February 2016.

trumpet, as these both have non-percussive onsets and therefore present a more challenging task. For each of these, pieces in both twelve and nineteen equal divisions of the octave (EDO) were used,[5] as a robust onset detector should work for a broad range of music. Each group of test audio comprises three pieces of music and contains an average of 280 notes. Where necessary, the audio files were shortened to no more than 1 minute 30 seconds long. The test pieces used are detailed in Tables 1.3 to 1.6 and the recordings used are listed in the Discography.

Test results and discussion

For each algorithm, the number of correct detections (true positives, TP), erroneous detections (false positives, FP) and missed onsets (false negatives, FN) can be combined as in Equations (1.9) and (1.10) to calculate the precision, P, and recall, R, respectively (Witten et al. 2017) .

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1.9}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{1.10}$$

$\mathrm{TP} + \mathrm{FP}$ gives the total number of detections returned by the algorithm. Thus, the precision provides a measure of how many of the detections are correct.

$\mathrm{TP} + \mathrm{FN}$ is the number of hand-annotated onsets, so the recall tells us how many of the onsets are successfully found.

These two values provide useful information about how well the onset detector performs. They can be distilled into a single number, known as the

[5]Table 1.6 lists the 19 EDO trumpet pieces used. The programme notes for these recordings, available here https://microtonalprojects.files.wordpress.com/2016/12/yasser-programme-notes.pdf, note that one of the chosen pieces, *A Hundred Valleys*, is written in just intonation, resulting in pitches which are "are close to the natural harmonics arising out of each tube length of the 19-tone trumpet".

Table 1.3: Details of 12 EDO soprano voice data set

Title	Composer	Duration	Onsets
Four Late Poems and an Epigram of Rainer Maria Rilke: epigram - I. Idol	Oliver Knussen	01:28	71
Four Late Poems and an Epigram of Rainer Maria Rilke: replica - II. Gravity	Oliver Knussen	01:36	75
Ariel: No. 2. I boarded the King's ship	Jonathan Dove	00:59	114

Table 1.4: Details of 19 EDO soprano voice data set

Title	Composer	Duration	Onsets
Three Songs from the Turkish: 1. Wine	Graham Hair	01:08	87
Three Songs from the Turkish: 2. Ash	Graham Hair	01:04	81
Three Songs from the Turkish: 3. Dance	Graham Hair	01:13	113

Table 1.5: Details of 12 EDO trumpet data set

Title	Composer	Duration	Onsets
Fanfare Abblasen	Gottfried Reiche	00:36	104
Good Night for Trumpet Solo	Luciano Berio	01:13	46
The Big Turtle	György Ligeti	00:44	85

Table 1.6: Details of 19 EDO trumpet data set

Title	Composer	Duration	Onsets
Bye Bye	Alexander Grebtschenko	01:07	153
Melody in 19-division tuning	Michael Parsons	01:25	59
A Hundred Valleys	Michael H. Dixon	01:03	131

Table 1.7: Abbreviations used for each onset detection algorithm in Figures 1.8 and 1.9, derived from the maker and the method.

Label	Maker	Method
aub_cd	Aubio	Complex domain
aub_eb	Aubio	Energy-based
aub_hfc	Aubio	High frequency content
aub_kl	Aubio	Kullback-Liebler
aub_mkl	Aubio	Modified Kullback-Liebler
aub_pd	Aubio	Phase deviation
aub_sd	Aubio	Spectral difference
aub_sf	Aubio	Spectral flux
cpj_od	Böck & Widmer	Spectral-based
cpj_spf	Böck & Widmer	Super flux
qmu_ber	Queen Mary University	Broadband energy rise
qmu_cd	Queen Mary University	Complex domain
qmu_hfc	Queen Mary University	High frequency content
qmu_pd	Queen Mary University	Phase deviation
qmu_sd	Queen Mary University	Spectral difference
uoa_s	University of Alicante	Semitone filterbank

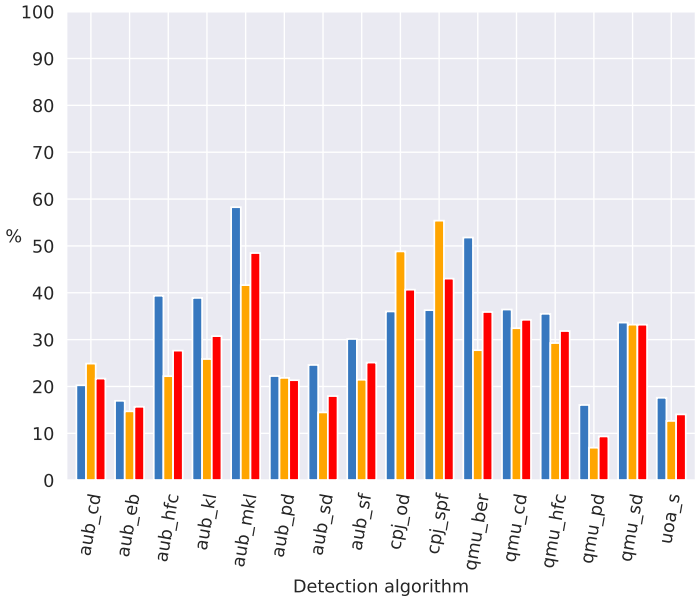
F-measure, F, in a simple fashion

$$F = \frac{2PR}{P+R} \tag{1.11}$$

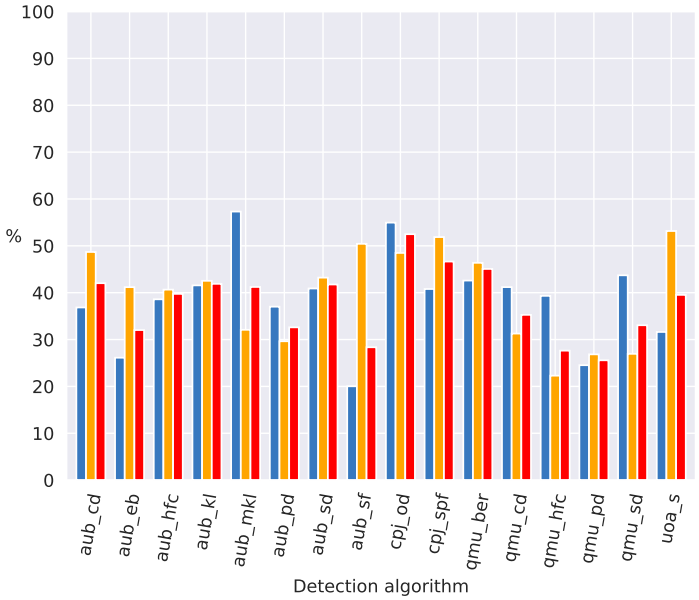
Expressed as a percentage, the F-measure provides a value for the overall effectiveness of the algorithm.

Figures 1.8 and 1.9 shows the precision, recall and F-measure of the algorithms for each test audio group and Table 1.8 summarises the minimum and maximum F-measure for each group in the data set. It can be seen that the instrumentation has a greater effect on the results than the scale, with vocal music being particularly bad. Although the detectors gave significantly better results for the trumpet music, still only one algorithm achieves an F-measure of greater than 90%.

Across all the test sets, detectors based on phase deviation performed badly, as did those employing spectral difference or spectral flux. Kullback-Liebler and modified Kullback-Liebler techniques, which look for increases



(a) Soprano 12 EDO



(b) Soprano 19 EDO

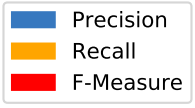
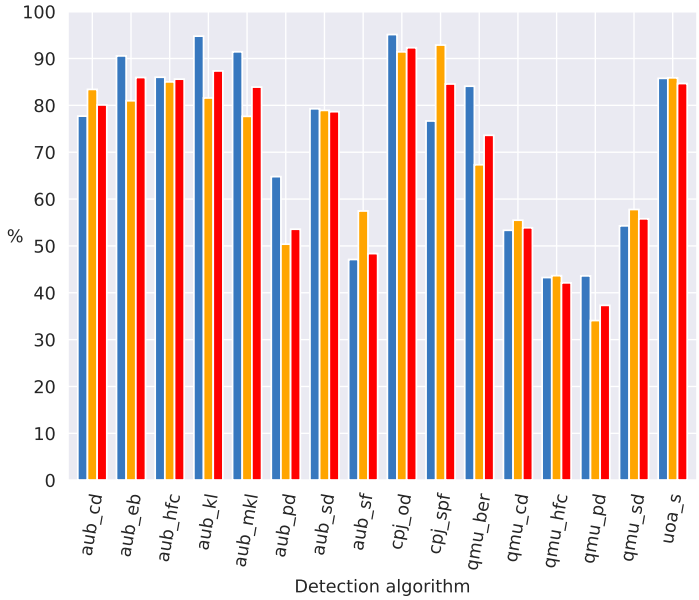
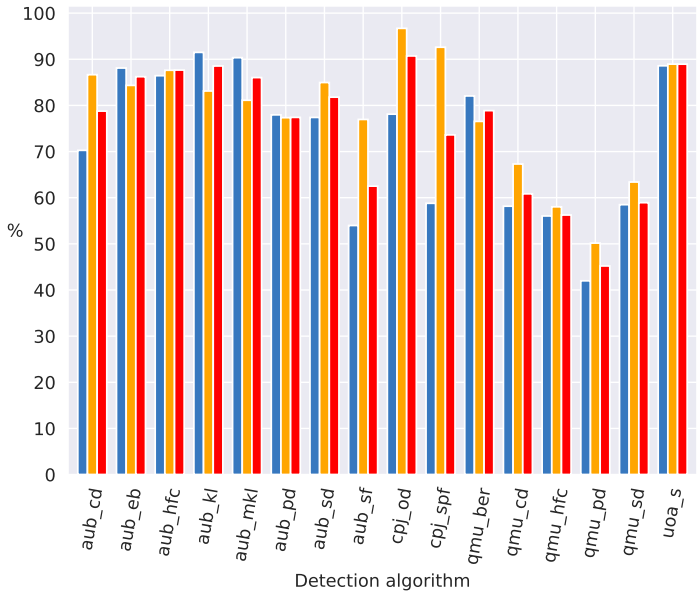


Figure 1.8: Graphs showing the precision, recall and F-measure of each algorithm for both soprano data sets. The keys for each algorithm are detailed in Table 1.7.



(a) Trumpet 12 EDO



(b) Trumpet 19 EDO

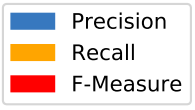


Figure 1.9: Graphs showing the precision, recall and F-measure of each algorithm for both trumpet data sets. The keys for each algorithm are detailed in Table 1.7.

Table 1.8: Minimum and maximum F-measures for each group in the data set, and the algorithm that achieved each.

Group	Min. F-measure, Algorithm	Max. F-measure, Algorithm
Soprano, 12 EDO	9.344%, qmu_pd	48.480%, aub_mkl
Soprano, 19 EDO	25.574%, qmu_pd	52.485%, cpj_od
Trumpet, 12 EDO	37.318%, qmu_pd	92.299%, cpj_od
Trumpet, 19 EDO	45.212%, qmu_pd	90.721%, cpj_od

in energy from one time frame to the next (Brossier 2006), tended to give the best results, along with the spectral-based algorithms detailed in Böck & Widmer (2013*a*,*b*). Aubio plug-ins generally gave better results than those developed by Queen Mary University or the University of Alicante. Interestingly, the semitone filterbank worked just as well for the microtonal trumpet pieces as the 12 EDO ones and considerably better for the microtonal than 12 EDO for the soprano pieces. It may be that it performed badly for chromatic passages in the microtonal, but for the most part, successive notes fell within different bands of the filter anyway.

Somewhat surprisingly, onset detection in vocal music was better for the nineteen EDO pieces than twelve EDO, although this may be due to the twelve EDO test pieces having a larger dynamic range. Adaptive whitening (Stowell & Plumbley 2007) is designed to rectify this by reducing amplitude differences in the spectrum so that quieter sections of audio are not neglected by the detector. This feature is available for the Queen Mary University plug-ins, but even with this, the algorithms do not give accurate results. It is possible that applying dynamic range compression before the onset detection function would improve this.

MIREX (Music Information Retrieval Evaluation Exchange) runs annual competitions in, amongst others things, onset detection. The results

are available online[6], the most recent of which are broadly similar to the experimental results above. In the 2018 results, the average F-measures of the tested algorithms are between 68% and 87%. When broken down by instrumentation, all algorithms perform poorest on solo singing voice — the minimum and maximum F-measures are 17% and 62%, respectively — but better on solo brass instruments, with F-measures ranging from 66% to 94%.

The criteria by which MIREX evaluate onset detection algorithms are similar to those used here: precision, recall and F-measure values are calculated based on the numbers of correct detections, false positives and false negatives; and the tolerance window for correct detections is also ±50 ms. The number of doubled or merged onsets, i.e. two detections for one correct onset or two onsets identified by a single detection time, are also taken into account by MIREX, along with the time taken to analyse the audio, two factors which are not measured here[7].

There are several features of vocal music which can cause the algorithms to fail. These were discussed in Section 1.2 of this chapter. To reiterate, vibrato can give rise to false positives, where changes in energy are wrongly taken to be new notes; melismata can result in false negatives, where the onset of a new note is not clear enough for the detector.

Figures 1.5 and 1.6 in Section 1.2 illustrated the complexity of the spectrum when one sung note comprises multiple speech sounds. This also raises questions regarding exactly where the onset occurs. For the experimental data used here, onsets were manually marked at the beginning of the word or syllable corresponding to that note (i.e. aligning with the score), but there is an argument that the onset should be marked at the beginning of the vowel sound (Sundberg 1994).

[6]Information about MIREX can be found here: http://www.music-ir.org/mirex/wiki/MIREX_HOME and the onset detection results from 2018 can be found here: https://nema.lis.illinois.edu/nema_out/mirex2018/results/aod/index.html

[7]More details on the evaluation procedures can be found on the MIREX website https://www.music-ir.org/mirex/wiki/2018:Audio_Onset_Detection

A robust note onset detector should be able to overcome the problems presented above, just as humans have no trouble with tasks like distinguishing between one note with vibrato and a series of notes or identifying note onsets in quiet passages of music.

1.4.3 Candidate methods for new onset detection software

The Fourier transform — or discrete Fourier transform (DFT), in the case of a digital signal — is a standard tool for translating a signal from the time domain to the frequency domain. It assumes the signal is stationary — that is, it does not change over time — and it requires that the signal be integrated over infinite time. This means that taking the Fourier transform of a non-stationary signal can reveal the spectral components present, but provides no information about the times any frequency appears or disappears.

The short term Fourier transform (STFT) attempts to rectify this by taking the Fourier transform of short segments of signal, thus showing how the spectrum changes over time (Stein 2000). However, frequency and time resolution are inversely proportional to each other: the shorter the time window used, the less accurately frequency within it can be measured and vice versa.

There are several techniques that try to combat this limitation and so would appear to be suitable as the engine of an onset detector: the chirp z transform, constant Q transform and wavelet transform. Each of these will be examined here, but ultimately none can overcome the restrictions described by the uncertainty principle, which will be discussed following this, in Section 1.4.4.

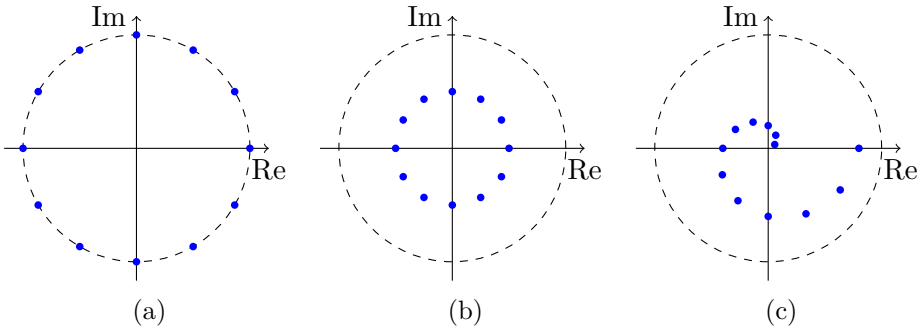


Figure 1.10: Contours (blue points) on which the z transform may be calculated in relation to the unit circle (dashed circle). (a) points on the unit circle, from which the DFT can be calculated, (b) points within the unit circle, and (c) points that spiral out from the centre. The points in (b) and (c) can be used to calculate the CZT.

Chirp z transform

The z transform, given in Equation (1.8) in Section 1.4.1 maps a digital signal $x[n]$ to the z-plane, in the same way that the Laplace transform maps a continuous function $x(t)$ to the s-plane (Rabiner et al. 1969*a*).

When Equation (1.8) is evaluated around the unit circle, $z = e^{j\omega}$, it is equivalent to the DFT; evaluating any other contour yields the chirp z transform (CZT) (Proakis & Manolakis 2007). Figure 1.10 shows various possible points at which Equation (1.8) can be evaluated, relative to the unit circle.

The full derivation of the chirp z transform is beyond the scope of the current investigation, but can be found in the literature.

One advantage of the CZT is that it enables better resolution than the DFT when a narrow band of frequencies is specified by the user. The DFT generates equally-spaced bins from 0 Hz up to the Nyquist rate; with the CZT, upper and lower frequency limits can be defined by the user, along with the spacing of bins in this bandwidth (Rabiner et al. 1969*b*).

At the higher end of the musical scale, the gap between adjacent fundamental frequencies can be large. Therefore, the ability to select frequencies

of interest and the resolution at which they are examined — that is, focussing on the fundamentals and their immediate environs, without wasting resources on the gaps in between — would be valuable when analysing musical signals.

However, DFTs are required to calculate the CZT, so ultimately this technique has the same pitfalls as the Fourier transform.

Constant Q transform

The constant Q transform uses logarithmically spaced, rather than linearly spaced, frequency bins, so that the ratio of centre frequency to bandwidth is constant, hence constant Q (Brown 1991). Therefore, this transform is more suited to musical applications, as the human perception of frequency is logarithmic. A number of studies present techniques for pitch tracking using this transform, for example Brown (1992), Smaragdis (2009) and Fuentes et al. (2012).

It could also be useful for onset detection, as this requires information about frequency components corresponding to musical fundamental frequencies. Simply using the Fourier transform will result in a dearth of data at low frequencies and an overabundance at high frequencies.

In a standard 12 EDO scale, the frequency of any note k steps from a reference frequency f_0 is

$$f_k = 2^{k/12} f_0 \tag{1.12}$$

The quality factor, Q, for each of these steps can be calculated like so:

$$Q = \frac{f}{(2^{1/r} - 1)f} \tag{1.13}$$

$$= \frac{1}{2^{1/r} - 1} \tag{1.14}$$

where f is the centre frequency in question and r is the resolution, i.e. number of divisions per octave. From this, we know that $Q = 16.8$ when

$r = 12$ EDO.

In order to construct a transform where the frequencies are spaced according to this Q factor, we first must consider the short time discrete Fourier transform:

$$X[k] = \sum_{n=0}^{N-1} W[n]x[n]e^{-j2\pi kn/N} \tag{1.15}$$

where $X[k]$ is the k^{th} frequency component, the digital frequency is $2\pi k/N$, $x[n]$ is the n^{th} input sample and $W[n]$ is a window function. Here, the window length, N, is fixed. In order to have greater resolution at low frequencies than high, the window length must change depending on the frequency, so, for the constant Q transform, the window is a function of k and n, i.e. $W[k, n]$, with length $N[k]$.

Equation (1.15) can then be adapted to yield the discrete constant Q transform:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k, n]x[n]e^{-j2\pi Qn/N[k]} \tag{1.16}$$

More computationally efficient implementations have been devised, using the fast Fourier transform (Brown & Puckette 1992) and adaptive bandwidth windowing (Velasco et al. 2011, Holighaus et al. 2013).

Although the constant Q transform presents method of investigating the timing of frequency components in a signal more suited to musical applications, it does not overcome the problem of time resolution and frequency resolution each coming at the cost of the other.

Wavelet transform

The wavelet transform is a multi-scale resolution method of analysing a signal (Nanavati & Panigrahi 2004). Like the chirp z and constant Q transforms, it enables a signal to be analysed with a different resolution at different points.

The problem of time and frequency resolution being inversely propor-

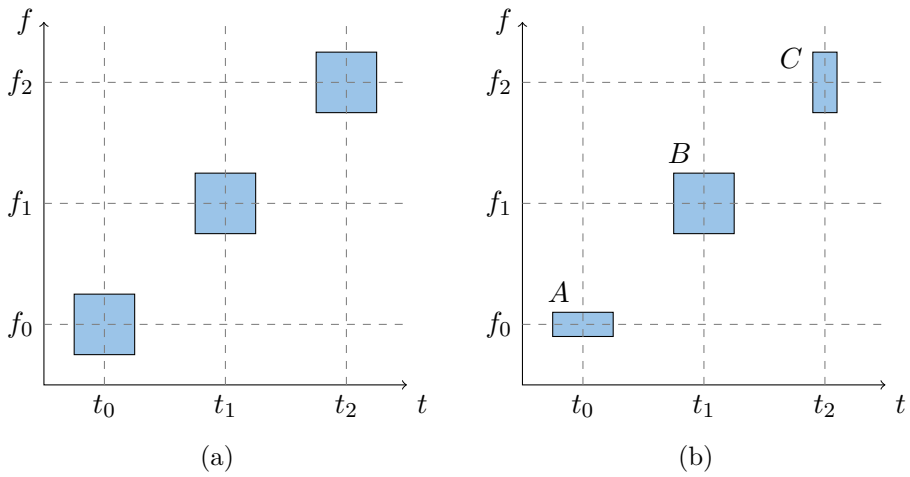


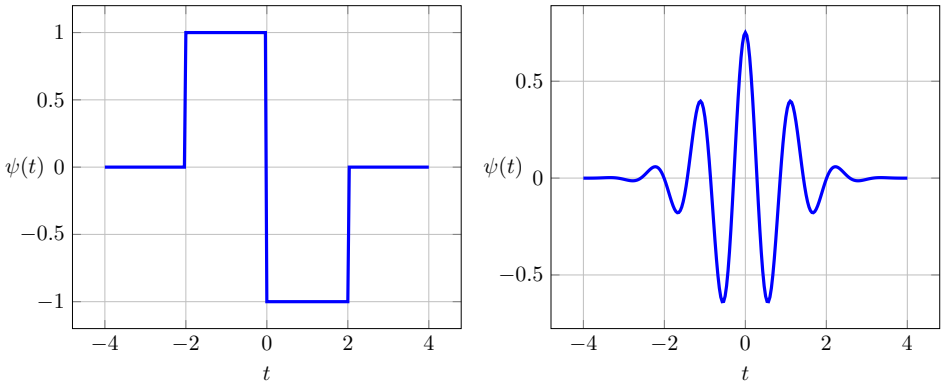
Figure 1.11: Heisenberg boxes illustrating the time and frequency resolution for (a) STFT and (B) wavelet transform. For the STFT, the boxes are the same size at all time–frequency values; the wavelet transform allows different resolutions by adjusting the dilation and location parameters of the wavelet.

tional to each other is illustrated by diagrams known in wavelet theory as 'Heisenberg boxes' (Addison 2002). The resolution of a STFT is determined by the window length, so all of the Heisenberg boxes are the same shape, no matter what the frequency. These are drawn in light blue in Figure 1.11a.

Figure 1.11b shows Heisenberg boxes for the wavelet transform. A high frequency resolution (box A) requires a low time resolution; a high time resolution (box C) reduces the frequency resolution. The wavelet transform allows a signal to be analysed at multiple scales: the resolution is not limited to one value. This gives the wavelet transform an obvious advantage over the STFT, as it can be used to gather information at different scales.

The wavelet transform is the convolution of a wavelet function with the signal. Two commonly used wavelets are shown in Figure 1.12. A wavelet function must have finite energy, no zero frequency component and, for complex functions — like the Morlet wavelet in Figure 1.12b — the Fourier transform must be real and must be zero for negative frequencies.

To implement the wavelet transform, a wavelet family must be selected



(a) Haar wavelet (b) Morlet wavelet (real part)

Figure 1.12: Two common wavelets, (a) the Haar wavelet, a discontinuous function used in the discrete wavelet transform, and (b) the real part of a Morlet wavelet, often used in the continuous wavelet transform.

(e.g. Haar, Morlet, Hermitian etc). The chosen wavelet function is then denoted as:

$$\psi_{a,b}(t) = w(a)\psi\left(\frac{t-b}{a}\right) \tag{1.17}$$

where b is the location parameter (i.e. the area of the time series to analyse), a is the dilation parameter (i.e. the magnification at b) and $w(a)$ is a weighting function. These parameters can be varied to carry out multi-scale analysis and zoom in on different parts of the signal.

The wavelet transform of a continuous signal $x(t)$ is:

$$T_{a,b} = \int_{-\infty}^{\infty} x(t)\psi^*_{a,b}(t)\mathrm{dt} \tag{1.18}$$

To discretize the wavelet transform, the parameters a_0 and b_0 set fixed dilation and localisation steps. The desired dilation and localisation for the wavelet are then set by scaling a_0 and b_0 by integers m and n. This results

in the following discretized form of the wavelet function:

$$\psi_{m,n}[t] = w[a]\psi\left[\frac{t - nb_0a_0^m}{a_0^m}\right] \tag{1.19}$$

$$= w[a]\psi[ta_0^{-m} - nb_0] \tag{1.20}$$

Orthogonal to any given wavelet is the scaling function $\phi_{m,n}(t)$, from which the average of a part of the signal can be calculated. The wavelet function and corresponding scaling function are sometimes known as the mother and father wavelets, respectively.

The procedure for calculating the continuous and discrete wavelet transforms can be found in Addison (2002). For our purposes, it will suffice to state that the multiresolution representation of the signal $x(t)$ can be given as a continuous approximation of itself plus a certain amount of variation, given at an arbitrary level of detail.

Formally:

$$x(t) = \sum_{n=-\infty}^{\infty} S_{m,n}\phi_{m,n}(t) + \sum_{m=-\infty}^{m_0}\sum_{n=-\infty}^{\infty} T_{m,n}\psi_{m,n}(t) \tag{1.21}$$

Here, $S_{m,n}$ represents approximation coefficients, which are calculated by taking a weighted average of the original signal. Combining these with the scaling function gives a continuous approximation of the signal.

The remaining part of Equation (1.21) represents the detail added at the chosen scale m_0. This is calculated by summing the product of the discrete wavelet transform values, $T_{m,n}$, with the wavelet function at all location points n for every dilation level up to m_0.

Whilst the wavelet transform offers a method of changing the resolution at different scales — which is useful for musical applications, given that perception of frequency is logarithmic — it is still the case that, as time resolution increases, frequency resolution must necessarily decrease. This means the wavelet transform is not suitable for detecting the onset of fre-

quency components at a high resolution in both domains.

1.4.4 Uncertainty

The uncertainty principle is most commonly associated with quantum physics, but is found in any wave-based system. Generally, it refers to the impossibility of having simultaneously sharply localised representations of a pair of properties of a function (Ricaud & Torrésani 2013).

When seen in signal processing applications, the uncertainty principle states that as time resolution increases, frequency resolution must necessarily decrease and vice versa (Williams et al. 1991). This can be understood by considering the mechanics of time–frequency measurements. We know that in order to increase the resolution of a frequency measurement, the length of time over which the signal is measured should be increased: in mathematical terms, the Fourier transform requires integration over infinite time. In practice, this means frequency measurement corresponds to a time window, rather than a time instant. Increasing the resolution of the time measurement requires shortening the window over which the frequency is calculated and therefore decreasing the resolution of the resultant frequency calculations.

Gabor (1946) investigates time and frequency analysis of signals and in doing so derives the uncertainty relation between time resolution, Δt, and frequency resolution, Δf:

$$\Delta t \Delta f \geq \frac{1}{2} \tag{1.22}$$

Cohen (1995) applies the concept of standard deviation as a representation of Δt and Δf, although more formally we would refer to the support of the function (Hill 2013). The standard deviation relates to the localisation of the signal in time or frequency. A small standard deviation in the spectral distribution implies a restricted bandwidth and the delivery of the majority of the energy of the signal in a short duration implies a restriction in time.

Uncertainty tells us that when measuring a signal and interpreting it simultaneously in time and frequency, we cannot have finite support in both domains; however, by changing the way in which the signal is observed it is possible to choose between high resolutions in frequency and time, because the uncertainty principle only applies to a single indivisible measurement.

Therefore, it is permissible to filter the signal in such a way as to produce another signal which has a specific time location, but preserves very little frequency information. Of course, the time–bandwidth product of this output signal will be greater than or equal to 1/2. The frequency information can be determined through other means and is not part of the same measurement.

The next section will summarise research into the auditory system. We shall see that perception of time and pitch in sound are similarly precise. This is achieved by physical processes in the human auditory system, rather than by psychoacoustic ones (Gomez & Stoop 2014).

Further sections will then explore using these processes to build the system with the desired properties, and examine how this system can be used to gather time and frequency information from audio.

1.5 A new approach

The software at the centre of this thesis takes a different approach to onset detection than the methods discussed in Sections 1.4.2 and 1.4.3, in order to circumvent the problems that arise due to the uncertainty principle. When designing this software, it was first necessary to consider the auditory system's response to time and frequency information, and the mathematical representation of this process.

1.5.1 Auditory system

Physiology

The physiology of the auditory system is immensely complex and intricate. The following description summarises the details which are relevant to this project. Readers in search of a fuller explanation are directed to Pickles (2012).

For the purposes of study, the auditory system can be split into three parts: the outer ear, middle ear and inner ear. The outer and middle ears are important for sound localisation and efficient transmission of pressure waves to the inner ear. However, they are of little consequence to the present inquiry into time and frequency perception. As reported by Gomez & Stoop (2014), perception of note events occurs in the part of the inner ear known as the cochlea, rather than any other part of the ear or the cerebral cortex. The other part of the inner ear — the vestibular system — is associated with balance, so is also not relevant here.

The cochlea comprises three chambers (scala vestibuli, scala media and scala tympani), separated by two membranes (Reissner's membrane and the basilar membrane). Vibration is transmitted by two membranous openings: the oval window, by which vibration comes in, and the round window, the movement of which allows the vibration to travel through the almost incompressible fluids in the scalae.

The basilar membrane is of particular interest here for two reasons. The first is the way in which it responds to sound stimuli. The second is the oscillation of the outer hair cells, which are receptor cells found in the organ of Corti, a structure which sits on the membrane. These will be discussed in more detail following the next section's summary of cochlear nonlinearties.

Pitch and time perception

As stated in Section 1.2, frequency is the rate of repetition of a pattern in a signal; pitch is a perceptual phenomenon. When we talk about pitch perception in the auditory system, that is exactly what we mean: the auditory system does not perceive individual frequencies in a sound, but a general 'pitch impression'. In some cases, we hear frequencies that are not present in the stimulus.

If we hope to find a mathematical function which describes cochlear behaviour in response to a tone at a given frequency — a cochlear transfer function — we must consider these 'phantom' frequencies.

Residue pitch is a term describing the perception of a missing fundamental frequency: if presented with a series of harmonics with the fundamental, f_0, removed, i.e. a sound composed of $\sum_{k=2}^{N} kf_0$ Hz, the perceived pitch of the sound corresponds to the missing fundamental (Moore 2012).

Combination tones occur when two tones at different frequencies, f_1 and f_2, are presented. In addition to f_1 and f_2, additional frequencies may be perceived at values corresponding to the form $f_1 - k(f_2 - f_1)$, where k is an integer. As frequencies of this form can be generated by a cubic distortion, these are often known as *cubic difference tones* (Smoorenburg 1972). *Difference tones* can also be heard at $f_2 - f_1$, although this cannot be explained by a cubic distortion (Pickles 2012).

These features demonstrate that the cochlear response is nonlinear. This nonlinearity is an essential part of the inner ear; indeed according to Eguíluz et al. (2000), "there is no audible sound soft enough that the cochlear response is linear". However, early experiments into auditory system mechanics were carried out post-mortem. The results of these showed linear responses to sound; it was not until experimentation on living cochleas became possible that the hypotheses explaining the perceived nonlinear phenomena could be physiologically corroborated.

Travelling waves across the basilar membrane in response to sound were first observed in harvested cadaveric cochlea samples in experiments carried out in the 1930s (Olson et al. 2012). The point of greatest displacement of the membrane varies depending on the frequency of the sound, with lower pitches exciting the apex of the membrane and higher pitches at the base. Therefore, the basilar membrane maps frequency.

Indeed, it is often modelled as a series of overlapping bandpass filters, referred to as critical bands, in order to explain phenomena such as masking. This occurs when multiple tones, presented at similar frequencies, are perceived as a single combination tone, rather than two distinct tones at different frequencies. The range of frequencies in which this one will be masked by another is known as the critical bandwidth (Pickles 2012). This concept will be returned to in Section 3.1.2, when designing the OnsetDetector.

The idea that the basilar membrane frequency mapping is the mechanism behind pitch perception is known as the place theory of pitch. However, it cannot explain the sharpness of human hearing: although the smallest detectable change in frequency (known as the difference limen for frequency or just-noticeable difference) varies with frequency and intensity of sound, it is always a fraction of a percent. This is far smaller than the 15% or so (Plack 2012) that place theory would suggest is possible.

The place theory also cannot account for the nonlinearities discussed above, as the residue pitch and combinations tones do not correspond to the area of greatest basilar membrane displacement, or the nonlinearities observed when the basilar membrane is measured in vivo (Rhode 1971, Rhode & Robles 1974).

The other main theory of pitch — the temporal theory — is based on the phase locking of auditory neurons. However, this too has flaws, as phase locking breaks down for sounds over 5kHz and despite this, people can still discern pitch changes above this threshold.

Gold (1948) proposed that there is an active process adding energy in the cochlea which would sharpen the response and thus explain the disparity between the actual capability of human hearing and what was suggested by hypotheses and experiments which assumed, or required, a passive cochlea.

For example, in a healthy living cochlea, the smallest detectable change in sound pressure is 20 μPa. The 'loudness' of a sound is often expressed in terms of the sound pressure level (SPL). This is the ratio of the amplitude of the sound (in Pascals) to this smallest detectable change. Therefore, the lower threshold of hearing is 0 dB SPL; however, in a purely passive cochlea, i.e. one that has been deprived of energy, the threshold of hearing rises to by up to 60 dB SPL (Hudspeth 2008). Furthermore, there is a factor of a million difference between the amplitude of the quietest detectable sounds (0dB SPL) and the loudest tolerable sounds (120 dB SPL), yet this corresponds to only a hundred-fold increase in basilar membrane displacement, from $\pm$0.1 nm to $\pm$10 nm (Hudspeth 2008). Experiments reported by Ruggero & Rich (1991) reveal that a 20 dB increase in stimulus level corresponds to a 20 dB increase in basilar membrane response in a cochlea measured post-mortem, but only a 4.6 dB increase in vivo.

The discovery of both stimulated and spontaneous otoacoustic emissions (Kemp 1978, 1979) due to oscillation of the outer hair cells suggested that the active cochlea theory is correct and energy is added at this stage, causing the response to be nonlinear.

All these phenomena — the accurate pitch perception, the large range of perceivable intensities, the otoacoustic emissions and the nonlinearities — can be explained by the mechanism known as the Hopf bifurcation (Kern & Stoop 2003, Hudspeth 2005, Hudspeth et al. 2010). When the outer hair cells are poised just on the stable side of the critical point of the bifurcation, they amplify, compress and tune the response greatly. On the unstable side, oscillation of the outer hair cells results in otoacoustic emissions.

Recent work by Majka et al. (2015, 2018) has demonstrated that pitch can be perceived in Gaussian pulses which last fractions of a millisecond, a phenomenon which seems to be in direct opposition to what the uncertainty principle tells us. Although Majka et al. consider cochlear nonlinearity to be outwith the scope of their experiments, it is clear that their results demonstrate that the acuity of people's pitch and time perception reaches far beyond the limitations of processes discussed in the previous section.

The research summarised here shows that nonlinearities are essential to pitch and time perception, and that the Hopf bifurcation is an excellent candidate for describing the behaviour of hair cells in the inner ear. This suggests that software which uses the Hopf bifurcation may be able to perform time–frequency detection in audio signals at a higher resolution than the standard linear approaches to this problem.

The investigations presented here will now proceed by studying the Hopf bifurcation, in order to find a form suitable for use as the engine of a bank of damped tuned resonators. The response characteristics of these will then allow simultaneous time and frequency discrimination at a level of precision that the uncertainty principle would suggest is not possible.

1.5.2 Hopf bifurcation

In general terms, bifurcation theory describes changes in a system's topology — for example, stationary points appearing and disappearing or periodic orbits arising — that occur when a parameter passes through a critical value (Wiggins 1990).

Such a system is often presented as a first-order differential

$$\frac{dx}{dt} = f(x, \mu),\ x \in \mathbb{R}^n,\ \mu \in \mathbb{R} \tag{1.23}$$

where μ represents the bifurcation parameter.

A bifurcation can be detected by finding the value of μ for which a

fixed point becomes non-hyperbolic, i.e. the eigenvalues of the Jacobian of Equation 1.23 have no real part (Glendinning 1994).

A Hopf bifurcation (sometimes referred to in literature as the Poincaré-Andronov-Hopf bifurcation) occurs when a system changes from stability to instability as the bifurcation parameter, μ, passes the critical point, μ_0, and the complex conjugate eigenvalues become purely imaginary (Guckenheimer & Holmes 1983, Rosales 2005). In this case, we have a pair of eigenvalues, rather than a single one, so the solutions are periodic orbits around an equilibrium, rather than stationary points (Glendinning 1994). Figure 1.13 illustrates the appearance of periodic orbits that arise from a Periodically forced Hopf bifurcation, which will be discussed later in this section.

The Hopf bifurcation is not only the mechanism behind the action of the outer hair cells in the cochlea, but also a wide variety of systems which exhibit spontaneous oscillation (or self-excitation), for example the appearance of periodic pulsations in detuned lasers (Ning & Haken 1990), the dynamics of neural networks (Song et al. 2005) and food webs in ecosystems (Zhang et al. 2018).

Derivation of Hopf bifurcation normal form

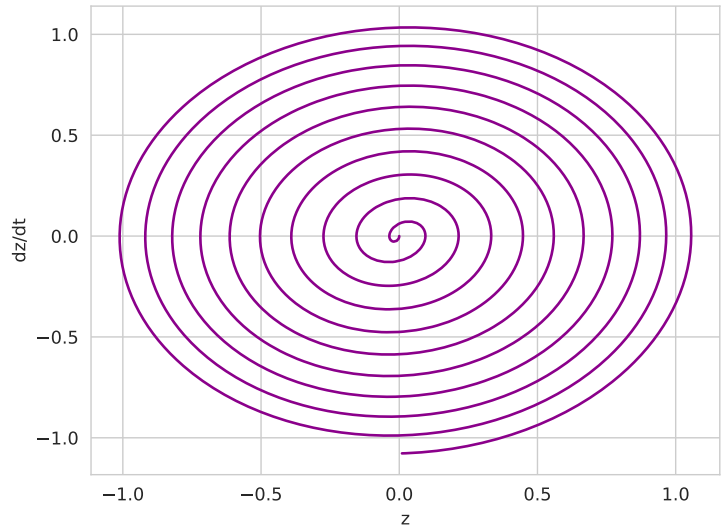
Following the procedures of Kuznetsov (2004), the system of equations

$$\dot{x_1} = \mu x_1 - \omega x_2 - x_1(x_1^2 + x_2^2) \tag{1.24a}$$

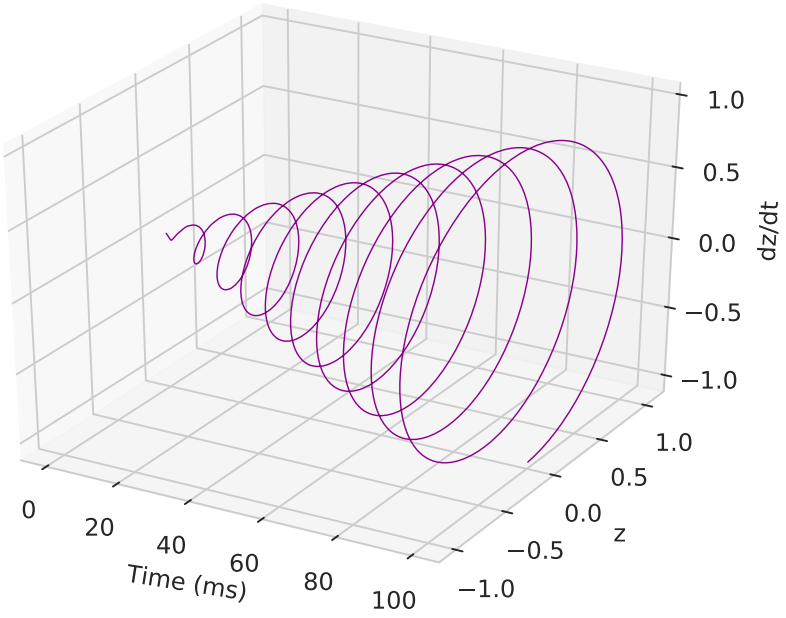
$$\dot{x_2} = \omega x_1 + \mu x_2 - x_2(x_1^2 + x_2^2) \tag{1.24b}$$

can be expressed as

$$\mathbf{f}(\mathbf{x}) = \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} - (x_1^2 + x_2^2)\mathbf{x} \tag{1.25}$$



(a) Emergence of orbits, with potential, z, against momentum, $\frac{dz}{dt}$



(b) 3D rendering of above figure.

Figure 1.13: State space diagrams, showing the growth of periodic orbits, when a resonator operating at a Hopf bifurcation is driven by a sinusoid at a rate equal to the characteristic frequency, ω_0. The bifurcation parameter is positioned at the critical point, the first Lyapunov coefficient is -1 and the momentum, dz/dt, has been scaled by ω_0.
The first Lyapunov coefficient is introduced during the Derivation of Hopf bifurcation normal form and the characteristics of a Periodically forced Hopf bifurcation is addressed later in this section.

where $\mathbf{A}$ is the Jacobian of the system, calculated thus:

$$\mathbf{A} = \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \tag{1.26a}$$

$$\frac{\partial f_1}{\partial x_1} = \mu - 3x_1^2 - x_2^2 \tag{1.26b}$$

$$\frac{\partial f_1}{\partial x_2} = -\omega - 2x_1 x_2 \tag{1.26c}$$

$$\frac{\partial f_2}{\partial x_1} = \omega - 2x_1 x_2 \tag{1.26d}$$

$$\frac{\partial f_2}{\partial x_2} = \mu - x_1^2 - 3x_2^2 \tag{1.26e}$$

When $x_1 = x_2 = 0$, $\dot{x_1} = \dot{x_2} = 0$ for all μ and ω (by (1.24)) and the system is in equilibrium. Therefore

$$\mathbf{A} = \begin{bmatrix} \mu & -\omega \\ \omega & \mu \end{bmatrix} \tag{1.27}$$

Observing the eigenvalues of the system, equivalently the poles of its transfer function, determines not only the stability of the system, but also the frequency of the periodic solutions. As μ tends to μ_0, the angular frequency of the orbits tends to ω_0. Solving to find the eigenvalues of Equation (1.27) gives

$$\lambda_{1,2} = \mu \pm j\omega \tag{1.28}$$

To find the normal form of the Hopf bifurcation, we introduce the complex variable

$$z = x_1 + jx_2 \tag{1.29}$$

from which follows

$$z^* = x_1 - jx_2 \tag{1.30a}$$

$$zz^* = |z|^2 = x_1^2 + x_2^2 \tag{1.30b}$$

$$\dot{z} = \dot{x_1} + j\dot{x_2} \tag{1.30c}$$

Combining eqs. (1.24), (1.29) and (1.30) gives the normal form of the Hopf bifurcation

$$\dot{z} = (\mu + j\omega)z - z|z|^2, \ z \in \mathbb{C} \tag{1.31}$$

where the positive eigenvalue (1.28) appears as the coefficient of z. The real part of the coefficient of the cubic term determines the stability of the orbits.

This cubic coefficient is known as the first Lyapunov coefficient and will be denoted b, where $b = -1$ in the above equation. b is a real value, the sign of which determines the stability of the solutions (Guckenheimer 2008). In the case $b < 0$, the bifurcation is supercritical and the periodic orbits are stable; when $b > 0$ there is a subcritical bifurcation and the orbits are unstable, therefore Equation (1.31) represents a supercritical Hopf bifurcation.

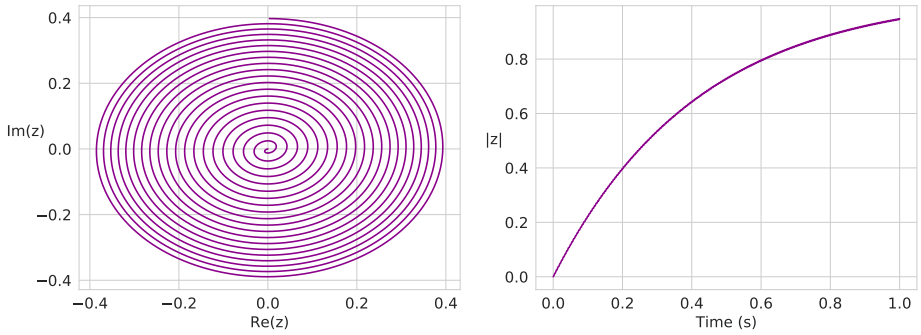
Explicitly including the first Lyapunov coefficient gives a new form of the Hopf bifurcation

$$\dot{z} = (\mu + j\omega)z + b|z|^2 z, \ z \in \mathbb{C} \tag{1.32}$$

Periodically forced Hopf bifurcation

Per Section 1.5.1, we wish to have a form of the Hopf bifurcation which represents the response of the outer hair cells to an input. When including this in the expression, it takes the form of a sinusoidal forcing function F.

For a Hopf resonator at the bifurcation point, $\mu = \mu_0$, tuned to a characteristic frequency ω_0 with forcing frequency ω_{in} (Zhang & Golubitsky 2011)



(a) Complex response (b) $|z|$ response

Figure 1.14: Output of Hopf system with characteristic and forcing frequencies both at 100 Hz. (a) first 200 ms of the response in the complex plane, (b) absolute value of response over 1 second.

$$\dot{z} = (\mu_0 + j\omega_0)z + b|z|^2 z + F \tag{1.33}$$

where $F = Xe^{j\omega_{\text{in}}t}$ or, in a real system, $F = Xcos(\omega_{\text{in}}t)$.

As was stated at the beginning of this section, a condition of the Hopf bifurcation is that, at the bifurcation point, the eigenvalues lie on the imaginary axis, therefore $\mu_0 = 0$. Therefore, Equation (1.33) can be simplified to

$$\dot{z} = j\omega_0 z + b|z|^2 z + F \tag{1.34}$$

If the forcing frequency ω_{in} is equal to the characteristic frequency of the system ω_0, periodic orbits with angular frequency ω_0 will appear. For example, Figure 1.14 shows the response of a forced system with a real input, where $\omega_0 = \omega_{\text{in}} = 200\pi$ rad s^{-1} and $X = 25$. Figure 1.14 (a) shows the first 200 ms of the response in the complex plane as the periodic orbits are being established, and (b) shows the absolute value of z over 1 second.

Constructing a bank of these resonators tuned to a range of characteristic frequencies and analysing the $|z|$ responses of each will allow those frequencies to be detected in the signal.

Degenerate Hopf bifurcation

The Derivation of Hopf bifurcation normal form introduced the first Lyapunov coefficient and its effects on the stability or instability of the solutions. If the first Lyapunov coefficient vanishes, i.e. $b = 0$, a degenerate bifurcation occurs. This is also known as a Bautin or generalised Hopf bifurcation (Kuznetsov 2004).

$$\dot{z} = (\mu + j\omega_0)z, \; z \in \mathbb{C} \tag{1.35}$$

The nonlinear term disappears, but there are still periodic orbits, as the limit cycle 'degenerates' into the plane at $\mu = 0$ in (x_1, x_2, μ)-space. This creates a relatively simple equation for a degenerate bifurcation exhibiting a periodic solution:

$$\dot{z} = j\omega_0 z \tag{1.36}$$

and for a forced degenerate bifurcation:

$$\dot{z} = j\omega_0 z + F \tag{1.37}$$

1.5.3 Summary

As we have seen, the auditory system's response to sound can be described by the Hopf bifurcation, where the sound input takes the role of the forcing function.

Equation (1.34) can now be used at the basis for a new approach to automatic onset detection, one which is not subject to the same limitations on simultaneous time and frequency measurement as existing methods. The following chapters will discuss the development and results of this software.

It should be noted that, while the auditory system provides important background, the purpose of the software is not to model the auditory system, but to detect notes in musical audio.

Chapter 2

Building the DetectorBank

As discussed in Chapter 1, the behaviour of the Hopf bifurcation in response to audio will form the basis of the onset detection software developed here.

This chapter details the design and implementation of the DetectorBank: a bank of nonlinear tunable resonators (detectors), each of which operates at a Hopf bifurcation. The characteristics of this system are then explored, and new features implemented to overcome deficiencies.

Unlike the many attempts to solve time and frequency detection problems which are based on conventional signal processing techniques, our approach to this problem sidesteps uncertainty-related issues: we do not seek to measure the signal directly, instead the signal is used to drive a bank of tuned resonators. Frequency and time information can then be found from observable internal state variables, without the limitations imposed by the uncertainty principle.

The DetectorBank source code is available at https://github.com/keziah55/DetectorBank and additional Python scripts with which the DetectorBank was tested can be found at https://github.com/keziah55/ExtraThesisMaterial.

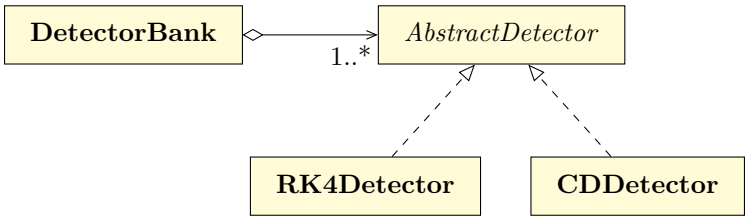


Figure 2.1: UML diagram of the DetectorBank and detectors

2.1 Implementation of the Hopf bifurcation

2.1.1 Structure

Per UML terminology (Fowler 2004), a DetectorBank is an aggregation of AbstractDetectors (the base class for a detector). A simplified UML diagram showing the DetectorBank and detectors is given in Figure 2.1. Each detector implements the expression for a forced Hopf bifurcation, as given in Equation (1.34), using either the central difference approximation or the fourth order Runge-Kutta method (see Section 2.1.2).

Two potential drawbacks of the idea of multiple detectors are the resultant resource and memory usage. The output of a DetectorBank comprised of k detectors, each of which operates on N input samples, will be a $k \times N$ array. Requesting more detectors or more input samples increases both the time a DetectorBank will take to run and the memory required to store the output array, which could potentially exceed the available RAM.

The effect of the first of these problems can be mitigated by incorporating multithreading. As each detector operates independently of all others — the DetectorBank is embarassingly parallel — they can be run in multiple concurrent threads, and utilising all available CPUs on a computer will reduce execution time.

There are two methods provided by the DetectorBank which utilise multiple threads: `getZ()`, which calculates the result of the Hopf bifurcation, and `absZ()`, which calculates the absolute value of the `getZ()` output.

Table 2.1: Ratios of time taken when one thread is used, T_1, to time with four threads, T_4, for the two multithreaded methods provided by the DetectorBank

Method	T_1/T_4
`getZ()`	2.895
`getZ()` and `absZ()`	2.532

Table 2.1 shows the speed increase measured when testing these multithreaded methods on a computer with Intel i5-4210M CPUs, which operate at 2.6 GHz and are capable of running four concurrent threads.[1] 12 GB of RAM were available; there was no swap usage, so none of the time taken can be attributed to transferring data to virtual memory.

The second problem is averted by the DetectorCache object. This fills a fixed number of segments with DetectorBank output. After the last segment is filled, the first segment is re-used. This not only keeps the memory usage under control, but also provides a mechanism by which results could be generated in response to a continuous live input stream.

2.1.2 Realisation

As Equation (1.34) cannot be discretized, numerical approximations are used in the implementation. The software allows users to choose between the central difference approximation and the fourth order Runge-Kutta method. These are implemented as subclasses of the AbstractDetector class: CDDetector and RK4Detector. These methods, and their implementation in software, are presented in Appendix C.

Throughout the investigations in this thesis, the Runge-Kutta method is predominantly used, as it is known to be well-conditioned when applied to the Hopf bifurcation (Christodoulou 2008).

[1] The full specification can be found at https://ark.intel.com/content/www/us/en/ark/products/81012/intel-core-i5-4210m-processor-3m-cache-up-to-3-20-ghz.html

Effect of numerical methods

The central difference approximation is significantly simpler than the Runge-Kutta. This reduces the time taken to execute it, but also reduces the accuracy of the results.

When the detectors are operating at a degenerate Hopf bifurcation, both of these numerical approximations introduce errors, where, at high frequencies, the detector frequency and the input frequency can appear to be mismatched. Adjusting the characteristic frequency of the detector can mitigate this. For example, when the fourth order Runge-Kutta method is selected, a detector nominally operating at 2 kHz requires its frequency to be shifted by 0.066%; a 3 kHz detector requires frequency adjustment of 0.25%.

A method for frequency normalisation — referred to as search normalisation — is provided. This iteratively searches for the detector frequency which provides the optimal response at a given input frequency and automatically adjusts the characteristic frequencies of the detectors accordingly, thus allowing the software to be used over a wider range of frequencies. These errors arise due to the numerical methods and there is no analytical expression for search normalisation.

These effects are not seen in the responses of non-degenerate detectors, as the widening bandwidth (as discussed in Section 2.2.5) masks any such errors.

The threshold up to which the responses do not require normalisation is higher when the Runge-Kutta method is used than central difference, however central difference method typically performs about three times faster than the Runge-Kutta.

Damping factor

When Equation (1.34) is implemented in software and sinusoidal forcing is applied, the periodic orbits increase in magnitude to a maximum. When the

forcing is stopped, the orbits remain at this magnitude. In order to cause the system behaviour to relax, a damping factor is introduced, which scales each output z value.

This is included in the implementation described in Appendix C; and its effect on bandwidth and time response is investigated in Section 2.2.6.

2.2 Empirical investigation of DetectorBank characteristics

In order to characterise this system, we must measure the bandwidth and time response of a single detector, the maximum bandwidth of the DetectorBank and the effect of varying the forcing amplitude. It is also desirable to determine how a detector behaves when its characteristic frequency is presented in the presence of others. As the system is nonlinear, each of these attributes cannot be analysed in isolation: the effect on the whole system must always be considered.

The following investigations are presented with musical applications in mind, so typical audio sample rates are used and the frequency range of interest covers the range of fundamental frequencies in music (27.5 Hz to approximately 4.2 kHz) and the range of human hearing (20 Hz to 20 kHz).

The control parameter μ is set to zero (i.e. at the bifurcation point). Unless otherwise stated, the first Lyapunov coefficient is also zero so that the detectors are operating at a degenerate Hopf bifurcation; a damping factor of $1 \cdot 10^{-4}$ and forcing amplitude $X = 5$ are used. These values produce an output which generally falls within the range $|z| \leq 1$. As will be discussed in Sections 2.2.5, 2.2.6 and 2.2.8, the system turns out to be well-conditioned when these parameters are altered.

As seen in Figure 1.14b of Chapter 1, plotting the absolute value of z gives a line clearly showing the growth of the periodic orbits. Henceforth,

"the detector response" will refer to the absolute value of z over time. It is important to consider not only the magnitude of the response, but also its shape, as this provides more information about how the system is reacting.

Both the fourth order Runge-Kutta and central difference methods are used, with and without search normalisation, over a range of frequencies and with a range of sample rates (48 kHz, 96 kHz and 192 kHz). The test audio comprises sine waves generated at various frequencies, either a single tone or a series of tones. All end with a period of silence, so the response in relaxation is also visible. Using signals of this form has the advantage that the precise frequencies are known, so a poor response for a certain frequency can be attributed to the detector, not the input. However, a pure sine wave has neither the complexity nor variation found in musical audio.

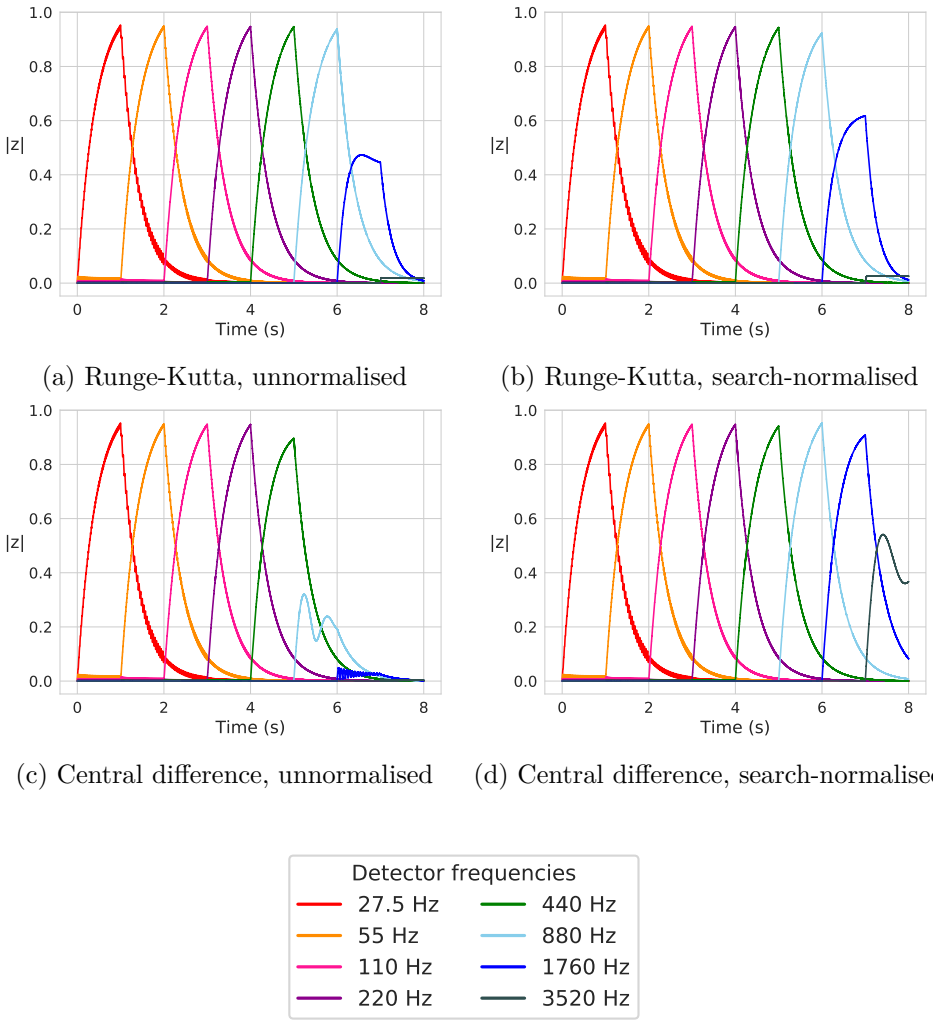
2.2.1 Responses to frequencies across musical range

This section details the system responses at selected frequencies covering the range of fundamentals found in music and seeks to develop a general impression of the performance one can expect from the DetectorBank. The following sections in this chapter investigate individual aspects of the system, as well as discussing methods of compensating for any deficiencies in the responses.

Octaves

Figure 2.2 shows the response to eight consecutive tones, each lasting for one second. The frequencies ascend by octave from 27.5 Hz to 3520 Hz (i.e. A0 to A7 in music). The sample rate of the audio is 48 kHz. It can be seen that, for Runge-Kutta detectors, search normalisation somewhat improves the 1760 Hz response, but the effect on the 3520 Hz detector is negligible. The central difference detectors are much improved by search normalisation.

Small oscillations are noticeable in all the low frequency responses. Fig-



(a) Runge-Kutta, unnormalised

(b) Runge-Kutta, search-normalised

(c) Central difference, unnormalised

(d) Central difference, search-normalised

Figure 2.2: DetectorBank responses to consecutive tones, lasting one second each and rising by octave from A0 to A7. Eight detectors are used, tuned to the corresponding frequencies, from 27.5 Hz to 3520 Hz. The sample rate used here is 48 kHz.

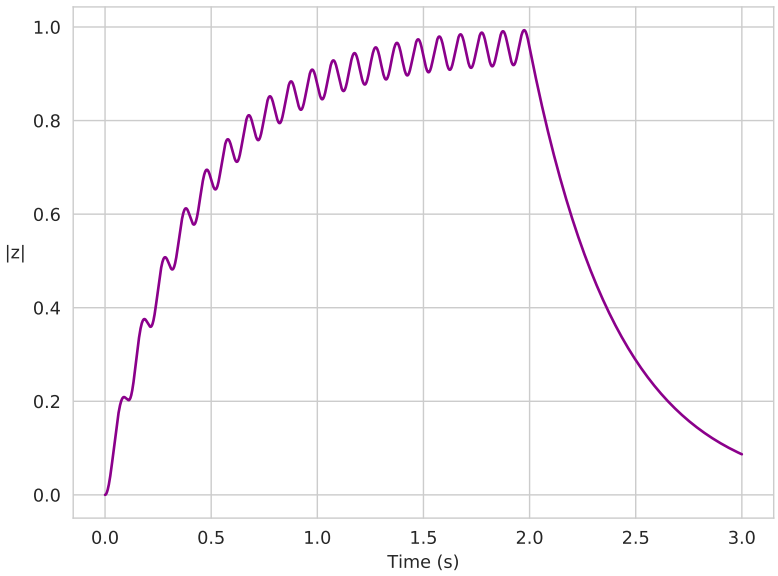
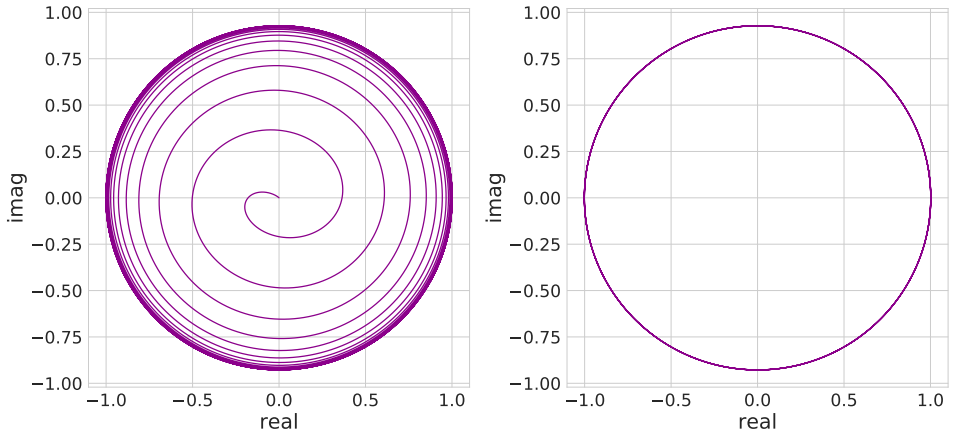


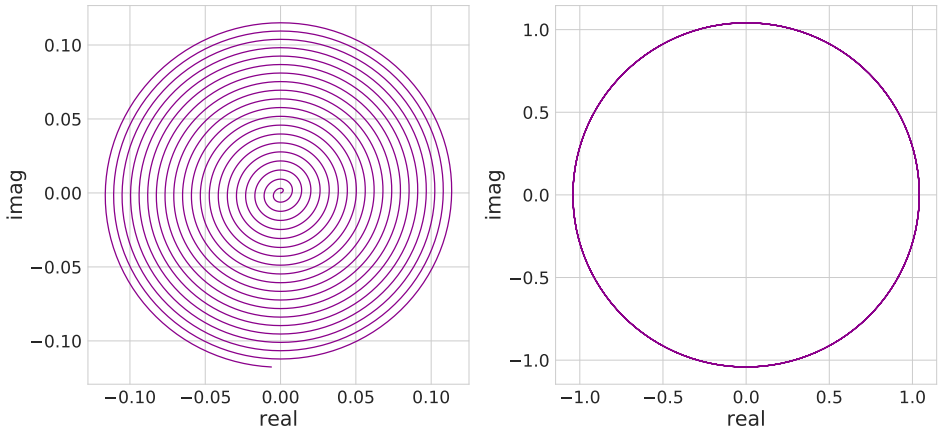
Figure 2.3: Detector response to 5 Hz tone



(a) First 20 periods

(b) Last five periods

Figure 2.4: The (a) beginning and (b) end of the complex response at 5 Hz



(a) First 20 periods (b) Last five periods

Figure 2.5: The (a) beginning and (b) end of the complex response at 400 Hz

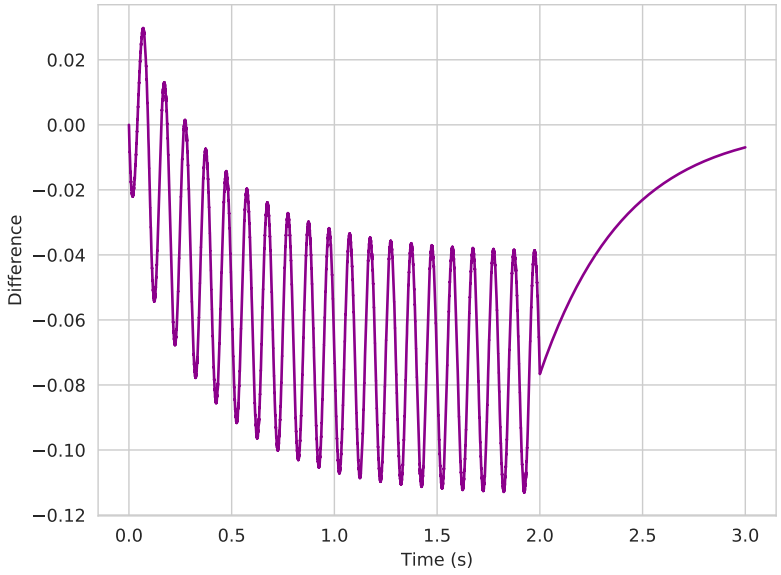


Figure 2.6: 5 Hz response, as shown in Figure 2.3, with a 400 Hz response subtracted, leaving the 10 Hz oscillations.

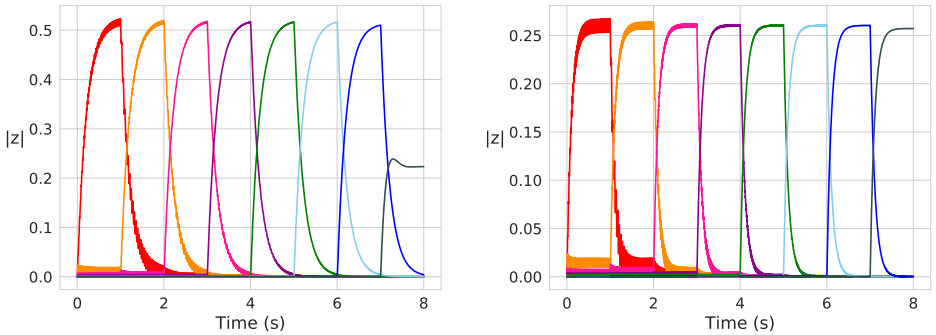
ure 2.3 provides a clearer representation of this, showing the response to a 5 Hz tone, which lasts for two seconds. These oscillations are at twice the driving frequency and arise because, as can be seen in Figure 2.4, the response in the complex plane is not quite circular, but slightly elliptical: it is slightly wider at two points and slightly narrower at two points, hence the frequency-doubled oscillations. This is more pronounced at lower frequencies, as can be seen by comparing Figure 2.4 with Figure 2.5, with the result that the oscillations at higher frequencies are much smaller in magnitude.

The plots of the final five periods in response to the tone (Figures 2.4b and 2.5b) show this difference particularly clearly. A circle should have an eccentricity, e, of zero; for an ellipse, $0 < e < 1$ (Brannan et al. 2012). Measuring the eccentricity of the final periods of the responses yields results which get closer to zero as the frequency is increased: $e = 7.622 \cdot 10^{-2}$ for the final periods of the 5 Hz response; by 400 Hz this has dropped by two orders of magnitude to $e = 8.878 \cdot 10^{-4}$.

In Figure 2.6, the response to a 400 Hz tone has been subtracted from the 5 Hz response shown in Figure 2.3, leaving the prominent oscillations of the 5 Hz response. Measuring the time between these oscillations yields a frequency of 9.974 Hz. As the duration of the tone is increased, this measurement reaches 10 Hz.

One of the functions of the amplitude normalisation feature, which will be introduced in Section 2.2.8, is to correct this orbital eccentricity and reduce the oscillations.

Figure 2.7 shows the eight octave response at two higher sample rates (the responses for unnormalised Runge-Kutta detectors are shown; central difference detectors did not give noticeably different results). It can be seen that oscillations in the lower frequency responses become more pronounced at higher sample rates.



(a) $f_s = 96\,\text{kHz}$ (b) $f_s = 192\,\text{kHz}$

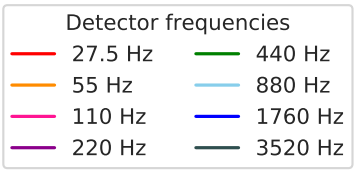
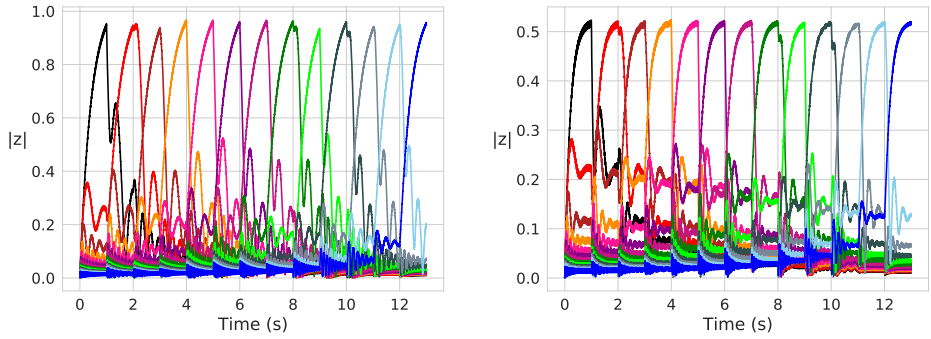


Figure 2.7: Responses of unnormalised Runge-Kutta detectors to consecutive tones, lasting one second each and rising by octave from A0 to A7. The detectors are tuned to the corresponding frequencies, from 27.5 Hz to 3520 Hz. The sample rates are (a) 96 kHz and (b) 192 kHz.

Low frequencies

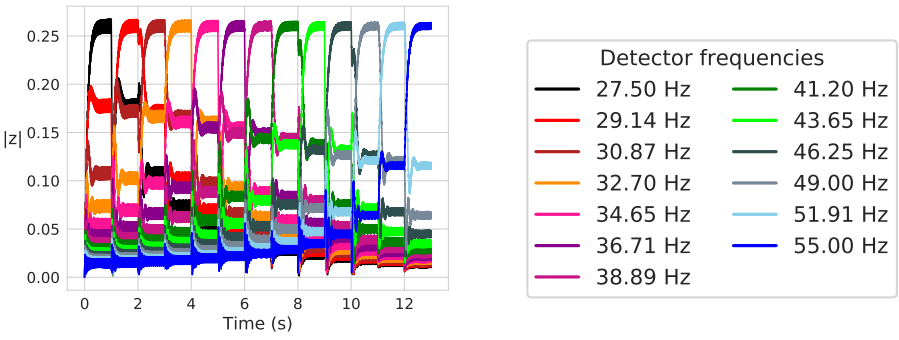
Figure 2.8 shows low frequency responses at the three sample rates. The graphs shown are only Runge-Kutta detector output, as again there was no discernible difference between this and central difference. All the detectors are used without normalisation. The audio input spans a chromatic scale in the lowest octave on a piano (i.e. from A0 to A1, 27.5 to 55 Hz). The responses at low sample rates are more desirable, in that, although they have a slower rate of response, they have better rejection of neighbouring semitones. For example, at all sample rates, when the 27.5 Hz tone sounds, the 29.1 Hz detector also reacts briefly, however the maximum amplitude of the 29.1 Hz detector is 8.5 dB lower than that of the 27.5 Hz detector when the sample rate is 48 kHz and only 2.3 dB lower when $f_s = 192\,\text{kHz}$.

This rejection of neighbouring frequencies also happens very quickly.



(a) $f_s = 48\,\mathrm{kHz}$

(b) $f_s = 96\,\mathrm{kHz}$



(c) $f_s = 192\,\mathrm{kHz}$

Figure 2.8: Responses to a chromatic scale in the lowest octave on a piano. Again, each tone lasts for one second, and the detectors are tuned to the corresponding frequencies. The tones were generated at three sample rates: (a) 48 kHz, (b) 96 kHz and (c) 192 kHz.

Again, in the case of the 27.5 Hz tone, the 29.1 Hz detector begins to react at the same time, but quickly reaches a maximum and stops responding. At $f_s = 48$ kHz, this difference of 1.6 Hz is being discriminated in about 265 ms. When the sample rate is increased to 96 kHz, then 192 kHz, this time resolution drops to 247 ms, then 230 ms. These are all significantly quicker than the 312.5 ms that would be expected from the definition of the uncertainty relation given in Gabor (1946), which was stated by Equation (1.22):

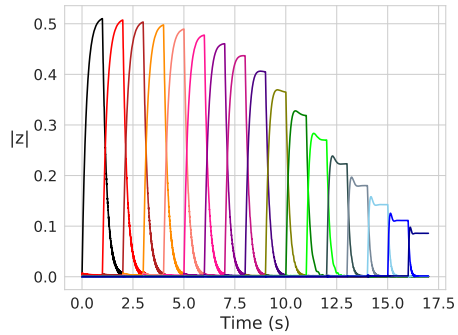
$$\Delta t \Delta f \geq \frac{1}{2}$$

At the three sample rates tested here, $\Delta t \Delta f = 0.433$, 0.404 and 0.375.[2]

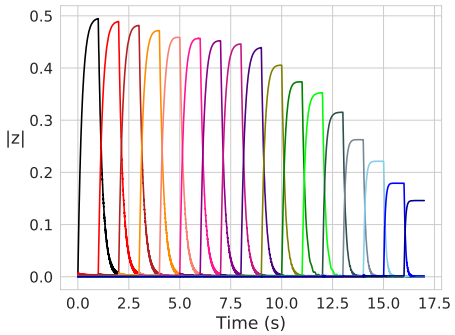
Given the work of Majka et al. (2015, 2018), as discussed in Pitch and time perception in Section 1.5.1 of Chapter 1, it should come as no surprise that a system based on the mechanism of the inner ear can achieve results that surpass what is deemed possible by the uncertainty principle when deriving the time and frequency from a single observation.

As will be discussed fully in Section 2.2.4, when a detector responds to a driving frequency which is close to, but not the same as, its characteristic frequency, the response will oscillate at a rate of the difference between these frequencies. These oscillations are distinct from the small amplitude oscillations which occur due to eccentricity in the complex orbits. When analysing these difference oscillations, one might expect the local maximum to occur at half the period of oscillation; however, a time shift is observed due to the initial rate of change of the response, which results in a shorter time to maximum. Various frequency differences and their corresponding time shifts will be presented in Table 2.3 of Section 2.2.4.

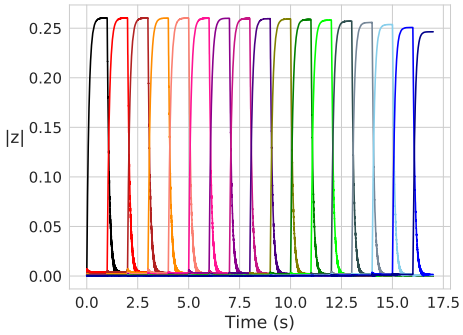
[2] Of course, the reader will recall that the uncertainty principle applies only when Δt and Δf refer to the same observation (as would be the case in an attempt to derive the sub-band signals from, for example, a DFT). In this case, Δf is a property of the detector, not the signal under measurement.



(a) Unnormalised; $f_s = 96\,\text{kHz}$



(b) Search-normalised; $f_s = 96\,\text{kHz}$



(c) Unnormalised; $f_s = 192\,\text{kHz}$

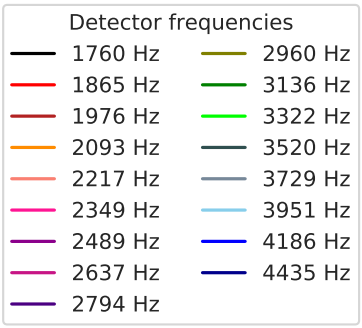
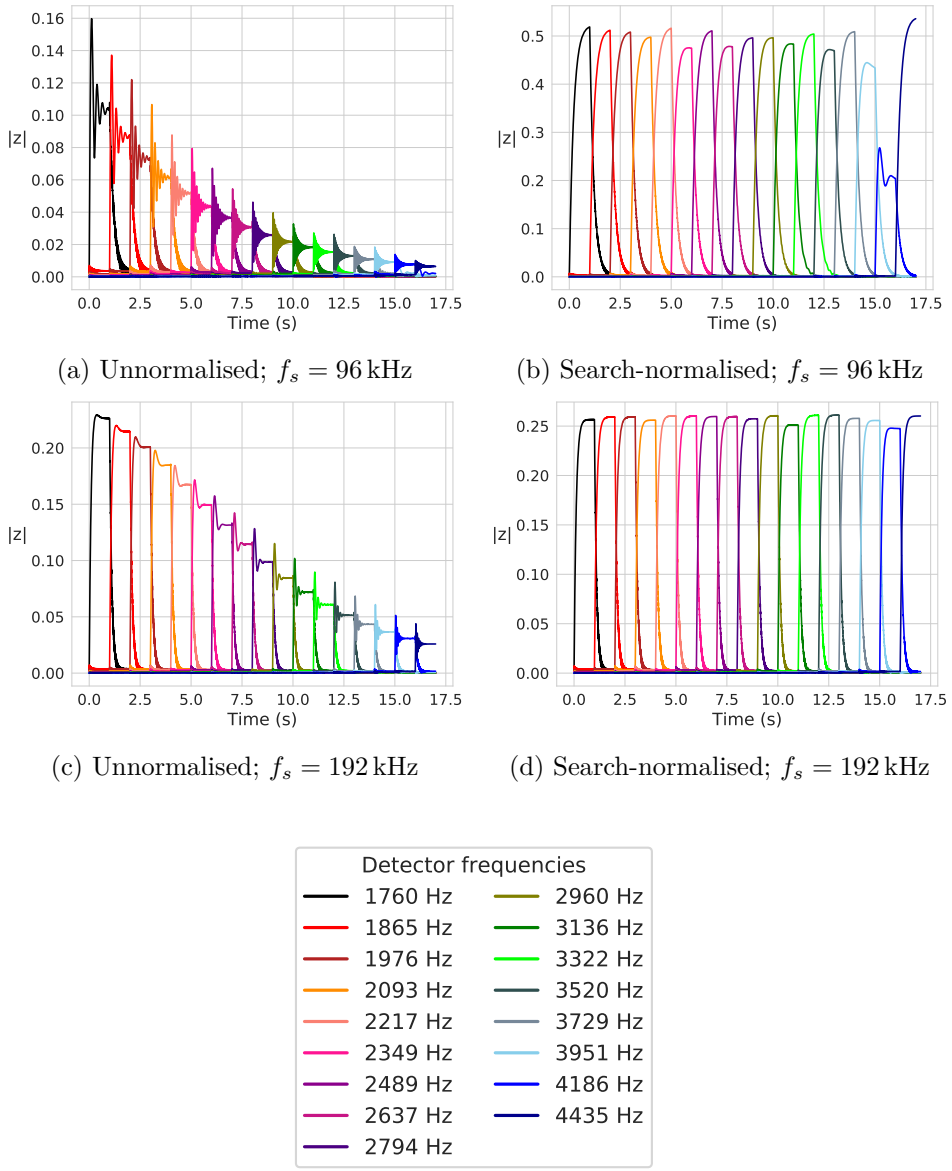


Figure 2.9: Runge-Kutta detector responses to tones generated at frequencies corresponding to the highest 17 notes on a standard piano (A6 to C♯8), with the detectors tuned accordingly. (a) and (b) show results when $f_s = 96\,\text{kHz}$; it can be seen that search normalisation improves the responses somewhat. In (c), at the higher sample rate of 192 kHz, unnormalised detectors give consistent responses.

High frequencies

Figures 2.9 and 2.10 look at higher frequencies for Runge-Kutta and central difference detectors respectively. Only higher samples rates are shown here, as the frequencies being tested are out of the range for which the detectors at lower sample rates can adequately respond. At 192 kHz, no normalisation is necessary for Runge-Kutta detectors: they respond well to all frequencies up to the limit of fundamental frequencies found in music. When the sample rate is 96 kHz, the shape of the responses of unnormalised Runge-Kutta detectors becomes distorted above about 3 kHz. With search



(a) Unnormalised; $f_s = 96\,\text{kHz}$

(b) Search-normalised; $f_s = 96\,\text{kHz}$

(c) Unnormalised; $f_s = 192\,\text{kHz}$

(d) Search-normalised; $f_s = 192\,\text{kHz}$

Figure 2.10: Responses to the same input as in Figure 2.9, but using central difference detectors. At both sample rates, 96 kHz and 192 kHz, the responses, (a) and (c), are much improved when search normalisation is used, as seen in (b) and (d).

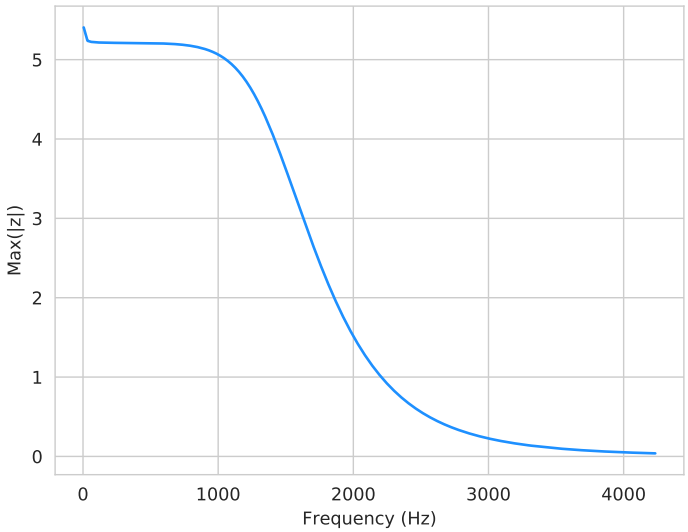


Figure 2.11: The maximum response amplitude decreases as detector frequency increases. Values calculated here used an unnormalised Runge-Kutta detector, with input samples generated at $f_s = 48\,\text{kHz}$.

normalisation, the range can be extended to 4 kHz. For both normalised and unnormalised detectors, the amplitude of the responses decreases as the frequency increases. The distorted shape is due to the detector's characteristic frequency not quite matching the input frequency; decreasing amplitude means the detector is not responding as strongly, although the frequency may still be correct. As we have seen, the first of these problems can be mitigated with frequency normalisation, as described in Section 2.1.2; the second suggests amplitude scaling may be necessary. This will be introduced in the next section.

The central difference detectors (Figure 2.10) do not respond as well at high frequencies. For either of the two higher sample rates normalisation is required, although the responses of the frequency normalised detectors at $f_s = 96\,\text{kHz}$ are somewhat erratic.

2.2.2 Amplitude scaling

As seen in Figures 2.9 and 2.10 of Section 2.2.1, as the characteristic frequency and input frequency of a detector are increased, the amplitude of the responses decreases. Despite this amplitude decay, the shape of the responses is retained, therefore this is not the result of distorted frequencies: the detector response is weaker, but equally as sharp, at higher frequencies.

Maximum response values for various characteristic frequencies were found for all detector types at a sample rate of 48 kHz. The results for unnormalised Runge-Kutta detectors can be seen in Figure 2.11. Using these values to scale up the detector responses creates a consistent output over a larger range of frequencies.

2.2.3 System bandwidth

Section 2.2.1 investigated how the DetectorBank responds to frequencies across the range of fundamentals typically found in musical audio. However, there may be applications where it is instructive to apply the DetectorBank to frequencies outwith this range; for example, analysing signals range of audible frequencies (20 Hz–20 kHz).

The figures in Section 2.2.1 — for example, 2.2, 2.9 and 2.10 — show that a uniform output can only be obtained up to a few kilohertz, depending on the conditions used to create the DetectorBank. Although the point at which the responses begin to deteriorate can by raised by changing parameters like the sample rate, numerical method and normalisation, the maximum frequency is still lower than desired.

This can be rectified by shifting the input signal down in frequency when detectors are requested at frequencies above the point where the DetectorBank can adequately respond. This has the benefit that the signal only has to be processed once before being applied to the DetectorBank, thereby adding less overhead than, for example, increasing the sample rate.

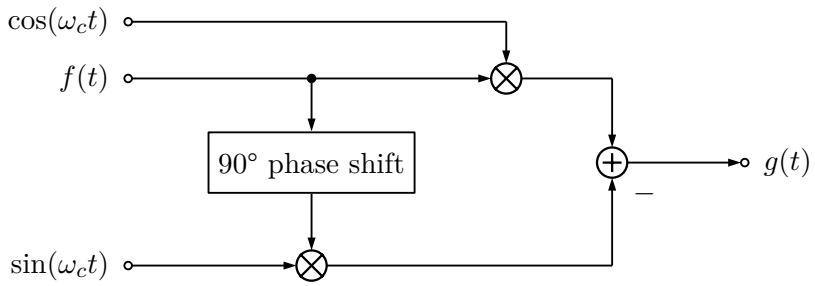


Figure 2.12: Procedure for shifting an input signal $f(t)$ by $\omega_c/2\pi$ Hz.

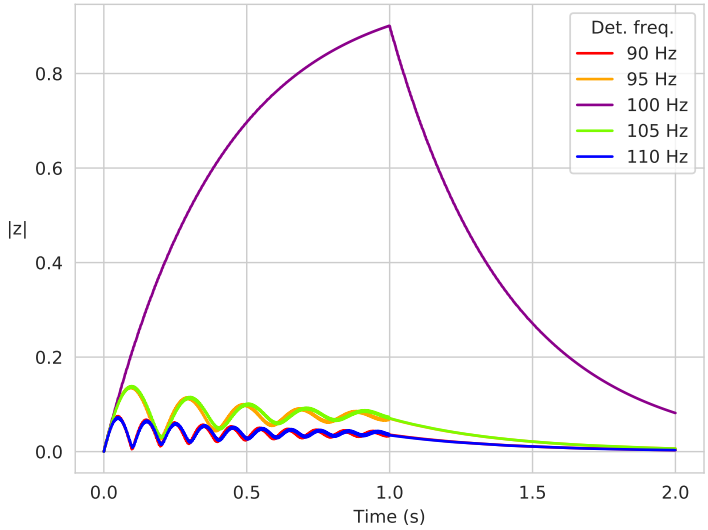


Figure 2.13: Response of five unnormalised Runge-Kutta detectors around 100 Hz to a tone which was generated at 4 kHz and shifted down to 100 Hz.

In our system, frequency shifting is achieved by generating a double sideband signal, then subtracting a quadrature phase shifted version of the signal, which leaves only the upper sideband of the positive frequencies and the lower sideband of the negative frequencies (Van Trees 2001). Figure 2.12 shows how this can be implemented to shift a signal $f(t)$ by $\omega_c/2\pi$ Hz. A phase shift can be implemented using the Hilbert transform (Rabiner & Gold 1975). More information on the Hilbert transform and its application ro frequency shifting is provided in Appendix E.

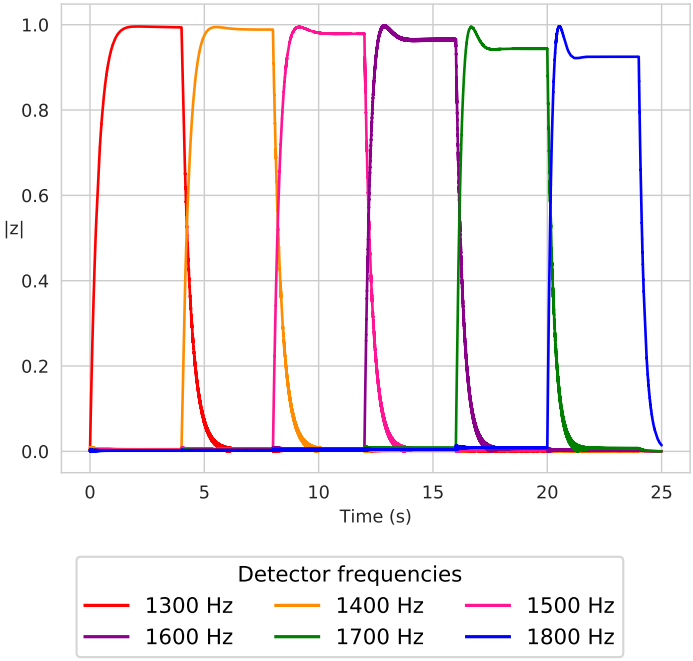


Figure 2.14: Unnormalised Runge-Kutta detector responses to a range of frequencies, increasing by 100 Hz. As the frequency increases, the shape of the responses becomes distorted. From this, the maximum frequency to represent without frequency shifting was chosen to be 1.6 kHz.

Figure 2.13 shows the responses of five unnormalised Runge-Kutta detectors, tuned to 90, 95, 100, 105 and 110 Hz and driven by a sine wave which was generated at 4 kHz — far beyond the empirically determined maximum for an unnormalised Runge-Kutta detector, see Figure 2.2a — and shifted down to 100 Hz. The tone lasts for one second and is followed by one second of silence and the sample rate is 48 kHz.

This process can also be used to shift frequencies up. This will generally not be required for audio signals, as detector responses for 27.5 Hz and greater are adequate when the sample rate is 48 kHz, but for other applications it may be useful. Examples of this are given in Appendix E.3.4, where the various different implementations of the Hilbert transform are also discussed.

When running the software, frequency shifted versions of the input will

Table 2.2: Empirically determined frequency thresholds, f_t, above which frequency shifting will be applied for each combination of numerical method (fourth order Runge-Kutta, RK4, and central difference, CD) and normalisation (search normalised or unnormalised). These values were found for a sample rate of 48 kHz.

Method	**Normalisation**	f_t **(Hz)**
RK4	none	1600
RK4	search	2200
CD	none	500
CD	search	700

automatically be generated if any of the requested detector frequencies are above a certain threshold, f_t, given by the numerical methods and normalisation. Frequencies will be shifted into the range 50 Hz to $f_t + 50$ Hz to guarantee a clean response.

The threshold values, presented in Table 2.2, were obtained empirically from results like those shown in Figure 2.14, which shows the response of unnormalised Runge-Kutta detectors to frequencies ranging from 1.3 kHz to 1.8 kHz. As the frequency increases, the shape of the response begins to distort, with the response reaching a maximum, then decreasing to a steady value. While search normalisation can increase the frequency at which the distortion begins, it cannot prevent it entirely. In the case of the unnormalised Runge-Kutta detectors shown in Figure 2.14, a threshold frequency of 1.6 kHz was chosen. As the shifted signal will be between 50 Hz and $f_t + 50$ Hz, the maximum frequency represented by unnormalised Runge-Kutta detectors will be 1650 Hz. At this frequency, the steady amplitude of the response is within 95% of the maximum value.

In musical terms, a frequency range of 27.5 Hz to 1.6 kHz covers the fundamental frequencies of notes from A0 to G6 or 71 out of the 88 notes on a standard piano: 80% of the note range one may expect to find in music will be adequately covered without frequency shifting.

Figure 2.15 shows the amplitude of steady state responses of a Detector-

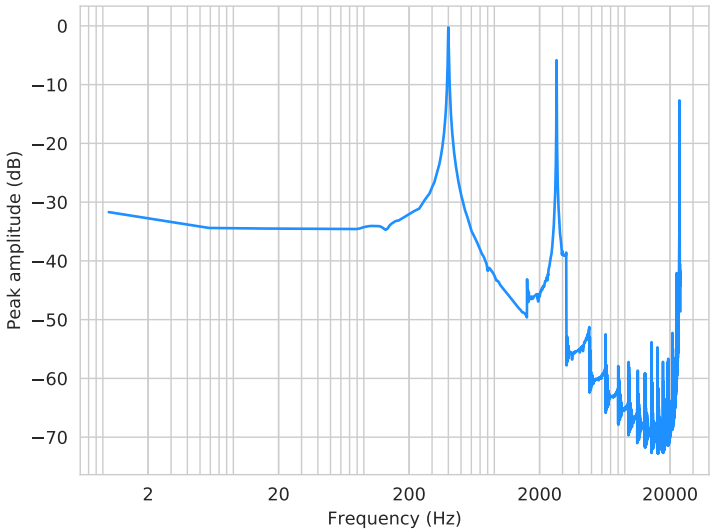


Figure 2.15: Amplitude of steady state responses of DetectorBank to 400 Hz tone; $f_s = 48$ kHz.

Bank with unnormalised Runge-Kutta detectors ranging from 1 Hz to the Nyquist rate (24 kHz) when presented with a 400 Hz tone. In addition to the peak at 400 Hz, the detector at 23.6 kHz also responds, reaching a maximum value 12.7 dB less than that of the 400 Hz detector. At 2.7 kHz, another peak is present. This occurs 5.84 dB below the 400 Hz one and appears to be an artefact of frequency shifting. This can be explained by considering that frequencies above 1.6 kHz are modulated to the range 50 Hz to 1650 Hz. Initially, this is a shift down of 1550 Hz. At frequencies above n times 1.6 kHz, the resultant shift is $1600(n-1) + 1550$ Hz. Peaks then appear at double 1550 Hz minus the input frequency: in this case $3100 - 400 = 2700$ Hz. There are also small peaks at 1.6 kHz intervals above this, but always below −50 dB. A future implementation of the DetectorBank may solve this problem by employing a better method of frequency shifting than single sideband modulation via the Hilbert transform.

The peak at $f_s/2 - f_{\text{in}}$ may be due to the nonlinearity of the system, suggesting a higher sample rate may be required to represent the input

without aliasing. However, as this is not deleterious to the system when used for musical applications — and the scale, location and extent are known — these are not investigated further in this project.

Testing the DetectorBank response to Low frequencies demonstrated that there is less orbital eccentricity and better rejection of nearby frequencies when signals were generated at lower sample rates. Therefore, despite the spurious responses at the Nyquist frequency and double the shifting frequency minus the input, from here on, unless otherwise stated, the sample rate used to characterise the DetectorBank responses will be 48 kHz as, when used with frequency shifting and amplitude scaling, this provides the best results for the range of fundamental frequencies typically found in music.

2.2.4 Propinquitous frequencies

As was observed in discussion of Figure 2.8, a degenerate detector will still respond when the driving frequency and characteristic frequency are not the same, but are close. As the discrepancy between the frequencies increases, the response becomes weaker. The range of frequencies to which it will respond, and the strength with which it responds, is determined by the bandwidth of the detector.

Section 2.2.3 introduced frequency shifting, which is used to extend the system bandwidth of detectors operating on a 48 kHz input up to the Nyquist frequency. Therefore, the effect of higher sample rates will not be considered here. The detector bandwidth will be investigated in Section 2.2.5; this section focusses on the response of detectors with a characteristic frequency in close proximity to the driving frequency.

When the detector's characteristic frequency and the driving frequency are similar, the response will oscillate at the difference between the frequencies.[3] This suggests the response will reach its maximum after half a period

[3]This oscillation is not to be confused with the oscillations in the response due to eccentricity in the orbit, which was discussed earlier in this chapter, in Octaves, and

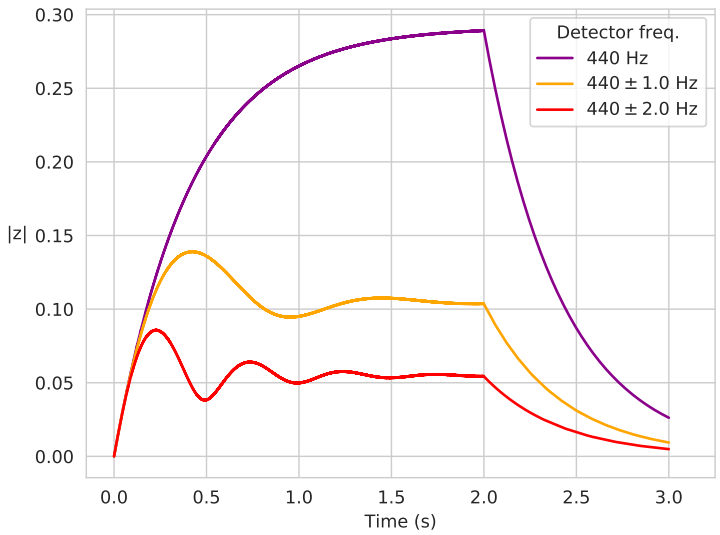


Figure 2.16: Responses of degenerate detectors at the driving frequency and at ± 1 Hz and at ± 2 Hz from this frequency.

of oscillation, $1/2(f_{\text{in}} - f_0)$ seconds in this case, however it invariably takes a shorter time than this. For example, in Figure 2.16, the orange line shows the response of a detector at ± 1 Hz from the driving frequency. One might, therefore, expect the maximum to occur after 500 ms, but the response is clearly reaching maximum significantly earlier than this. This cannot be attributed to errors introduced by numerical approximation, as the time shift is the same for both Runge-Kutta and central difference detectors.

As shown in Table 2.3, for degenerate detectors operating at 0.5 Hz from the driving frequency (and with a damping of $1 \cdot 10^{-4}$) the time advance is 231 ms, which decreases to 3.96 ms when the frequency difference increases to 5 Hz.

As the oscillation frequency may be ascertained by measuring the time from maximum to the following local minimum, this time shifting will allow Δf to be obtained sooner.

The characteristics of the response immediately after forcing begins will

illustrated in Figures 2.3 to 2.6.

Table 2.3: For detectors Δf Hz from the input frequency (440 Hz), this table gives the expected maximum time (half a period, T), the time at the which the maximum is found in practice, t_m, and the resultant time shift. The damping factor used here is $1 \cdot 10^{-4}$.

Δf **(Hz)**	$T/2$ **(ms)**	t_m **(ms)**	$t_m - T/2$ **(ms)**
0.5	1000	768.9	−231.1
1.0	500	423.4	−76.62
1.5	333.3	295	−38.37
2.0	250	226.8	−23.25
2.5	200	184.7	−15.29
3.0	166.7	155.1	−11.52
3.5	142.9	134.7	−8.169
4.0	125	118.8	−6.228
4.5	111.1	106.3	−4.840
5.0	100	96.04	−3.958

be investigated in Section 2.2.6.

2.2.5 Detector bandwidth

Any algorithm which analyses the responses may miss events where the detector and driving frequencies do not quite match. For example, as can be seen in Figure 2.16, the maximum response for detectors at ±1 Hz from the driving frequency is roughly half that of the correct detector. In fact, the ratio of maximum amplitude of the matched response to that of a slightly mismatched response is approximately:

$$\frac{\max(|z_d|)}{\max(|z_0|)} \approx \frac{1}{|f_{\text{in}} - f_0| + 1} \tag{2.1}$$

where z_d is the mismatched detector response, z_0 is the matched detector response, f_{in} is the driving frequency and f_0 is the detector frequency.[4]

In signal processing, the standard definition of bandwidth is the frequency at which the response has decreased by $1/\sqrt{2}$. This is equivalent to a drop of 3 dB, hence it is commonly referred to as the 3 dB point. In

[4] This only holds when the sample rate is 48 kHz. At higher sample rates, the maximum values of propinquitous detectors are higher.

Table 2.4: Empirical values for first Lyapunov coefficient required for a 3 dB point at ± 1 to 5 Hz, i.e. a detector bandwidth of 2–10 Hz, at $f_s = 48$ kHz.

Bandwidth (Hz)	**First Lyapunov coefficient**
2	−0.160
4	−1.278
6	−4.303
8	−10.183
10	−19.863

this system, '3 dB point' will be used to refer to the frequencies at which detectors have a steady state response 3 dB lower than a detector tuned to the input frequency. S Equation (2.1) suggests that the 3 dB point of a detector is about ±0.41 Hz from the detector's frequency. Experimental measurements put the 3 dB point at ±0.46 Hz, i.e. the detector bandwidth is 0.92 Hz (when the sample rate is 48 kHz, the damping is $1 \cdot 10^{-4}$ and the first Lyapunov coefficient is zero).

This extremely sharp cutoff will be good for many applications, but for some — e.g. music analysis, where performed notes will rarely be at exactly the 'correct' frequency, but will be perceived as being the correct pitch — it may be desirable to widen the response of the detectors. This can be achieved by using a non-degenerate Hopf bifurcation, i.e. one where the first Lyapunov coefficient, b, is non-zero. For a supercritical bifurcation, i.e. one which exhibits stable periodic solutions, b should be negative.

The value of b required for a 3 dB point at ±1 Hz from the detector frequency for a purely sinusoidal input with amplitude $X = 25$ was experimentally determined as approximately -0.16, which rapidly increased as the frequency difference (in Hertz) is increased (see Table 2.4).

A proportionality is observed between the log of the values in Table 2.4. Genetic algorithms[5] were used to confirm this relationship for bandwidths

[5] As it cannot be ruled out that there are local minima, classical techniques for solving these problems, such as gradient descent, may not be appropriate. As the calculation only has to be run once, the computational overhead is not a significant consideration. The

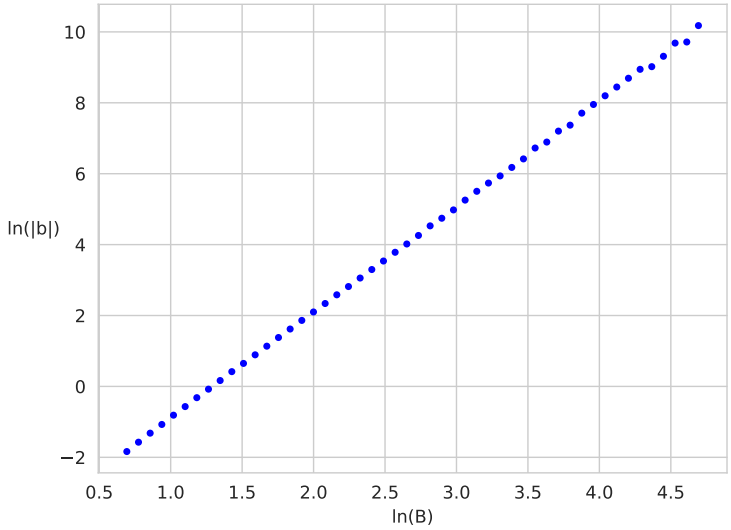


Figure 2.17: $\ln(|b|)$ against $\ln(B)$ for the b values found by the genetic algorithm.

up to 110 Hz. The results of this can be seen in Figure 2.17. As these values closely approximate a straight line, an equation relating bandwidth, B, and first Lyapunov coefficient, b, can be derived using data from Table 2.4:

$$b = -\exp\Big(m\Big(\ln(B) - \ln(x_0)\Big) + \ln(y_0)\Big) \tag{2.2}$$

where $x_0 = 2$, $x_1 = 10$, $y_0 = 0.16$, $y_1 = 19.863$ and $m = \Big(\ln(y_1) - \ln(y_0)\Big)/\Big(\ln(x_1) - \ln(x_0)\Big)$. With these values, $m = 2.999 \approx 3$; using this and log rules, Equation (2.2) can be greatly simplified to

$$b = -0.02B^3 \tag{2.3}$$

When tested with a range of target bandwidths from 1 Hz to 24 kHz,

GAs were implemented in Python using the DEAP (Distributed Evolutionary Algorithms in Python) framework (Fortin et al. 2012); for full implementation, please see the Python scripts in the GitHub repository at https://github.com/keziah55/ExtraThesisMaterial/tree/master/DetectorBank_development.

Table 2.5: Empirical values for first Lyapunov coefficient, b, required for a 3 dB point at bandwidths, B, of 2–10 Hz at high sample rates, with $X = 25$.

Bandwidth (Hz)	**First Lyapunov coefficient** $f_s = 96\,\text{kHz}$	**First Lyapunov coefficient** $f_s = 192\,\text{kHz}$
2	−0.04	—
4	−1.25	−0.32
6	−4.44	−3.60
8	−10.64	−9.99
10	−20.61	−20.43

the results of Equations (2.2) and (2.3) differ by no more than 4%. The discrepancy increases with the requested bandwidth. Bandwidths that may be used for musical applications are unlikely to be greater than hundreds of Hertz. The difference between the fundamental frequencies of the two highest notes on a standard 88-key piano, C8 and B7, is 235 Hz. In the case $B = 235\,\text{Hz}$, the maximum variation between b as calculated by the full expression and the simplified version is 2%. Discrepancies of this scale will not be deleterious to the output, therefore Equation 2.3 is an acceptable substitution in musical applications.

The first Lyapunov coefficients required for bandwidths of 2–10 Hz at higher sample rates (96 kHz and 192 kHz) did not display the same trend (see Table 2.5). As the sample rate is increased, the minimum detector bandwidth increases and so smaller values of first Lyapunov coefficient are required to widen the response. For 192 kHz, there is no value of b small enough to bring the maximum responses of detectors at ±1 Hz down to −3 dB. The limit seems to be around −1.36 dB. However, for larger bandwidths, e.g. 8 or 10 Hz, the first Lyapunov coefficient required is similar at all tested sample rates. Therefore, clearly at neither of these higher sample rates will plotting $\ln(|b|)$ against $\ln(B)$ yield a straight line.

As the first Lyapunov coefficient is increased, the magnitude of all the responses decreases, even those where the input frequency and the detector

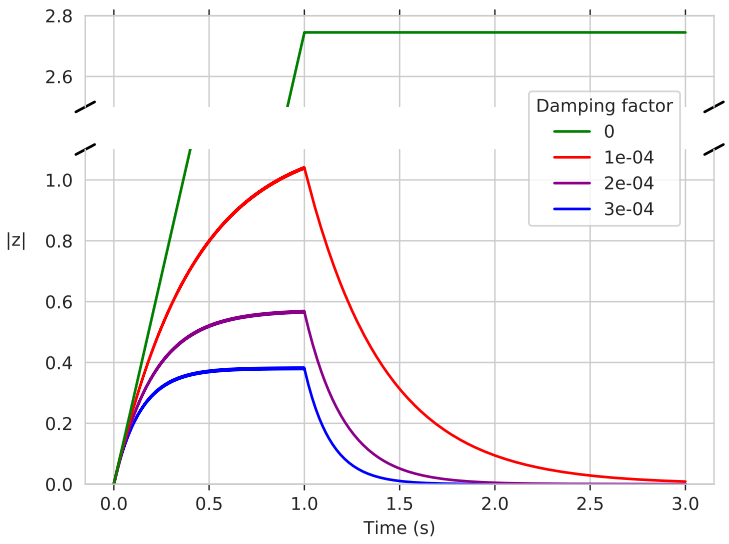


Figure 2.18: The responses of four detectors, three damped with different dampings, $1 \cdot 10^{-4}$, $2 \cdot 10^{-4}$ and $3 \cdot 10^{-4}$, and one undamped.

frequency are exactly matched. This problem is addressed by amplitude normalisation, discussed in Section 2.2.8.

2.2.6 Damping

Figure 2.18 shows the responses of four detectors to a one second sine tone. One detector is undamped — the response reaches its maximum, but does not return to zero after the tone stops. The other three detectors are damped with different damping factors: $1 \cdot 10^{-4}$, $2 \cdot 10^{-4}$ and $3 \cdot 10^{-4}$ (the first of those being the damping which has been used in all the experiments here thus far).

The choice of damping factor effects several aspects of the response, from the minimum bandwidth of a detector to the time response.

Minimum detector bandwidth

As would be expected, as the damping factor increases, the maximum amplitude becomes lower and the detector bandwidth widens.

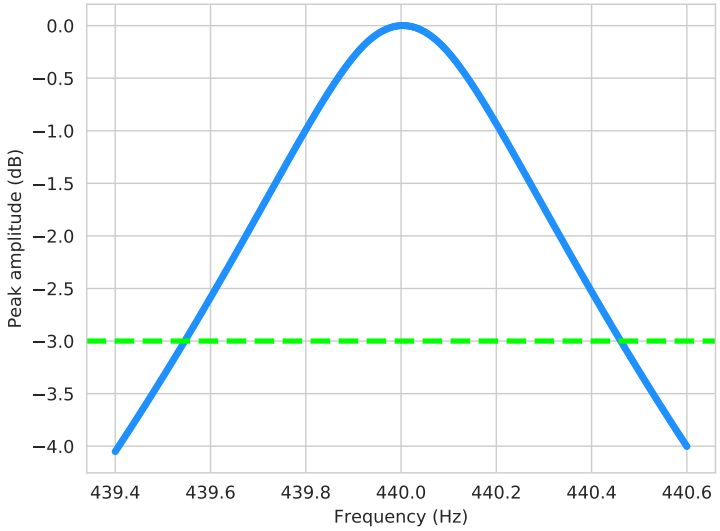


Figure 2.19: Maximum steady state amplitude of 501 detector responses around 440 Hz, when $f_s = 48$ kHz and damping factor is $1 \cdot 10^{-4}$. The 3 dB points found here are 439.546 Hz and 440.461 Hz.

Table 2.6 shows experimentally derived values for the natural bandwidth of degenerate Hopf detectors (at three sample rates) for five damping factors from $1 \cdot 10^{-4}$ to $5 \cdot 10^{-4}$. For each damping factor, the bandwidth was found by making a DetectorBank with 501 detectors around the centre frequency (at increments of a few millihertz), then finding at which frequencies the response amplitude is closest to 3 dB from that of the centre frequency. As these values are not guaranteed to be positioned exactly around the centre, the bandwidth is taken to be twice the largest difference from centre, i.e. the minimum value that covers both 3 dB points. Figure 2.19 shows the peak amplitudes for a DetectorBank around 440 Hz, when sample rate is 48 kHz and damping factor is $1 \cdot 10^{-4}$. The 3 dB points are found at -0.454 Hz and $+0.461$ Hz, giving in a minimum bandwidth of 0.922 Hz.

From the values in Table 2.6 it can be seen that the detector bandwidth changes linearly with damping at all three sample rates.

Genetic algorithms were used to verify these results. The method of

Table 2.6: Minimum bandwidths for various damping levels and sample rates.

Damping	**Bandwidth (Hz)** $f_s = 48\,\text{kHz}$	**B'width (Hz)** $f_s = 96\,\text{kHz}$	**B'width (Hz)** $f_s = 192\,\text{kHz}$
$1 \cdot 10^{-4}$	0.922	1.824	3.653
$2 \cdot 10^{-4}$	1.832	3.648	7.307
$3 \cdot 10^{-4}$	2.752	5.496	11.040
$4 \cdot 10^{-4}$	3.606	7.328	14.700
$5 \cdot 10^{-4}$	4.560†	9.160	18.367

† Please note that this value will be increased to 4.860 Hz following the investigation in Section 2.2.7.

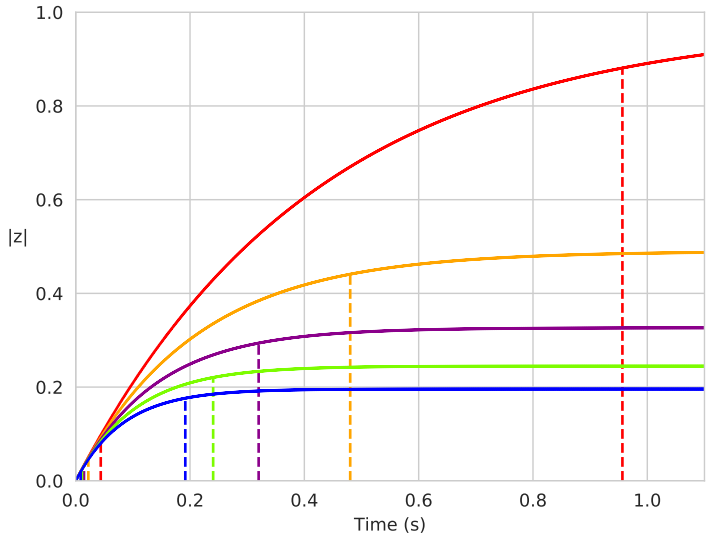
determining bandwidth used here differs from that described above. Rather than creating a large DetectorBank with detectors at fixed, discrete frequencies, the genetic algorithm creates detectors at frequencies plus and minus half an estimated bandwidth from centre. Many potential values are generated and tested, with the final result being the bandwidth at which the responses were closest to $-3\,\text{dB}$ from centre.

The genetic algorithm yielded values very similar to the original results. For a sample rate of 48 kHz, the largest difference between the minimum bandwidths is 0.327%. At higher sample rates, the difference ranged from 0.073% to 0.633%.

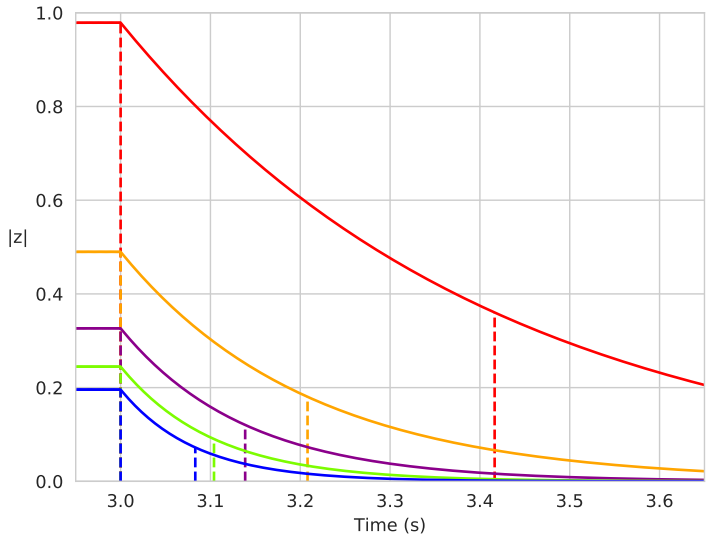
Equation (2.3), which finds the first Lyapunov coefficient for a given bandwidth (when the sample rate is 48 kHz), was derived from experiments which used a damping factor of $1 \cdot 10^{-4}$. For damping factors greater than this (up to $5 \cdot 10^{-4}$) and bandwidths above the minimum values given in Table 2.6, this relation still holds.

Time response

Figure 2.20a shows the responses of five detectors with different damping factors with the 10%–90% rise times marked by dotted lines and Figure 2.20b shows the same responses with the relaxation time (time taken for the am-



(a)



(b)

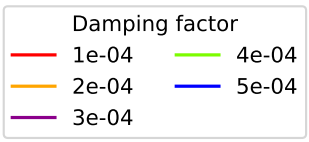


Figure 2.20: The responses of five damped detectors with (a) the rise times marked and (b) the relaxation times marked.

Table 2.7: Rise times for different damping factors ($f_s = 48\,\text{kHz}$)

Damping	**10% time (ms)**	**90% time (ms)**	**Rise time (ms)**
$1 \cdot 10^{-4}$	43.8125	956.438	912.625
$2 \cdot 10^{-4}$	22.0833	480.229	458.146
$3 \cdot 10^{-4}$	14.4583	320.042	305.583
$4 \cdot 10^{-4}$	10.8958	240.458	229.562
$5 \cdot 10^{-4}$	8.66667	191.729	183.062

Table 2.8: Relaxation times for different damping factors ($f_s = 48\,\text{kHz}$)

Damping	**Relaxation time (ms)**
$1 \cdot 10^{-4}$	416.396
$2 \cdot 10^{-4}$	208.104
$3 \cdot 10^{-4}$	138.667
$4 \cdot 10^{-4}$	103.958
$5 \cdot 10^{-4}$	83.1250

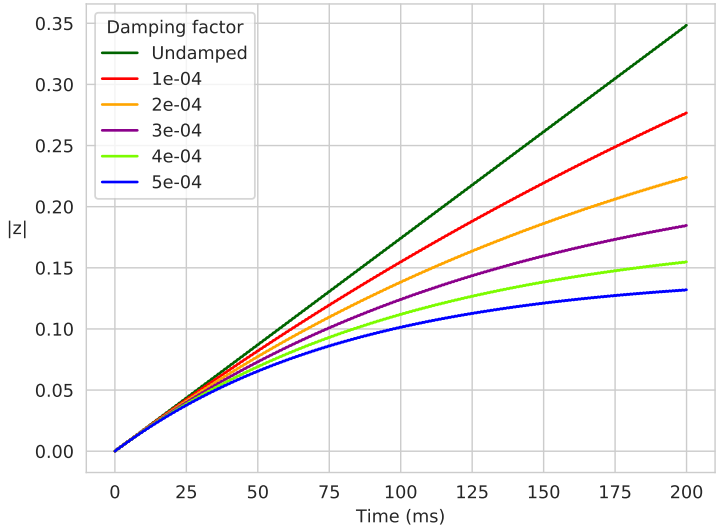
plitude of the response to fall to $1/e$ of the maximum) marked. The exact times are given in Tables 2.7 and 2.8. It can be seen that the times measured here are inversely proportional to changes in damping factor.

When in relaxation, the responses drop by $(1 - d)^{1/2}$ at every sample, where the factor of two appears in the power because each value of z is calculated using the previous two z values. Therefore, for a desired relaxation time, t ms, the required damping factor can be found with

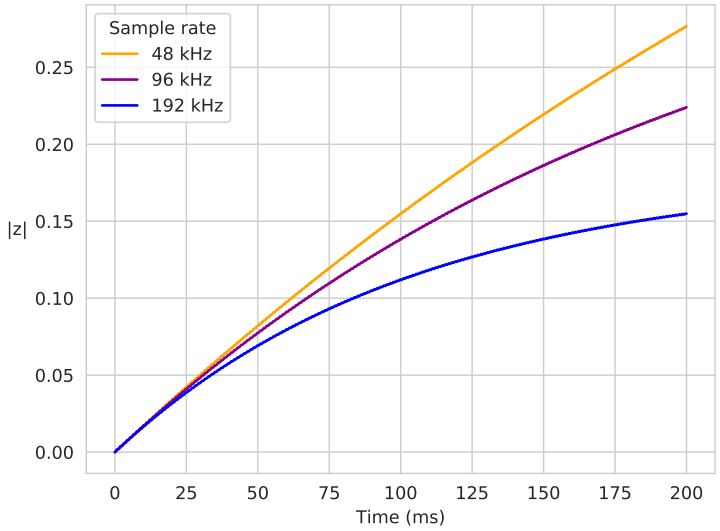
$$d = 1 - \exp(-2t/f_s) \tag{2.4}$$

Taken together, Tables 2.6 to 2.8 show that there is a trade-off between improving time performance of the detectors and reducing the frequency selectivity.

Figure 2.21 plots the initial responses when the damping factor and sample rate are varied. In both cases, although the detectors all respond at the same rate initially, the responses start to diverge at approximately 25 ms. Responses with lower damping factors or sample rates increase in amplitude at a faster rate and reach a higher maximum value.



(a) Initial response of 440 Hz detectors to 440 Hz tone at 48 kHz, at a variety of damping levels.



(b) Initial response of 440 Hz detectors to 440 Hz tone generated at a variety of sample rates, with a constant damping factor of $1 \cdot 10^{-4}$.

Figure 2.21: Initial responses at different damping levels and sample rates.

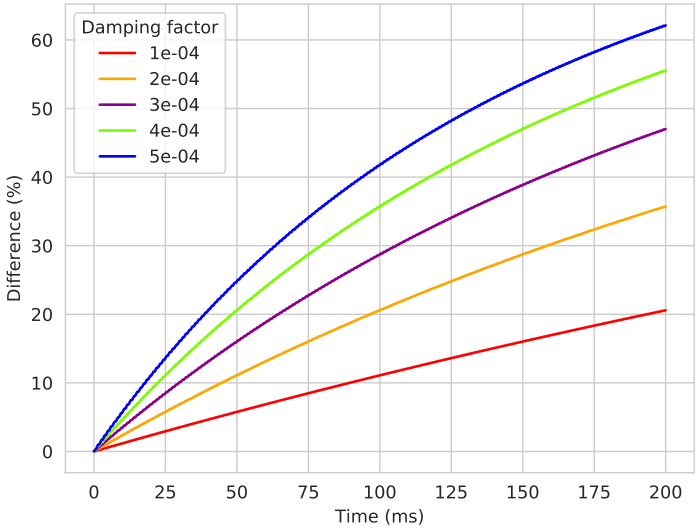


Figure 2.22: Divergence between damped and undamped responses of Figure 2.21a, as a proportion of the undamped response.

Figure 2.22 shows the difference between damped and undamped responses as a proportion of the undamped response at every sample. The least damped response diverges most slowly, taking 89.54 ms to differ from the undamped response by 10%; the most damped response differs by 10% after only 18 ms.

We will return to the difference in response rate of change and maximum value due to damping factor when discussing Amplitude normalisation in Section 2.2.8.

Summary

Increasing the damping has a similar effect to increasing the sample rate. Figure 2.8 shows that using a higher sample rate leads to greater oscillations in the response. Section 2.2.5 discussed the effect of the first Lyapunov coefficient on bandwidth, mostly considering a sample rate of 48 kHz, but mention was made of the bandwidth and first Lyapunov coefficient at higher sample rates, the effect of which is very similar to that of increasing the

damping. This is because when implemented, in both the Runge-Kutta and central difference methods, each z value is scaled by both $1/\text{sr}$ and $1 - d$.

2.2.7 Input amplitude

The bandwidth of a detector is also seen to widen when the forcing amplitude is increased. The relationship between bandwidth and first Lyapunov coefficient given in Equation (2.3) was experimentally determined for sinusoidal forcing with a constant amplitude $X = 25$. It is therefore necessary to adapt this to accommodate varied forcing amplitudes.

Genetic algorithms were once again used to analyse the relationship between first Lyapunov coefficient and bandwidth at different damping factors as the input amplitude is varied. These results can be seen in Figure 2.23. Although the lines are a somewhat irregular shape at bandwidths below 6 Hz, it can be seen that the output follows a similar pattern for each amplitude tested. Above the minimum detector bandwidth, the relationship between first Lyapunov coefficient and amplitude is the same for all damping factors.

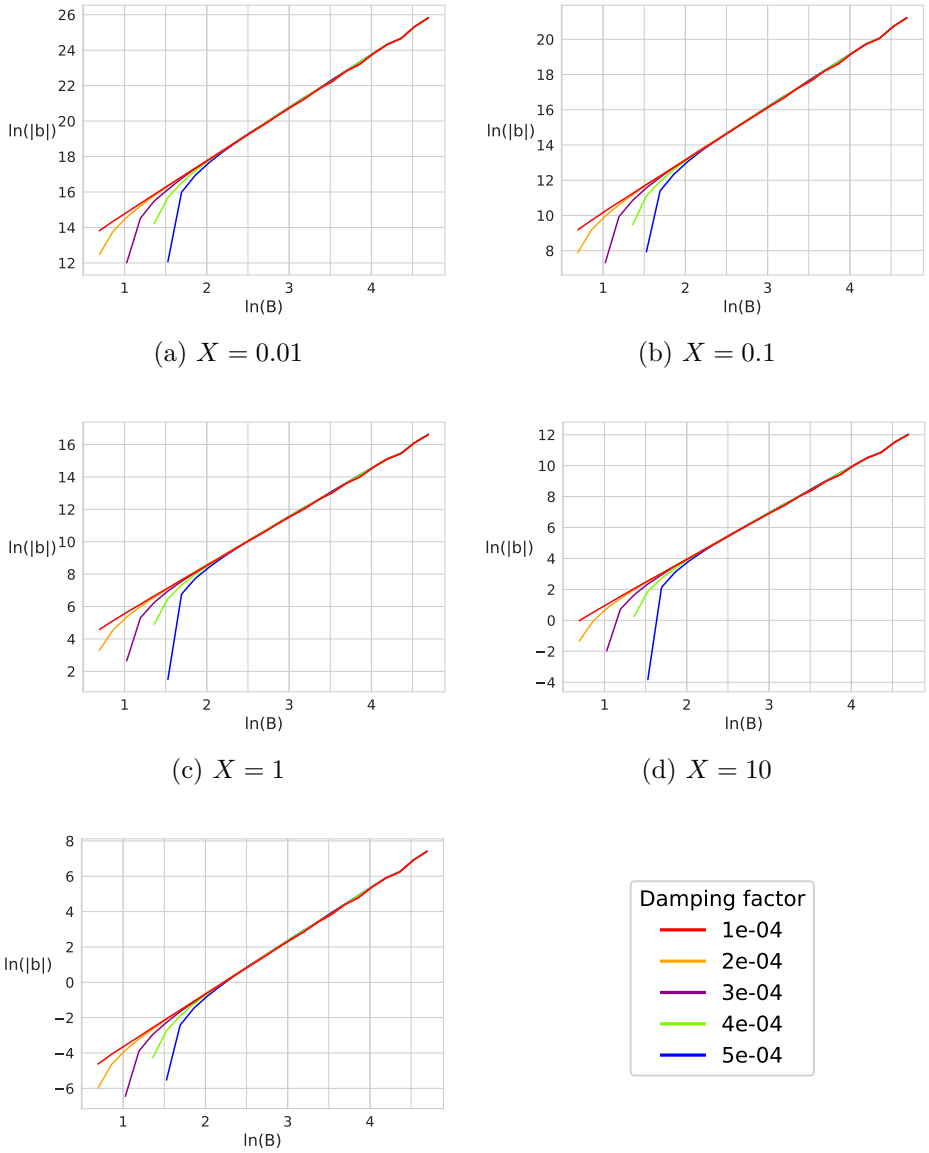
From these results, we can say that a known first Lyapunov coefficient, b_0, can be scaled by the ratio of amplitudes squared to find the first Lyapunov coefficient required when the forcing amplitude is changed.

$$b_1 = b_0 \left(\frac{X_0}{X_1} \right)^2 \tag{2.5}$$

Equation (2.3) calculates the first Lyapunov coefficient for a given bandwidth B when the amplitude $X_0 = 25$. Substituting these values into Equation (2.5) yields the following expression for the first Lyapunov coefficient with an arbitrary input amplitude X:

$$b = -\frac{12.5B^3}{X^2} \tag{2.6}$$

When attempting to verify this relationship with the data from the ge-



(a) $X = 0.01$ (b) $X = 0.1$

(c) $X = 1$ (d) $X = 10$

(e) $X = 100$

Figure 2.23: First Lyapunov coefficient (b) values required for various bandwidths (B) at different levels of damping ($1 \cdot 10^{-4}$ to $5 \cdot 10^{-4}$), when the forcing amplitude (X) is varied, as given by genetic algorithms. Results are shown for values above the minimum bandwidth for the given damping factor.

netic algorithm, some variation was found for values approaching the minimum bandwidth. For the two smallest damping factors, this was within 5%; however for larger damping factors this error increases. When calculating b for a bandwidth of 4.602 Hz, at a damping factor of $5 \cdot 10^{-4}$, the error was up to 95%. This error reduced to within 5% when the desired bandwidth was increased to 5.437 Hz. This suggests that the minimum bandwidth for detectors with a damping of $5 \cdot 10^{-4}$ should be widened. Considering bandwidths above 4.860 Hz at this damping factor reduces the largest error to 8%.

2.2.8 Output amplitude

In nonlinear systems, the principle of superposition does not apply, so all input parameters must be considered when describing the output of the system.

Amplitude scaling (see Section 2.2.2) corrects any decay in output amplitude due to characteristic frequency, numerical method or frequency normalisation. However, other DetectorBank parameters — forcing amplitude, sample rate and damping factor — can affect the output amplitude.

The effect of changing forcing amplitude can be seen in Figure 2.24. As the input amplitude is increased while all other parameters remain the same, the output amplitude approaches the cube root of the input. For the special case of a degenerate Hopf bifurcation, given in Equation (1.37), the cubic term disappears, and the system becomes linear.

The effect of varying damping factor was discussed in Section 2.2.6. This focussed on the relationship with minimum bandwidth and time responses; however, the results also provide insight into the effect on output amplitude. From Figure 2.20a, it can be seen that the output amplitude decreases as the damping is increased. Considering this in conjunction with the results in Tables 2.7 and 2.8 — which show that increasing the damping reduces

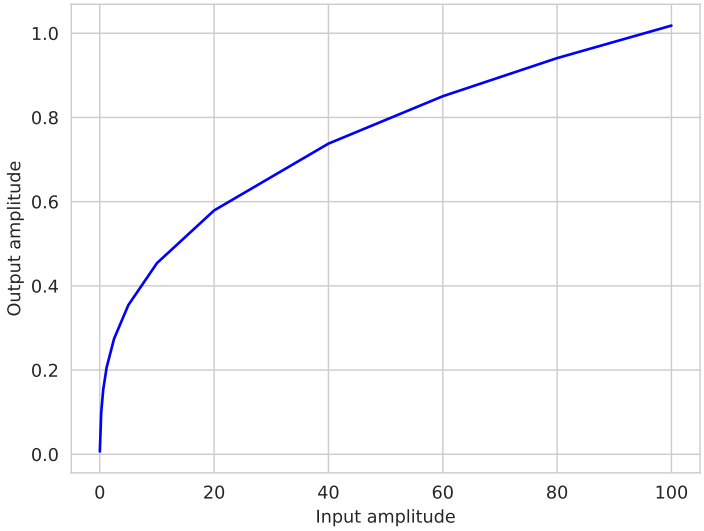


Figure 2.24: As the input signal amplitude increases, the maximum response amplitude increases at a rate approximate to the cube root of the input amplitude. The bandwidth used when generating this figure was 5 Hz, i.e. the first Lyapunov coefficient was non-zero.

the rise and relaxation times — suggests that using a higher damping and scaling up the response can improve on the response of a detector with a smaller damping factor.

The effect of varying the sample rate can be seen by comparing the axes in Figures throughout Section 2.2.1, for example Figure 2.8: as the sample rate increases, the output amplitude decreases.

The situation may arise where a user wishes to analyse or compare the responses of different DetectorBanks. This would require consistency in the output of any DetectorBank given the same input. To achieve this, we employ amplitude normalisation.

Amplitude normalisation

Amplitude normalisation is implemented by finding the real and imaginary parts of the maximum value in a detector's response to a 60 second tone

at the characteristic frequency. These values are then used to scale the detector's response. It also corrects any eccentricity in the orbit by scaling the imaginary part of the response, an aberration which was discussed in Octaves.

To do this, two scale factors are employed: s_a and s_i. The former is a complex number which scales the z results and the latter uses the ratio of the real and imaginary parts of the orbit to scale only the imaginary part of z.

The amplitude scale factor, s_a, is simply $1/z$, where z is the point at which $|z|$ is at maximum.

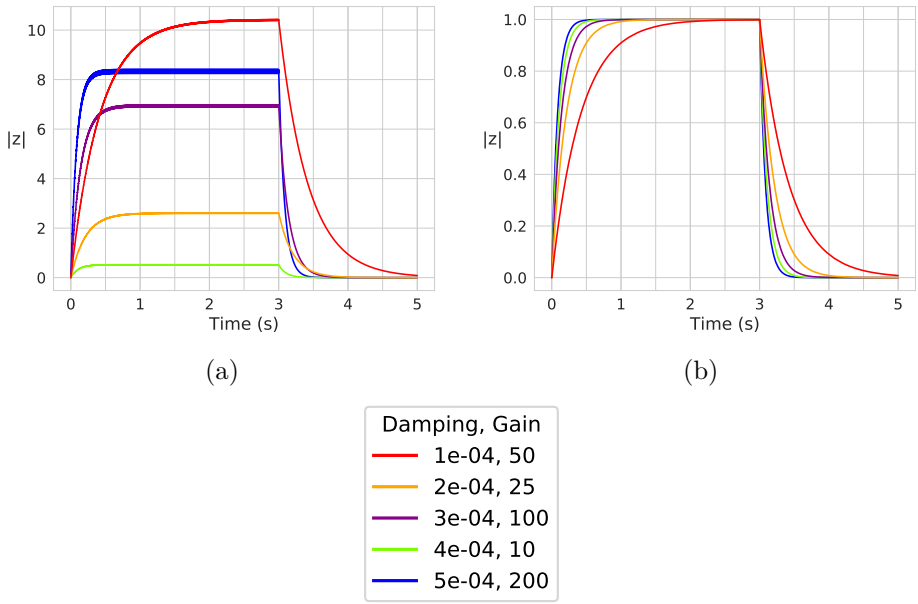
The eccentricity scale factor, s_i, is found by analysing the periodic orbits towards the end of the response, once the orbits have reached their full extent (see, for example, Figure 2.4 for a comparison of the first and last n periods of a response). The scale factor is the ratio of the maximum real and maximum imaginary values in these periods.

z values generated in response to the user's input buffer will then be normalised using the s_a and s_i obtained from the test tone.

Figure 2.25 shows the results of amplitude normalisation on detectors with different damping and gain levels. All responses now reach a maximum of 1, with the detectors with larger damping factors reaching their maxima earlier.

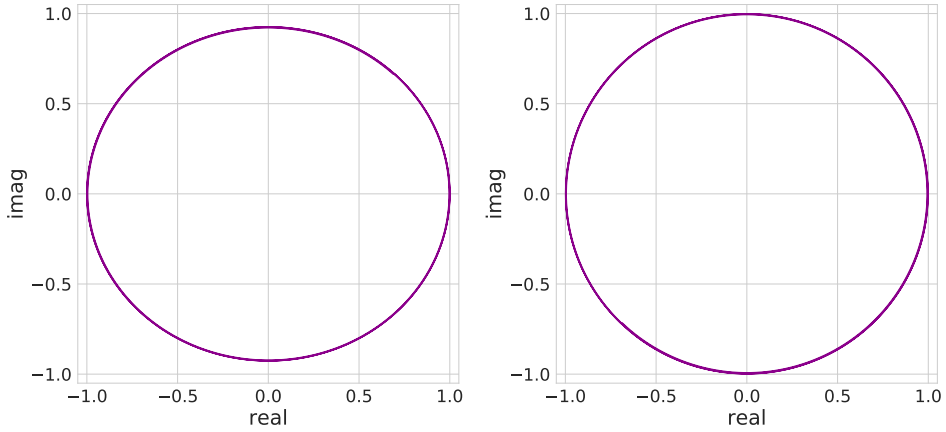
Figure 2.26 shows the effect of amplitude normalisation on orbital eccentricity. The eccentricity of the ellipse in Figure 2.26a (a reprint of Figure 2.4b in Section 2.2.1) is $7.475 \cdot 10^{-2}$; in Figure 2.26b, the eccentricity has dropped by a factor of eight to $9.347 \cdot 10^{-3}$.

Measuring the rise and relaxation times with amplitude normalisation applied shows small differences between the results given in Tables 2.7 and 2.8, generally in the order of tens or hundreds of milliseconds, up to a maximum time difference of 0.325%



(a) (b)

Figure 2.25: Responses of five detectors with different damping factors and gains (a) unnormalised, and (b) with amplitude normalisation.



(a) Unnormalised (b) Normalised

Figure 2.26: Last five periods of 5 Hz response (b) with and (a) without amplitude normalisation. It can be seen that the unnormalised orbits have a greater eccentricity than the normalised ones.

2.2.9 Multiple simultaneous frequencies

Nonlinearities may introduce artefacts at frequencies within the range of interest. The discussion of Pitch and time perception in the auditory system mentioned the the perception of frequencies not present in a stimulus. This phenomenon arises because of cochlear nonlinearities. There is, therefore, a legitimate cause for concern that a system which operates with the same mechanical process as the outer hair cells in the cochlea — the Hopf bifurcation — will give false results when multiple frequencies are presented simultaneously. However, this does not occur; Appendix D presents demonstrations of this.

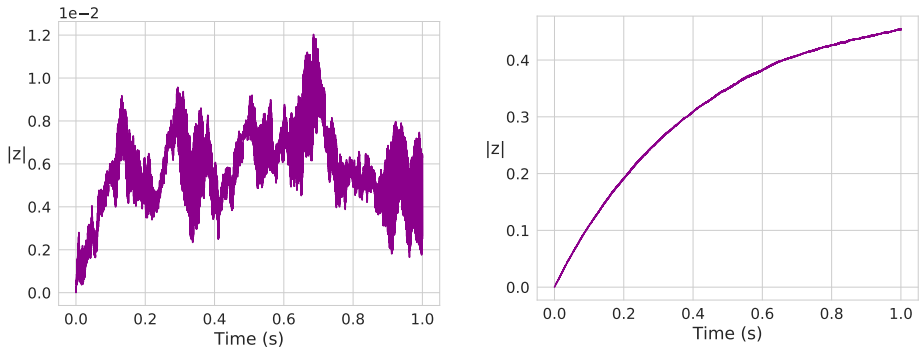
The response to a signal in the presence of wideband noise presents a more legitimate cause for concern, as, due to the nonlinear nature of the system, it is not possible to deal with the wideband response of the system by considering the superposition of partials. This is investigated below.

Wideband noise

Figure 2.27a shows a detector response to uniformly distributed wideband noise; the power spectral density is uniform up to the Nyquist sampling limit and the probability density function is uniform between $+1$ and -1. It can be seen that the range of resulting values is two orders of magnitude less than typical responses shown earlier in this chapter. (See, for example, Figure 2.16.)

Figure 2.27b shows the response to white noise and a sine tone presented simultaneously. It can be seen that there is little difference between the shape of the response to only a sine tone (again as seen in Figures preceding these) and the response when noise is introduced; however the amplitude has halved. The signal-to-noise ratio of the signal used to generate Figure 2.27b is 1.75 dB.

Negative signal-to-noise ratios still produce useable results, although the



(a) Response to white noise.

(b) Response to noise and tone.

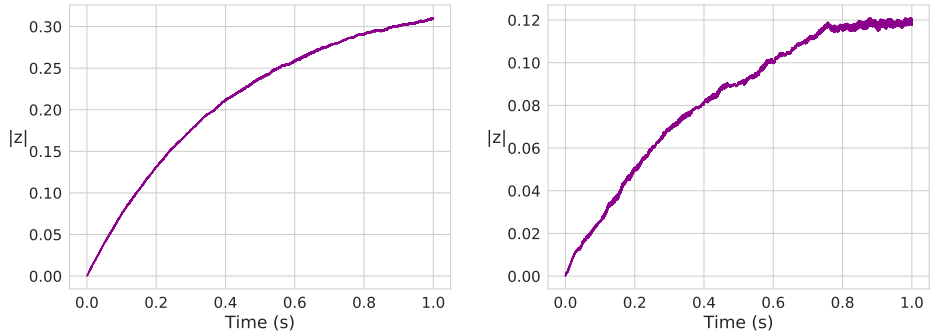
Figure 2.27: Detector response in the presence of white noise. When (a) noise is presented alone, the response of a single detector is low amplitude and noisy. When (b) noise is presented along with a 440 Hz tone at a signal-to-noise ratio of 1.75 dB, the response shape is largely unaffected by the noise, although the amplitude is lower than might be expected.

output amplitude is significantly reduced and the shape of the response becomes noisier. For example, Figure 2.28 shows the output when the signal is (a) 4 dB and (b) 15 dB below the noise.

Figure 2.29 shows the frequency response of a DetectorBank comprising unnormalised Runge-Kutta detectors covering the range 1 Hz to 24 kHz (the Nyquist rate) when the input is white noise. The noise rejection is particularly good at low frequencies; the maximum response amplitude does not exceed −20 dB until 3.2 kHz. Above this point, a series of peaks at 1.6 kHz increments can be seen. As was noted in the discussion of Figure 2.15 in Section 2.2.3, these are artefacts of frequency shifting.

Figure 2.30a similarly recalls Figure 2.15. It shows the frequency response of a DetectorBank presented with a 400 Hz tone and white noise simultaneously, at a signal-to-noise ratio of −4 dB. The 400 Hz peak reaches −9.26 dB and, as seen before, there is another peak at 2.7 kHz. Here, it is 5.12 dB lower than the 400 Hz peak. The response at higher frequencies here is similar to that of only noise, as shown in Figure 2.29.

When the signal-to-noise ratio is further decreased to −15 dB, the 400 Hz



(a) SNR = −4 dB (b) SNR = −15 dB

Figure 2.28: Detector responses to a 440 Hz tone in the presence of noise, where the signal-to-noise ratio, SNR, is negative. As the noise amplitude increases, distortions in the shape of the response become visible.

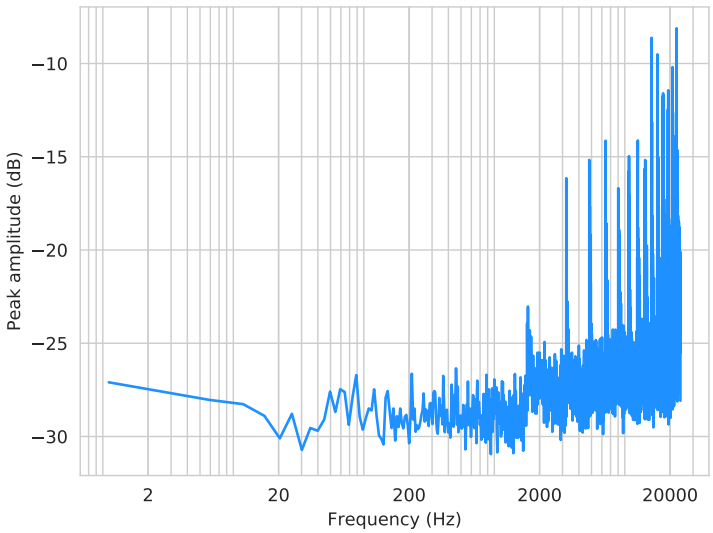
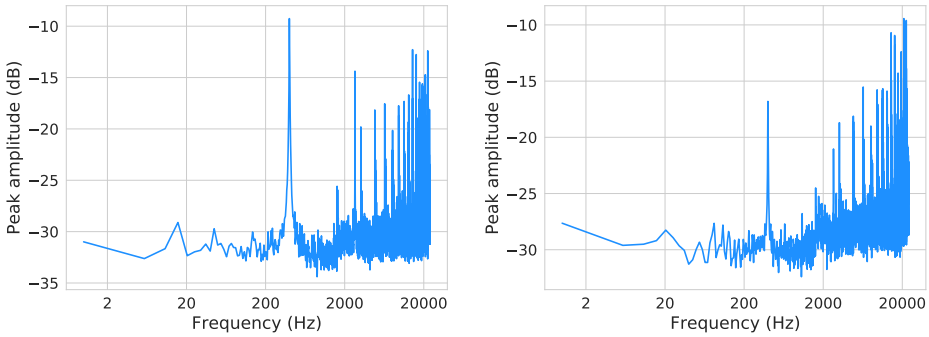


Figure 2.29: Amplitude of steady state responses of DetectorBank to uniformly distributed wideband noise.



(a) SNR = −4 dB (b) SNR = −15 dB

Figure 2.30: Responses across DetectorBank to 400 Hz tone presented with white noise at different signal-to-noise levels.

response only reaches −16.8 dB. Once again, there is a peak at 2.7 kHz; however at −21.0 dB, it is smaller than subsequent peaks which appear due to the noise.

2.3 Summary

Despite the limitations imposed by the uncertainty principle, it is possible to obtain precise data about both the time and frequency characteristics of a signal simultaneously. This is achieved not by directly measuring the signal, but by using it to drive a bank of tuned, nonlinear resonators (‘detectors’). Time and frequency information can then collected by accessing the state variables of the system.

The characteristics of the detectors depend on the input parameters, but it is possible to construct detectors with a narrow bandwidth (down to a minimum of 0.922 Hz) which will reject frequencies outwith this range in less than half the period of the frequency difference.

The deficiencies brought about by the use of numerical approximations to implement the Hopf equation can be circumvented with various techniques including frequency shifting, frequency normalisation, amplitude scaling and amplitude normalisation, with the result that the system bandwidth can be

extended to the Nyquist frequency.

Although the current implementation of frequency shifting introduces artefacts at high frequencies, the fundamental frequencies of 80% of the note range of standard 12-EDO music can be covered without recourse to frequency shifting.

Information about the frequency of the input is available not only from the response of the detector that matches the frequency, but also from the difference oscillations in the responses of propinquitous detectors. Additionally, the bandwidth of a detector can be widened from its minimum to encompass a greater range of the spectrum.

Frequency differences cause responses to diverge in a shorter time than the uncertainty principle suggests is possible, although this is in keeping with results of measuring human hearing.

Various factors affect output amplitude; these can be managed by employing amplitude normalisation, which also corrects eccentricity in the periodic orbits, and thus reduces the small oscillations visible in the low frequency responses.

Potential problems associated with nonlinear systems and multiple simultaneous input frequencies do not emerge here: the system can withstand a large amount of noise in the input signal.

The desired time response must be considered when selecting the DetectorBank parameters, as the rise and relaxation times of the responses can be shortened in proportion to increases in the damping factor. However, features such as amplitude normalisation allow the DetectorBank to produce uniform results as other parameters are changed, such as first Lyapunov coefficient (i.e. bandwidth), characteristic frequency and input amplitude.

The following chapters will consider methods of analysing the DetectorBank output to identify notes in the input. This will focus on detecting onset times; however, the DetectorBank could also be used as the basis for

pitch tracking software, as it enables discrimination of pitch changes within a note. The onset detection software presented in the next chapter could form a note detector, if used in conjunction with a pitch tracker.

Chapter 3

Onset Detection

3.1 Overview

When looking for notes in audio, both the time and frequency domains must be considered. We need to know when the note begins (its onset time) and what frequencies are present (its pitch).

By providing a bank of narrow-bandwidth tuned resonators, referred to here as detectors, the DetectorBank enables us to detect the frequency variation within a note, rather than simply declaring one value to be the frequency of the entire note. When a given detector's characteristic frequency in present in the input signal, the detector will resonate. The time at which it begins to resonate is the onset time of the note. The features of the response, like amplitude and oscillations, provide information about the frequency of the note. Considering these in conjunction with the responses of propinquitous detectors may enable us to calculate the pitch of the input. Although building a pitch tracking algorithm is beyond the scope of this project, the note detection software presented here is designed in such a way that a pitch tracker could be integrated in the future.

A note detector ('NoteDetector' in software) should consist of an onset detector and a pitch tracker, both of which operate on the output samples

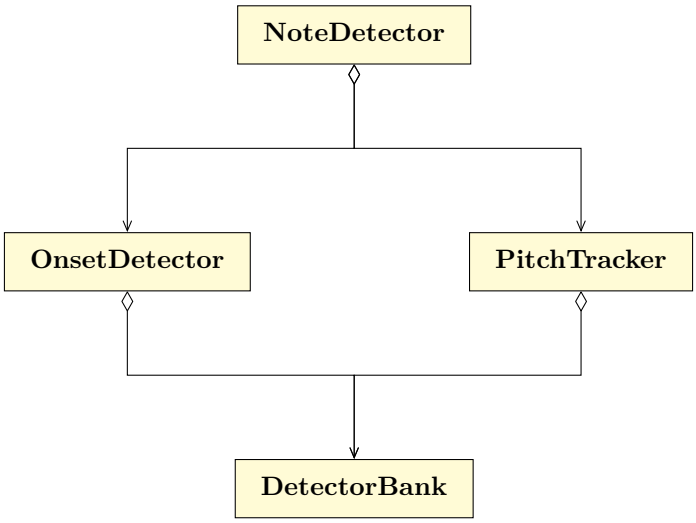


Figure 3.1: UML diagram of a NoteDetector and its components

of a DetectorBank. Figure 3.1 gives a simple UML outline for this structure. In this diagram, the OnsetDetector and PitchTracker share a DetectorBank; in practice, there must be a means of providing these objects with the data they require in an efficient manner.

When presenting the Structure of the DetectorBank software, potential problems in computation overhead were discussed, along with methods of avoiding them: multithreading and the DetectorCache object. Onset detection may have a similar problem, as the output of the DetectorBank must be processed in some way in order to identify the points which correspond to note onsets, a task which is likely to occupy much CPU time and RAM usage. Although the methods presented here are not realtime, a desired outcome of this work is to make possible realtime response and, as such, CPU utilisation is an important design consideration. This informs the overall design presented in Section 3.1.4.

This chapter discusses techniques for onset detection. The methods presented in this chapter are tested with short audio extracts, the details of which are given in Table 3.1. Scores for these are provided in Appendix B. The digital piano extracts were performed by the author; the extract from

Table 3.1: Test audio details

Title	Instrument	Note range	Freq. range
Dream a Little Dream of Me	Digital piano	D4–B4	293.7 Hz–493.8 Hz
Alice	Digital piano	F3–A♭4	174.6 Hz–415.3 Hz
Swan Lake excerpt 1	Digital piano	G1–F♯2	49.0 Hz–92.5 Hz
Swan Lake excerpt 2	Digital piano	B2–C♯4	123.5 Hz–277.2 Hz
Before All Things	Soprano voice	F4–A♭5	349.2 Hz–830.6 Hz

Before All Things is from a private recording.

3.1.1 Initial idea

The initial onset detection algorithm was very simple. Nevertheless, it embodies important concepts which are crucial to a successful implementation of a more sophisticated version.

It works by thresholding and backtracking: when a response exceeds a given threshold, it backtracks, sample by sample, to find the point at which the detector began to react.

When backtracking, the current value is compared with the mean of the log of the preceding N samples. If this mean is less, the sample under consideration is moved back one step and checked again. When the mean log is no longer smaller, the onset has been located.

```
1  # get value at current sample, n,and channel, k
2  current = getResultItem(k,n)
3
4  # get mean of log of previous N values
5  mean = 0
6  for i in range(n-N, n):
7      mean += log(getResultItem(k,i))
8  mean /= N
9
10 # begin backtracking
11 while mean < current:
12
13     # decrement current sample number
14     n -= 1
15
16     # get new 'current' value
```

```
17      current = getResultItem(k,n)
18
19      # remove most recent value from mean
20      mean -= log(current/N)
21
22      # add new (older) value to mean
23      mean += log(getResultItem(k,n-N)/N)
```

Code Extract 3.1: Initial backtracking algorithm

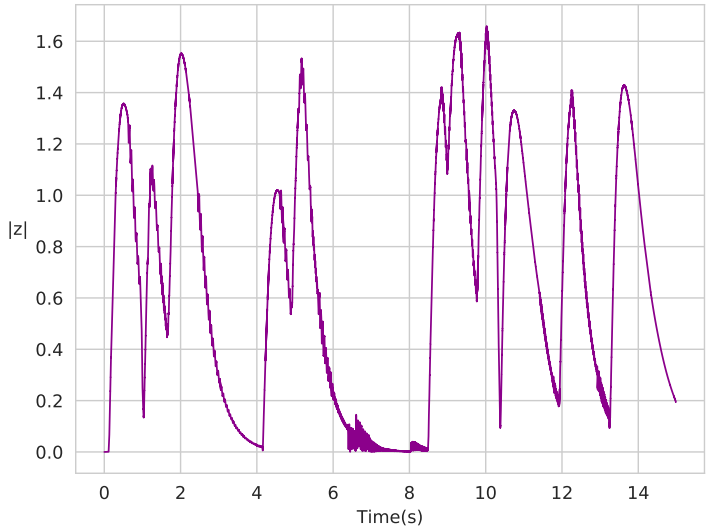
Code extract 3.1 provides pseudocode for backtracking, from current sample `n`, where the function `getResultItem(k,n)` returns the value of sample `n` in channel `k` from the DetectorBank. When the `while` loop exits, the sample number `n` is the onset time.

Evaluation

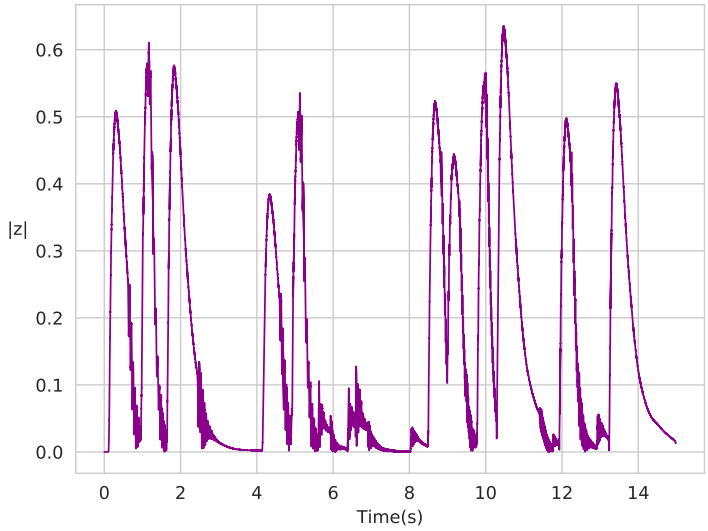
The extent to which this onset detector works is largely dependent on choosing the correct DetectorBank parameters and threshold. Figure 3.2 shows the responses of two detectors at different damping levels, when driven by the test extract *Dream a Little Dream of Me*. A larger damping factor causes faster rise and relaxation times and therefore more clearly defined peaks in the response (at the expense of widening the minimum bandwidth from 1.16 Hz to 4.86 Hz, as presented in Table 2.6). However, the rough locations of the notes are clear from visual inspection of both graphs.

We can also determine thresholds from these graphs: 0.8 and 0.3 would appear to be suitable for Figures 3.2a and 3.2b respectively. These values are roughly half the maximum in each case.

Zooming in on the responses between about 8 and 10 seconds shows two potential problems for the onset detector (see Figure 3.3). First, when the damping is $1 \cdot 10^{-4}$, the note at around 9 seconds is missed because the response has not fallen enough from the previous note to pass the threshold. Second, the responses are oscillatory: there will be threshold-exceeded events that must be ignored as they are due to oscillations during relaxation.

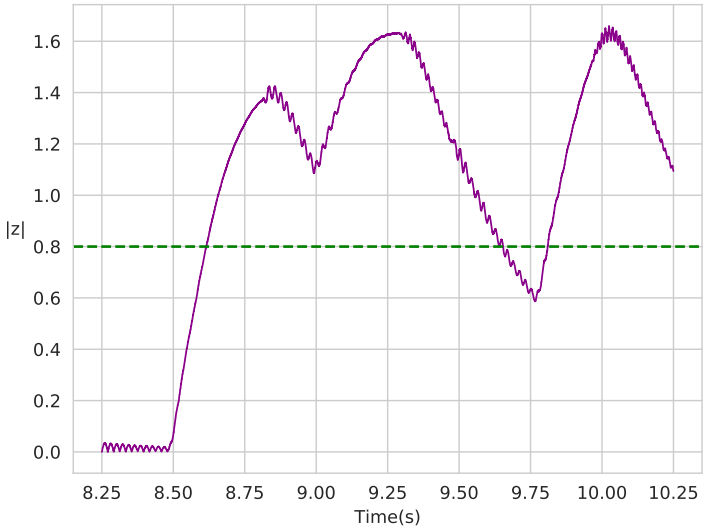


(a) Damping $= 1 \cdot 10^{-4}$

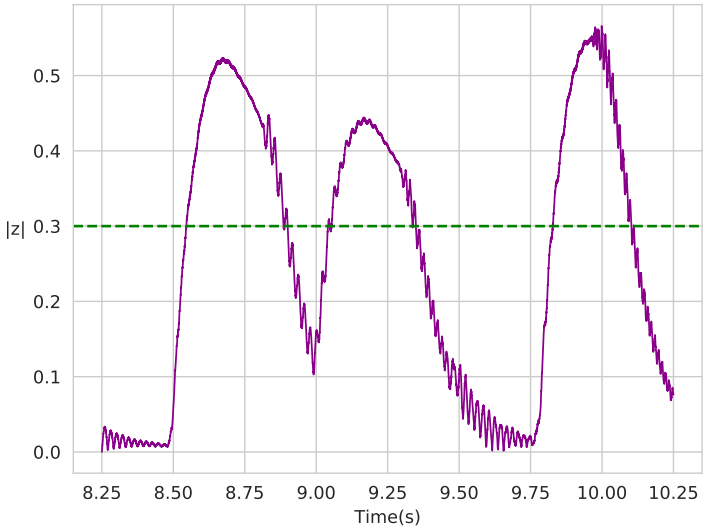


(b) Damping $= 5 \cdot 10^{-4}$

Figure 3.2: Response of detector at 391.995 Hz (G4) to the melody of *Dream a Little Dream of Me* with two different damping factors (a) $1 \cdot 10^{-4}$ and (b) $5 \cdot 10^{-4}$.



(a) Damping $= 1 \cdot 10^{-4}$, threshold $= 0.8$



(b) Damping $= 5 \cdot 10^{-4}$, threshold $= 0.3$

Figure 3.3: Zooming in on responses in Figure 3.2, with the thresholds marked. Three notes occur in the input in this time; at a low damping factor, one of these notes would not be detected, as the relaxation is not fast enough for the threshold to be passed.

An onset detector must be able to address both of these issues: (i) the amplitude of the peaks in the response, corresponding to notes, can vary a lot — in both graphs in Figure 3.2, the maximum peak amplitude is more than 60% larger than the minimum peak amplitude — and (ii) the responses are not smooth.

When testing onset detectors, true positives will be defined as any detection within ±50 ms of the (manually determined) correct time. Although we perceive sounds that occur within 30 ms of each other as simultaneous, this generous range of ±50 ms allows us to gauge whether onset detector is somewhat in the right area or completely wrong, as well as compensating for human error in the manual markup.

The concept of backtracking appears to work quite well, if sub-optimally. With a damping factor of $5 \cdot 10^{-4}$ and a threshold of 0.3, the onset detector can find all notes in the *Dream a Little Dream of Me* melody: 33 notes in 15 seconds, between D4 (293.665 Hz) and B4 (493.883 Hz). The F-measure of this is 100%. 22 of these detections are within ±15 ms of the manually found onset time; the remaining 11 detections are between 16 and 32 ms late.

However, these results cannot be replicated when the damping factor is $1 \cdot 10^{-4}$; the fixed threshold causes many notes to be missed, as well as some false positives. The F-measure drops to 73%. When a threshold of 0.6 is used (lower than that determined by inspection of Figure 3.2a to accommodate the output of all detectors required), only nine of the 22 true positives fall within ±15 ms.

In both of these tests, the backtracking algorithm had a tendency to exit too early. Two onsets are detected at exactly the right time, one is before the correct time and the rest of the detections are late.

Backtracking gives promising results, but it needs to be refined if it is to be an effective part of onset detection. Such improvements will be detailed in Section 3.5 of this chapter, but first we must determine the structure and

features of the OnsetDetector.

3.1.2 Return to the auditory system

Sine tones, generated at various frequencies, were used when characterising the DetectorBank, as knowing the exact form of the input allows us to draw conclusions from the output. However, musical signals are more complex than this. They contain components at multiple frequencies, which can change over the course of the note. Additionally, although there are some instruments which play notes at fixed pitches — pianos or percussion instruments like xylophones and marimbas — there are many capable of creating sounds at any pitch, depending on the instrument itself, as well as the technique of the performer.

Therefore, for woodwind, brass or string instruments, attempting to find a note by using a single detector — tuned to the expected fundamental frequency — may result in events being missed by an OnsetDetector, even though they would be perceived as the correct pitch class by a listener.

The counter this potential problem, we return to the physiology of the auditory system. In audiology, the phenomena of auditory masking and beating are attributed to the behaviour of the basilar membrane, which can be modelled as a series of overlapping bandpass filters (Pickles 2012). The term *critical band* refers to a rectangular filter which is equivalent to the auditory filter.

We will use a slightly different definition of critical band to that found in audiology, as the aim of the project is not to build a model of the auditory system, but rather to detect notes in musical audio. Given a frequency, f_0, the NoteDetector will construct a DetectorBank of n detectors centred around f_0. Together, these detectors will respond to the band of frequencies that could be considered to be in the same pitch class. Therefore, if the user specifies p frequencies, pn detectors must be created. Although this

requires more memory and CPU usage, it is necessary if we wish to create software that can analyse music, rather than simply tones at set frequencies.

The concept of critical bands allows us to regard all the detectors in the band as a single entity, rather than a collection of components. Designing an OnsetDetector which utilises this idea will compress the information in n detector responses down to representation more suited to purpose. The OnsetDetector can then operate unimpeded by superfluous minutiae.

Using critical bands will also be useful for identifying pitch, as the relative strength of response amplitudes within the band could be used to determine the exact frequency sounding at any given moment.

3.1.3 Why not use machine learning?

Although many aspects of this project draw heavily on biological processes, neural networks or similar machine learning techniques are not suitable here. These were considered, but not pursued; the main reason for this decision was the data sets that would be required.

In machine learning terms, the process of identifying what features of the DetectorBank responses correspond to onsets is called *classification learning* (Witten et al. 2017). This is implemented by providing a training set that includes the correct onset times. Definition 2 stated that

> A note is an entity with *at least* an onset time and pitch, where the onset is a single moment in time which marks the beginning of the note. It may also contain intra-note events corresponding to changes within the note.

This suggests that it may be useful for the training set to include data about the timing of events within notes, i.e. the onset times of frequency components that do not correspond to note onsets.

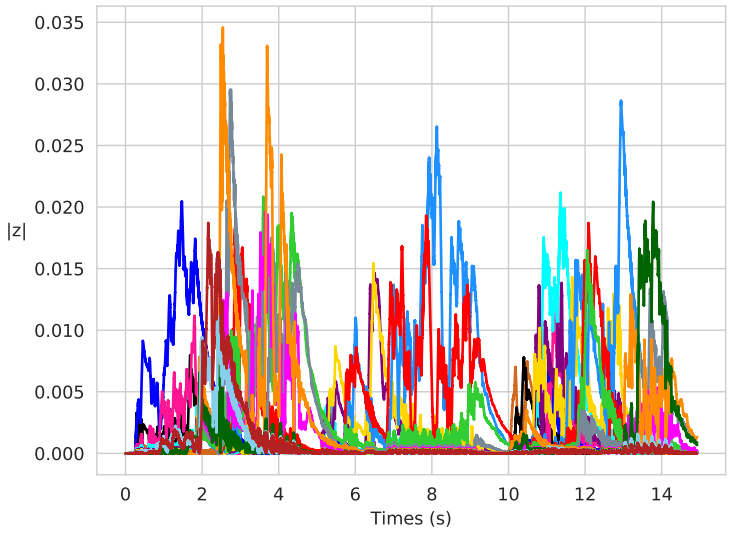
Given that there is so much variation possible in a musical signal, that this would likely require a very large training set to ensure that onsets can

reliably be detected in different contexts. Another independent data set would also be required to evaluate the resulting algorithm. To the author's knowledge, there is no extant data set of note onset times in a large body of samples that includes timings of intra-note events.

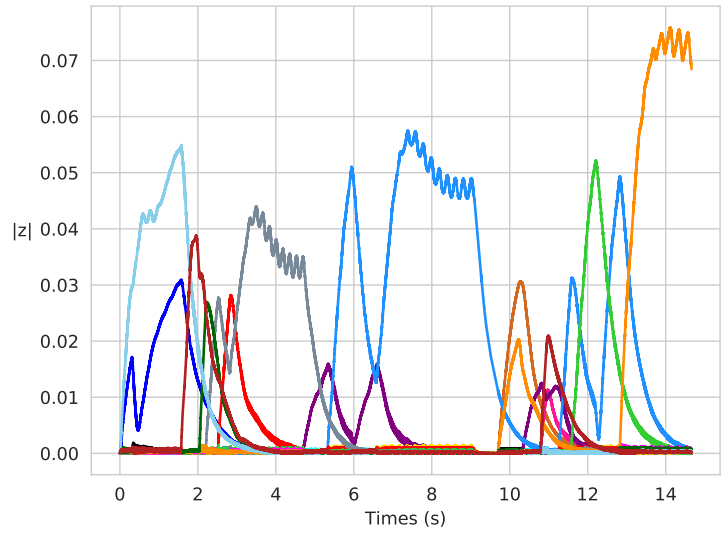
The data set that will ultimately be used in Chapter 4 to evaluate this software comprises notes played on a range of instruments and with a range of playing techniques. Each note is played individually and left to ring out. This means it is suitable for evaluating the accuracy of the detections in a controlled manner and comparing results for different categories of input; however, the ideal training set would include data more like the samples we wish to analyse with the software, i.e. full performances.

The most practical way to obtain a sufficiently large data set, for which both exact frequencies and onset times are known a priori, is to generate tones, with MIDI or a similar process. However, generated tones have significantly different response characteristics from organic musical tones.

This is demonstrated by Figure 3.4, which shows the responses of a DetectorBank, tuned to the fundamental frequencies of the notes G4–A♭5, when driven by the extract from *Before All Things*. In Figure 3.4a, the DetectorBank is driven by the original extract, performed by a singer; Figure 3.4b uses a MIDI rendering of the score to generate the responses. It is apparent from the responses that the MIDI signal is much less complex than the voice: each individual MIDI note is clear, whereas the vocal responses are far busier. For example, the vibrato in the vocal extract is much wider than the MIDI; this is particularly apparent in the notes from six to ten seconds. Is is clear from the MIDI responses that the two notes here are B♭4 followed by C5. Some vibrato can be seen from the small oscillations in the C5 response. However, the same period in the vocal sample is far busier. For both notes, it can be seen that the vibrato reaches up to the neighbouring semitone.



(a)



(b)

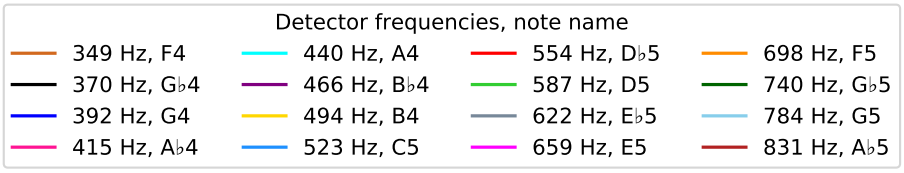


Figure 3.4: The first 15 seconds of *Before All Things*, with detectors tuned to the fundamentals of the notes from F4–A♭5. In (a) the melody is sung; in (b) a MIDI rendering of the melody is used the drive the DetectorBank. The complexity of the voice signal compared with the MIDI rendering is apparent.

This difference between generated signals and notes played by a performer is not a problem when characterising the system — in this instance, we want to know how the system responds in ideal or controlled situations — but an algorithm trained to find MIDI notes will not necessarily be able to find 'real' notes.

3.1.4 Outline of an OnsetDetector

The basic structure of an OnsetDetector comprises two stages. The first should be a 'rough' stage, which operates on every input sample. Most samples will not represent an onset, so the first stage should be able to analyse its input without expending vast amount of computation analysing samples which will be rejected anyway. There should be no false negatives in the output of this stage, although false positives are acceptable.

The second stage will be activated whenever the first stage finds a possible onset. It will examine the responses in more detail to either verify or reject the initial detection. If verified, this should return an exact onset time.

Three approaches to stage one were tried; the first two were unsuccessful. All three methods will be outlined here and the shortcomings of the first two will be discussed.

3.2 Sum gradient

The first idea prototyped and tested for stage one was to look at the rate of change of the responses. This was implemented by calculating a rough gradient over M samples. The gradient of each response in the band is summed at every sample. For this reason, we refer to this as the 'sum gradient' technique. Equation (3.1) shows this, where z is an $N \times K$ array

of DetectorBank output samples.

$$x = \sum_n \sum_k \frac{z[k][n] - z[k][n-M]}{M} \tag{3.1}$$

Although this approach may seem crude, it has the overwhelming benefit is that it satisfies the requirement that the first stage of an onset detector identifies potential onsets with minimal computational requirements and latency, as is shown in the evaluation in Section 3.2.1.

This method allowed us to regard the responses as states rather than numerical values. The two basic states were 'rise' and 'fall': when the responses are rising, there may be a note onset; when they are falling, the note may be ending. Rise or fall is determined on a sample-by-sample basis. When these are examined over time, we can broaden the states out to include steady states (either during a note or between notes).

The state of the band at sample n is given by the sign of the sum gradient at that sample. Regarding the magnitude of this as a confidence allows for more nuance in the state decisions: a large positive (or negative) value means the band is very likely increasing (or decreasing); a small value means the band may be in a steady state.

Regions where the state is rise or fall for a significant period of time (e.g. 30 ms) are then regarded as possible onsets or offsets by stage one. This then alerts stage two. As stated in Section 3.1.4, the job of stage two is to analyse the responses at this time more closely. For this prototype, a very basic stage two was implemented, which considered the confidence values immediately before the onset time given by stage one. If a certain proportion of these exceeded a given threshold, the onset was verified. Otherwise, it was rejected.

3.2.1 Evaluation

The method has a number of advantages. It makes effective use of the critical band concept: the responses of multiple detectors are reduced to a single value at every sample. The required memory for each band is kept to a minimum by considering the values N samples ago (where N will typically be a small proportion of the sample rate); samples older than this can be discarded. Also, the sum gradient algorithm requires only a small number of operations, thus deferring heavy computation until stage two is called. However, there are disadvantages which become apparent when considering an example.

The output of the prototype can be seen in Figure 3.5. A critical band, comprising 21 detectors at 1 Hz increments around 391.995 Hz, was set up and driven by *Dream a Little Dream of Me*: the same input and centre frequency as used in the Evaluation of the initial idea presented earlier in this chapter. The gradient was calculated over $N = 1000$ samples.

The rise and fall states are shown in this figure by dark and light grey regions in the background. Onsets, initially found by stage one then verified by stage two, are marked with green lines. This seems to have been successful, but there is one false negative (shown by a dashed red line). This onset was also missed when the Initial idea of thresholding and backtracking was tested, but this time it is due to the note beginning when the detectors are still in relaxation from the previous note, as can be seen in Figure 3.6. The magnitude of the sum gradient is smaller than it would have been if the responses were starting from zero, so the confidence at this time does not meet the threshold.

This is an insurmountable problem for this implementation of the OnsetDetector. Stage two would have to be implemented in ways which would negate the advantages. Despite this, from Figure 3.5 (and others like it) we can see that the offset periods are very clear: the long 'fall' states are obvi-

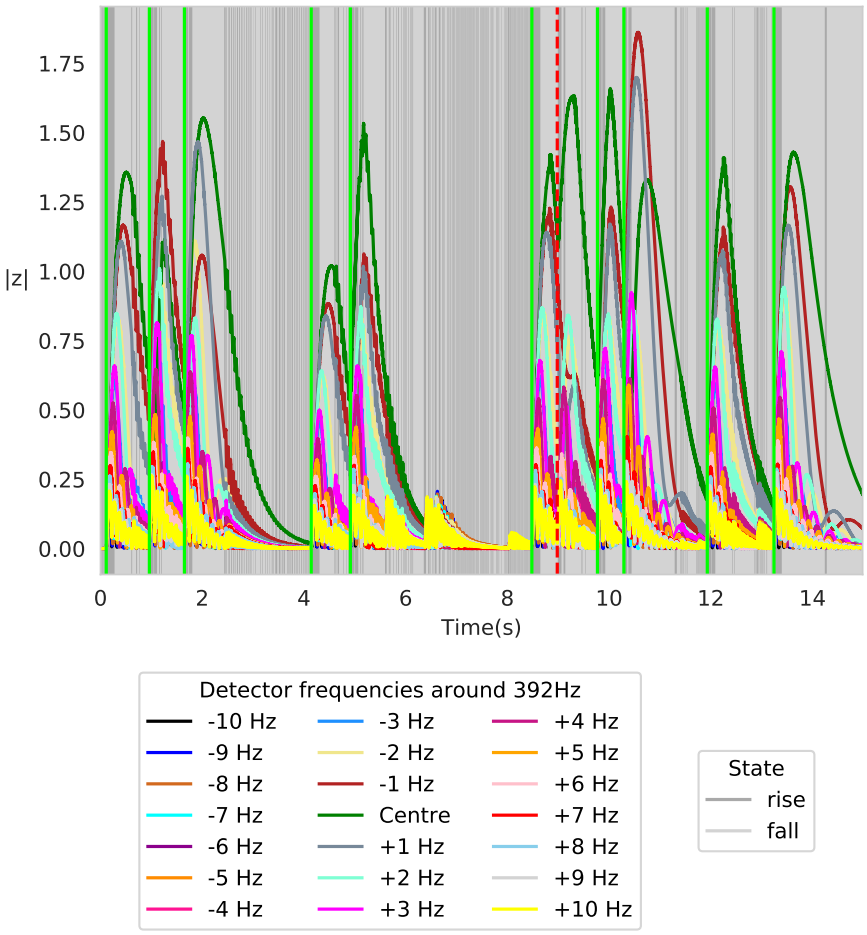


Figure 3.5: *Dream a Little Dreams of Me* melody, with detectors tuned to critical band frequencies around G4 (391.995 Hz). Onsets found by the sum gradient method are marked in green; the dashed red line marks a missed onset. The background colours dark or light grey, shows the whether sum gradient is positive or negative on a sample-by-sample basis, i.e. whether the state of the band is 'rise' or 'fall' at that point.

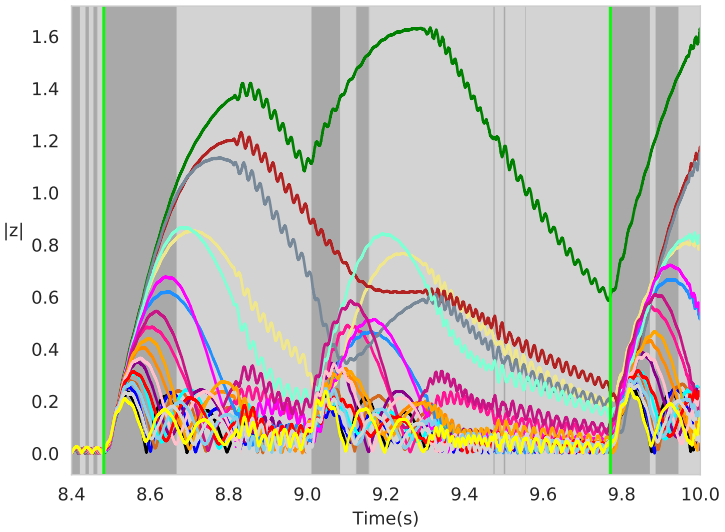


Figure 3.6: Zooming in on the DetectorBank responses shown in Figure 3.5, at the region where the onset shortly before 9 seconds was missed.

ous. Although the sum gradient method is problematic, it could potentially be useful if a method for offset detection is required. However, it must be abandoned as an approach to onset detection.

3.3 Hough transform

The next idea developed approached the problem in a completely different manner. Instead of calculating the rate of change at every sample, we look for the shape of a note in the DetectorBank output. The basic shape of responses to a note is known: a line, which rapidly increases to its peak, then decays. Figure 3.7 shows a detector response to a sung note, overlaid with lines which approximate the onset and offset shape: a straight line and a decaying exponential. An algorithm that can detect the appearance of this shape in the responses may be a suitable stage one of onset detection.

The Hough transform is a technique used to find patterns in data. It is often used in computer vision to find a shape in an image, regardless of size, location and rotation (Nixon & Aguado 2012), but has also been used to

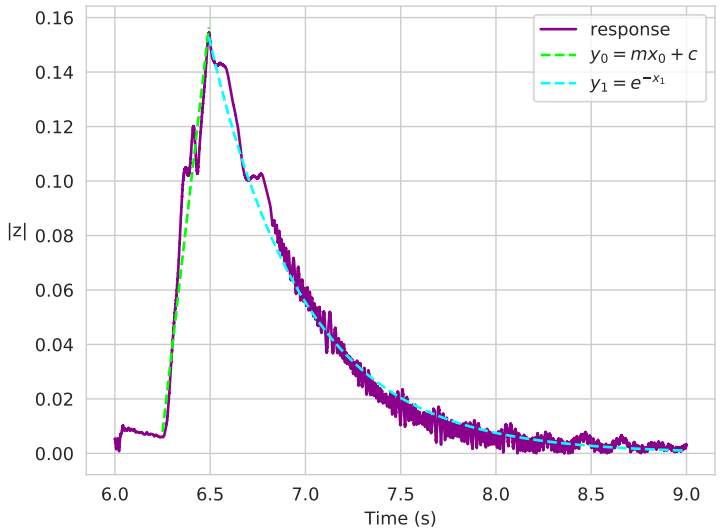


Figure 3.7: Detector response to a sung note (purple), with lines fit to the rise and decay.

analyse audio signals (Dennis et al. 2015). There are versions of the Hough transform for finding various geometric shapes, as well as arbitrary shapes. The Hough transform for straight lines is suitable for matching the onset and could be used for the offset too, as the log of a decaying exponential is a straight line, as shown in Figure 3.7.

The Hough transform is a good candidate for this task, as it is tolerant of noise in the signal and gaps that may occur if there is a glitch in the waveform. It is also generalisable to arbitrary shapes, if straight lines do not turn out to be the ideal shape to search for in the data.

Section 3.3.1 introduces the Hough transform for lines, as it is used in image processing. Sections 3.3.2 and 3.3.3 then present ideas for adapting this to operate on DetectorBank output and shortcomings of this technique.

3.3.1 Hough transform for lines

The polar equation of a line is integral to the Hough transform. It is defined as

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{3.2}$$

and can be found by drawing a line which is perpendicular to the original line and goes through the origin. The length of this line is ρ and the angle between it and the horizontal axis is θ.

For example, Figure 3.8 shows a blue line, defined by $y = x/2 - 1$. The perpendicular line to the origin is shown as a red dashed line. The length of the red line (ρ) and the angle between it and the x-axis (θ) define the blue line. In this case, $\rho = 0.894$ and $\theta = 1.107$.

From simple geometry, the relationship between the linear equation of a straight line, $y = mx + c$, and the polar equation, $\rho = x\cos(\theta) + y\sin(\theta)$, can be derived:

$$m = \frac{1}{\tan\theta} \tag{3.3}$$

and

$$c = \frac{\rho}{\sin\theta} \tag{3.4}$$

From this, we can see that ρ calculated from Figure 3.8 should be negative. By convention, $0 \leq \theta < \pi$; the sign of ρ shows whether the angle is being measured above or below the x-axis (see Figure 3.9).

In order to find the Hough transform, we first calculate ρ for a range of angles at every point of interest in the input. As a (θ, ρ) pair uniquely defines a straight line, points which form a straight line will have the same ρ values for a particular angle. Therefore, the ρ and θ values which occur most frequently correspond to the lines which are present in the input.

In order to find the Hough transform of an image, it must be preprocessed by an edge detector, as only the pixels corresponding to a discontinuities in brightness need to be considered in the calculation.

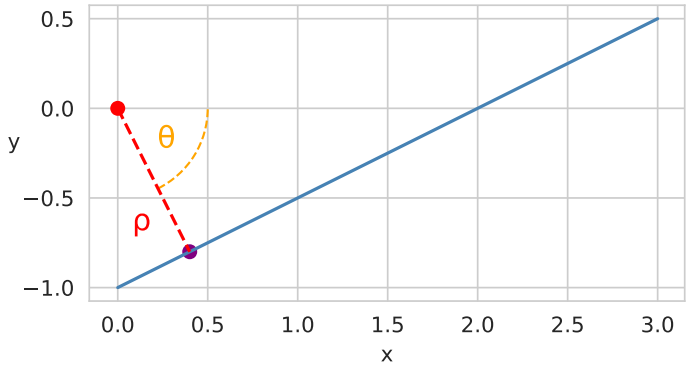


Figure 3.8: Line $y = x/2 - 1$ (blue) with perpendicular line to the origin (red). The magnitude of the red line (ρ) is -0.894 and the angle between it and the x-axis (θ) is 1.107 radians.

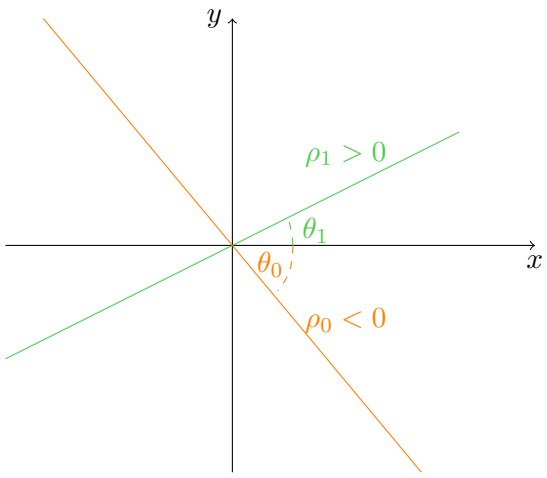


Figure 3.9: Illustration of different ρ sign when θ is calculated clockwise (orange) or anticlockwise (green).

The angles at which ρ will be found must be chosen by the user. Choosing a greater number of angles between 0 and π may in greater accuracy in the output. The maximum possible ρ is $\sqrt{\text{width}^2 + \text{height}^2}$, where width and height are in pixels. Together, the resolution of θ and the value of $\rho_{\max}$ determine the resource requirement and the accuracy of the output.

We then create a 2D array (known as the accumulator), initialised with zeros, to store the final output. The width and height of this array should be the chosen number of angles and $2\rho_{\max}$ respectively, where $\rho_{\max}$ has been rounded to the nearest integer and doubled to accommodate positive or negative values of ρ. At every pixel of interest in the input, we cycle through the range of angles. For each angle θ at every point (x, y), ρ is calculated using Equation (3.2) and rounded to the nearest integer. The value in the accumulator corresponding to the tested θ and calculated ρ is incremented.

On completion, the accumulator contains maxima at the (θ, ρ) positions of the associated lines in the input.

3.3.2 Using the Hough transform with the DetectorBank

The Hough transform returns parameters of lines. When used to analyse DetectorBank output, each set of parameters must then be translated into an onset time. The points at which a line described by (θ, ρ) overlap with the input can be found (i.e. the parts of the response θ and ρ actually represent). Then the response can be traced backwards from here to the point where it began to react. The time at this point is the output of stage one.

Applying the Hough transform to the DetectorBank output is a subtly different problem from applying it to an image. The points in the response are equivalent to the reduced set of points obtained by applying an edge detector to an image. When calculating ρ for an image, the pixels' x and y

coordinates are used. These are discrete integer values, and, although the output of the DetectorBank is sampled in the time domain, the amplitude values are continuous (and often very small). If the sample number (x) is orders of magnitude greater than the response amplitude (y), the $x \cos\theta$ term will almost always dominate. In order to apply the Hough transform to the DetectorBank output, the response must be quantised in amplitude, then referred to by index in the calculation. The number of amplitude quantisation steps and the length of the input signal determine $\rho_{\max}$.

For the purposes of note onset detection, we only need to find lines which satisfy the following criteria: (i) they have a steep, positive gradient and (ii) they cross the x-axis after zero (i.e. have a negative y-intercept). This means the perpendicular line to origin should be similar to the orange line in Figure 3.9. ρ is negative, therefore the accumulator size can be halved, and the range of angle to be searched can be reduced, to at least $0 \leq \theta < \pi/2$ and possibly further.

These optimisations represent significant gains in efficiency, but unfortunately the Hough transform has drawbacks which suggested its further development will not be productive.

3.3.3 Shortcomings

Peak picking in the accumulator will not be a simple task when the input is DetectorBank output, rather than an image. Initial tests, for instance the accumulator shown in Figure 3.10, suggest there are not clear discrete maxima, but clusters of peaks, representing short segments of straight line in the input. Finding the whole line (or lines) in the input requires a method of partitioning the accumulator, then reducing each of these groups of peaks to a single (θ, ρ) pair. From this pair, the onset time will be calculated. We know that the maximum error for an onset time is 30 ms; however, the maximum allowable error in (θ, ρ) required for the onset time to fall within

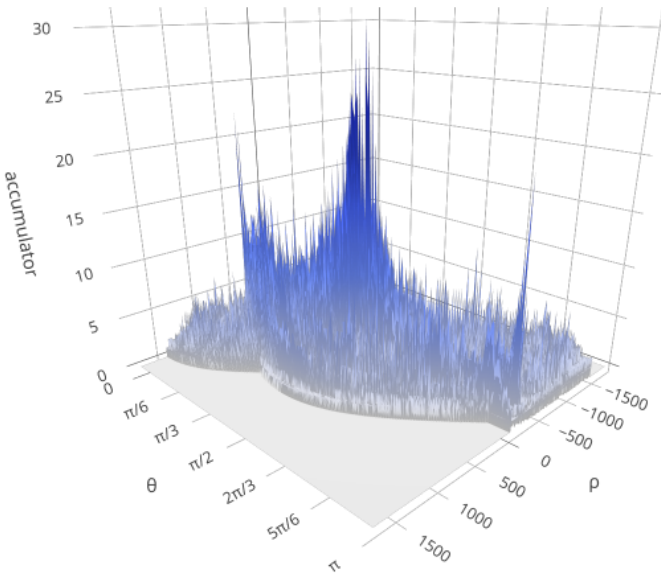


Figure 3.10: Accumulator generated by taking the Hough transform of the detector response to a sung note shown in Figure 3.7.

this window is, as yet, unknown. Additionally, the number of notes, i.e. number of maxima, present in any given input may not be known a priori, so determining line parameters from the accumulator represents a significant difficulty.

The form of both the input and output must also be considered. DetectorBank output is a 2D array, k channels by N samples. Reducing the number of samples to process reduces $\rho_{\max}$ and thus the accumulator size. Subsampling the responses will achieve this, as will providing the input in short buffers. These buffers would have to overlap so that onsets which occur on the edges would not be missed. To generate Figure 3.10, the three second extract from the responses was subsampled by a factor of 100. However, even with this, the accumulator is an array of 180×3506 values, the vast majority of which are irrelevant to the task at hand. Further development of this technique for onset detection would likely require an new implementation of the Hough transform to be devised, in order to store the

most frequently occurring θ and ρ values in a more efficient manner and thus reduce the memory needed.

Additionally, the prototype of this method did not utilise the idea of a critical band as a single entity; a single detector response was given as the input. A suitable method for generating a 1D input from k responses would be required, as running the Hough transform on every detector in a band is inefficient.

Ultimately, whilst the Hough transform is a useful tool for finding straight lines and other shapes in many applications, it is not suited to this task. It would require significantly more development to be an effective stage one of onset detection.

3.4 Mean log

All methods considered for note detection so far proved to have intractable problems. A new approach is needed.

When we look at the graph of responses of a critical band, we can generally identify areas which might be notes without too much difficulty. For example, examination of Figure 3.11 suggests there are notes shortly after six seconds and around eleven seconds. There are lower amplitude peaks around two, five and thirteen seconds: these could be notes in this band or neighbouring bands 'spilling over' into this one.

The initial idea behind this project is biologically inspired: creating sympathetic resonators using the Hopf bifurcation, as it models how the inner ear responds to sound. A similar motivation forms the basis of this method for note detection: we want to mimic what the brain is doing when it finds potential notes in Figure 3.11.

In the process of identifying notes in DetectorBank output by eye, we (i) compress a lot of data to make big picture observations and (ii) consider a wide context: all detectors simultaneously, over a significant time. These

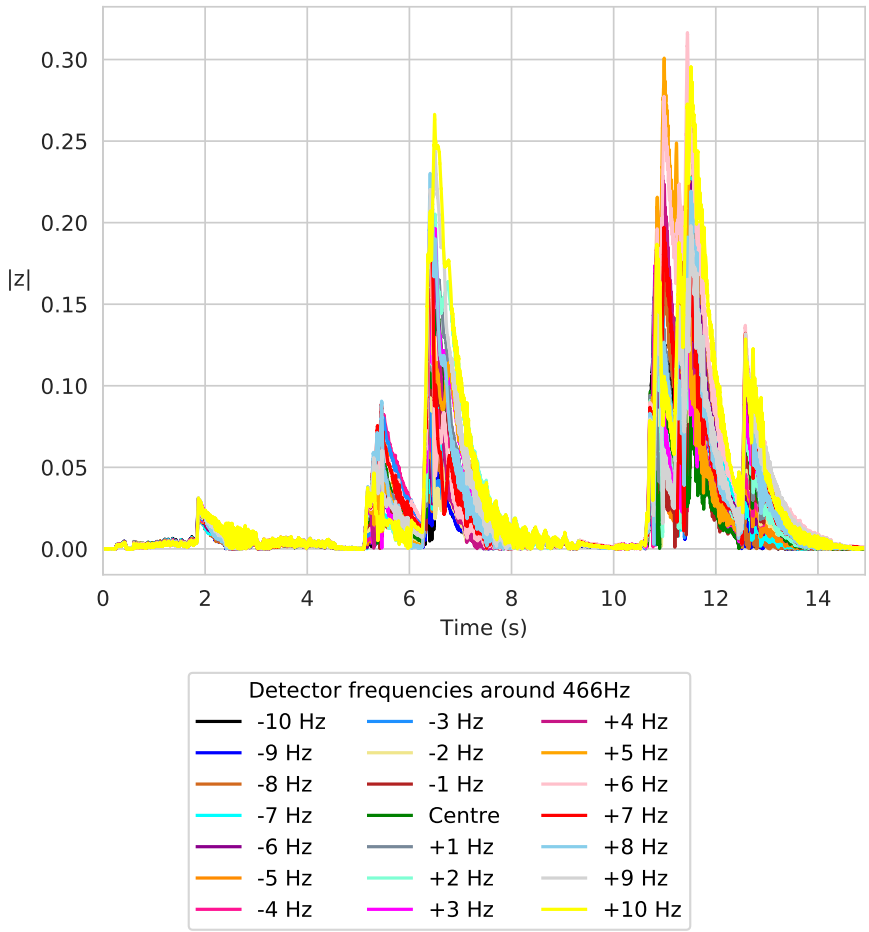


Figure 3.11: Band of 21 detectors around 466.164 Hz (B♭). The input signal is the first 15 seconds of the soprano part of *Before All Things*, by Graham Hair, recorded at sample rate of 48 kHz.

two strategies allow us to focus on a response when its amplitude is greater than others and ignore it when its amplitude is significantly lower.

Replicating this automatically provides stage one of an onset detector, as outlined in Section 3.1.4.

3.4.1 Proof of concept

This was implemented by splitting the responses of an entire band into 30 ms segments, then taking the mean of each segment. If at least four consecutive means are increasing — and the final value in this consecutive run is greater than a given threshold — this may be an onset.

Code extract 3.2 presents an algorithm for this. `getSegAvg()` is a function that returns the average of the next 30 ms segment; `findExactTime()` is a function which performs stage two of onset detection and returns a boolean and an integer: whether the onset was verified and the sample number of the verified onset.

```
count = 0       # no. of consecutively increasing segments
seg_count = 0   # current segment number
last = 0        # mean of previous segment
onsets = []     # list to store verfied onsets

while True:

    # get mean of next segment
    current = getSegAvg()

    # whilst current >= last, keep getting averages
    # and count how many have been increasing
    if current >= last:
        count += 1

    # otherwise, if at least 4 have been increasing...
    # (count starts from 0, so we say >=3)
    else:
        if count >= 3 and last >= threshold:
            # ...calculate sample numbers of the beginning
            # and end of the increasing run
            start = (seg_count-count)*seg_len
            stop = (seg_count-1)*seg_len

```

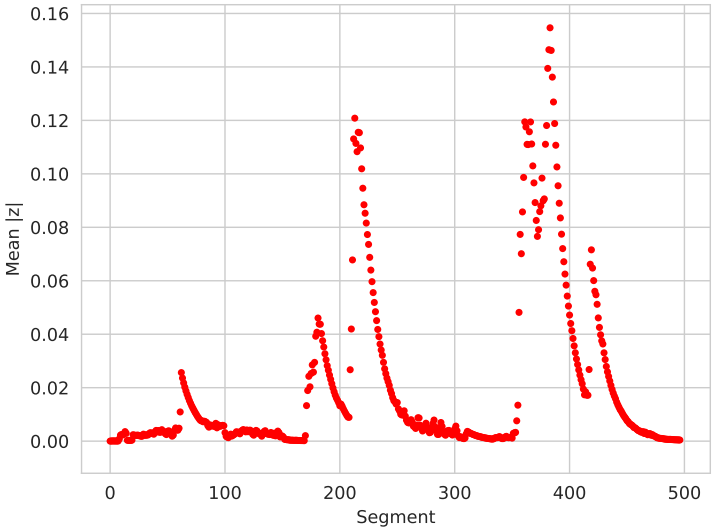
```
25              # verify or reject
26              verified, onset = findExactTime(start, stop)
27              # if verified, put in onsets list
28              if verified:
29                  onsets.append(onset)
30
31          # reset count
32          count = 0
33
34      # roll values round for next segment
35      last = current
36      seg_count += 1
```

Code Extract 3.2: Onset detection based on segment means

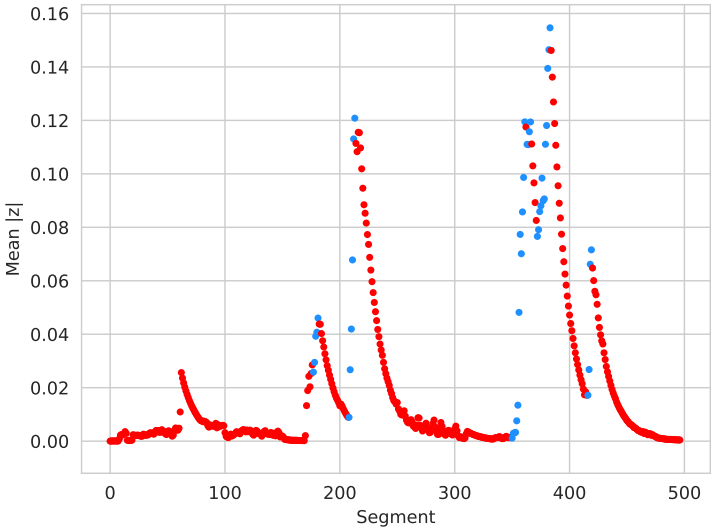
Figure 3.12a shows the average of the responses in Figure 3.11 over 30 ms segments. Now there are 500 data points to analyse, rather than the $21 \times 15 \times 48000 = 1.512 \cdot 10^7$ points in the critical band plotted in Figure 3.11.

Figure 3.12b highlights points where the average is increasing for four or more consecutive segments and the last segment exceeds a threshold of 0.05. In this example, there are eight such runs of points, covering the following segment indices:

- 177, 178, 179, 180, 181;
- 208, 209, 210, 211, 212, 213;
- 350, 351, 352, 353, 354, 355, 356, 357;
- 358, 359, 360, 361;
- 363, 364, 365, 366;
- 372, 373, 374, 375, 376;
- 377, 378, 379, 380, 381, 382, 383; and
- 416, 417, 418, 419.



(a)



(b)

Figure 3.12: Average $|z|$ values for the band of detectors shown in Figure 3.11. Each point marks the mean of the whole band over a 30 ms period. In (b), four or more successively increasing points are shown in blue.

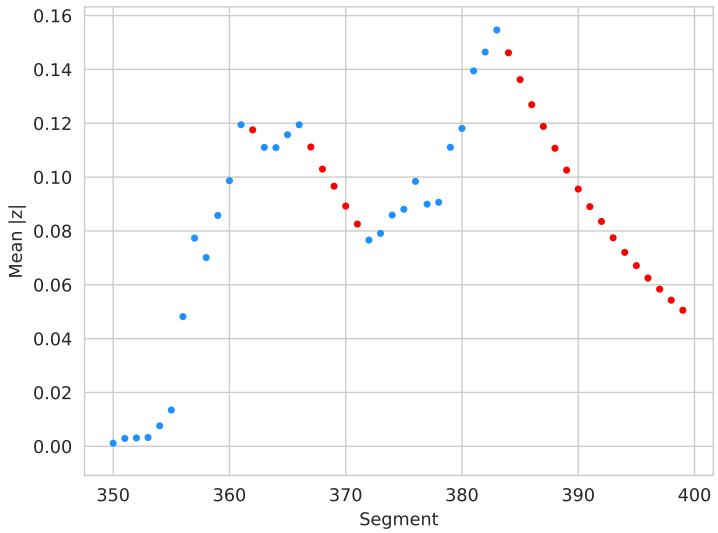
The first, second and last of these each correspond to a note; the other five sets of points correspond to two notes around eleven seconds.

Figure 3.13 zooms in on this area. Figure 3.13a shows the averages of segments 350–400 and Figure 3.13b shows the band response from which these are derived.

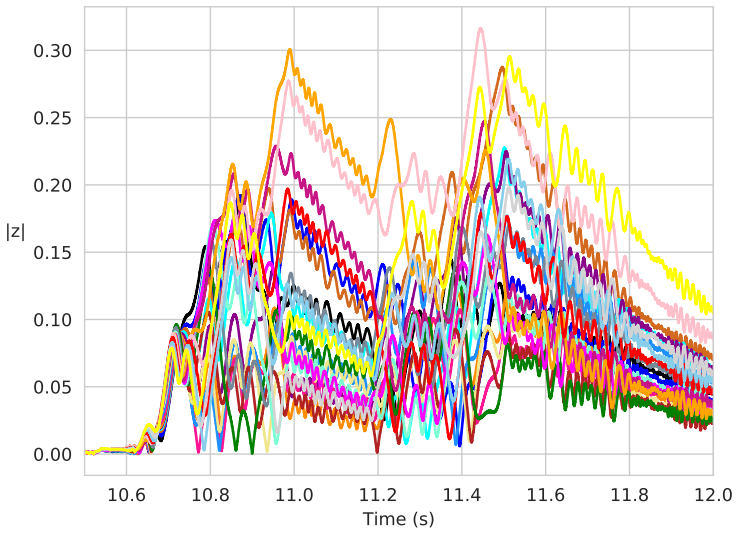
Examining these graphs, we can see why five areas of increase were found, rather than two. Runs of increasing segments 350–361 and 372–383 are each broken in two by a decrease in the average (at segments 358 and 377 respectively). The four increasing segments 363–366 are separated from the previous run of increasing values by a single segment. We can see from Figure 3.13b that these are not due to a fault in the algorithm: they are accurate reflections of the input data. There are two notes here (manually found to start at 10.592 s and 11.151 s). As this is a sung note, fluctuations in pitch are not unexpected. The occasional decreases in average amplitude are due to changing pitch in the input. These detections should not be dismissed as erroneous or inexact; we are not attempting to transcribe the input and assign a single pitch and duration to the whole note. Rather, we are attempting to measure what is happening when a musician performs. Pitch changes over the course of a note are an important part of this.

Evaluation

When this is tested with a short, monophonic piano sample (the melody to *Dream a Little Dream of Me* again) as the input, a damping factor of $1 \cdot 10^{-4}$ and a threshold of 0.2, all but one onset is found within 50 ms, resulting in a precision, recall and F-measure which are all 96%. This time window was also used in the Evaluation of the initial prototype and is wider than the 30 ms interval within which sounds will be perceived as simultaneous, but it is possible, given the inexact nature of manually selecting single instants from millions of samples, that the hand-annotated onsets contain errors.



(a) Segments 350–400 of Figure 3.12. Consecutively increasing segments are again shown in blue.



(b) Whole band responses, from which the averages in (a) were calculated.

Figure 3.13

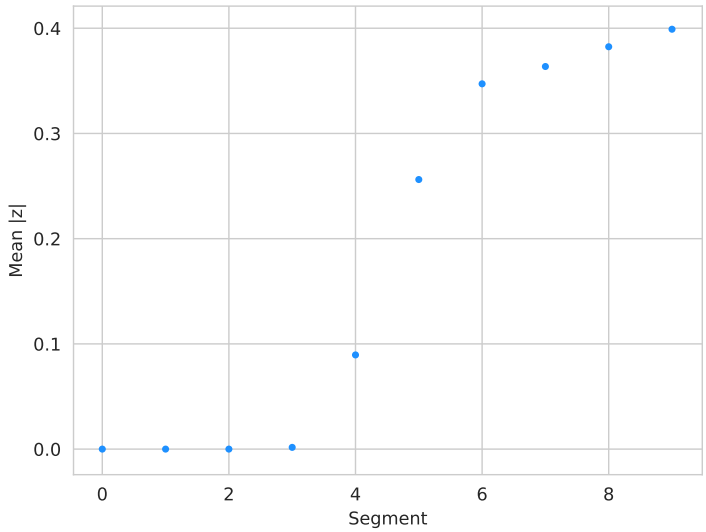


Figure 3.14: First ten segment averages for *Dream a Little Dream of Me*.

Setting a wider interval for detections to be considered correct should help to offset human error in the markup process.

The largest time delay for a detection in the output of this algorithm is 20 ms and the largest time advance is 14 ms. In total, six detections are more than 15 ms from the correct time.

The first note in the audio — a G4 which occurs at 101 ms — is the one missed by the algorithm, which instead returns an onset time of 0 seconds. The averages for the first ten 30 ms segments in the band around 391.995 Hz are plotted in Figure 3.14. It can be seen that the first four are all blue — indicating that the values are increasing, although they are all very close to zero — hence an onset time of 0 seconds is returned. The actual onset appears to be between segments 3 and 4 (i.e. between 90 and 120 ms). However, this is not a cause for concern, as the precision of detections like these will be improved by stage two.

This method gives promising results from initial tests. It also meets every criterion for success discussed so far in this chapter: by identifying areas where the average is increasing by any amount (and setting a threshold)

it side-steps problems associated with the variation in amplitude; taking the average also means the oscillatory nature of the responses is also not an issue here; it uses critical bands to its advantage, vastly reducing the amount of data which has to be kept in memory; and it employs a very simple algorithm.

3.4.2 Further improvements

Detector spacing

Heretofore, crude critical bands have been constructed with 21 detectors spaced at 1 Hz intervals. This band size and detector spacing were both chosen arbitrarily, as the initial tests were designed to assess whether the averaging technique was suitable.

Using 21 detectors at 1 Hz increments around the centre frequency has not compromised the tests carried out on *Dream a Little Dream of Me*, as the frequencies of interest have been within one octave: D4 to B4, corresponding to 293.665 Hz to 493.883 Hz. At the lower end, the critical bands overlap; at the higher end there is a gap of up to 7.7 Hz between adjacent bands. This gap has been inconsequential in these tests, as the instrument used to record the excerpt was a digital piano, so there is very little deviation from the ideal fundamental frequency.

Nevertheless, values for band size and spacing which are appropriate for all inputs must be found.

In music, fundamental frequencies are not linearly spaced. This suggests that a more apt method of determining the frequencies in a band of n detectors around centre frequency f_0 is to vary the spacing logarithmically: $f_k = f_0 \cdot 2^{k/(12n)}$, where f_k is the kth semitone from f_0.

However, a constant band size leads to detectors close together at low frequencies and far apart at high frequencies. The spectrum between the fundamental frequencies is not uniformly covered, and therefore some notes

may be missed by the OnsetDetector.

Instead of fixing the band size and varying spacing, a more even spread of detectors across the spectrum can be achieved by placing detectors at intervals according to their bandwidth until the boundary of the neighbouring pitch class is met.

Given that detector bandwidth bears no relation to the spacing of musical fundamental frequencies, the exact point between two semitones is unlikely to be met. Detector frequencies should therefore be generated until they are sufficiently close to the boundary. If the maximum gap at each end of a critical band is defined as being within a quarter of a bandwidth, then the maximum gap between two bands is half a bandwidth.

```
1  freq = []          # empty list to store frequencies
2  f = f0             # start calculating at centre frequency
3  f1 = f0*2**(1/24) # stop frequency (half a semitone up)
4
5  # difference between stop frequency 'f1' and current 'f'
6  diff = f1-f
7
8  # Run until difference between 'f1' and 'f' is at a minimum
9  # or until 'f' is within bw/4 of stop value 'f1'
10 while f1-f <= diff and f1-f > bw/4:
11     # before new 'f', get 'diff' for next time round loop
12     diff = f1-f
13     # generate the next frequency
14     f += bw
15     # put f in the list
16     freq.append(f)
```

Code Extract 3.3: Generating the upper half of a critical band

Code extract 3.3 demonstrates this method for calculating the critical band frequencies between the centre frequency, `f0`, and the point halfway to the semitone above, `f1`, for detectors with bandwidth `bw`. Frequencies are generated at increments of `bw`, until reaching a value within a quarter of a bandwidth of the 'stop' frequency, `f1`. In practice, when detector frequency is incremented by `bw`, it may not get within `bw/4` of this boundary frequency. In this case, it will stop when the difference between the current frequency

Table 3.2: Minimum, maximum and mean gaps between critical bands at different bandwidths

Bandwidth (Hz)	Min (Hz)	Max (Hz)	Mean (Hz)
0.922	0.016	0.339	0.131
1.832	0.007	0.702	0.259
2.752	0.200	1.037	0.612
3.660	0.017	1.732	0.567
4.860	0.080	2.313	1.159

Table 3.3: Minimum, maximum and mean overlaps between critical bands at different bandwidths

Bandwidth (Hz)	Min (Hz)	Max (Hz)	Mean (Hz)
0.922	0.009	1.238	0.546
1.832	0.012	2.428	1.269
2.752	0.004	3.869	1.940
3.660	0.186	5.375	2.895
4.860	0.470	7.124	3.781

`f` and boundary frequency `f1` is at a minimum. The lower half of the band can be created in a similar fashion.

Typical sizes of gaps and overlaps between adjacent bands can be found by creating critical bands covering fundamental frequencies from A0 to C8 (i.e. 88-key piano) at five bandwidths, corresponding to the minimum bandwidths at different damping factors, which were experimentally determined in Chapter 2 and listed in Table 2.6.

The minimum, maximum and mean values of gaps and overlaps are given in Tables 3.2 and 3.3 respectively. All of the maximum gaps in Table 3.2 are less than half the bandwidth and all mean gaps are within a quarter of a bandwidth. The mean overlaps are larger than the mean gaps, but are all still within a bandwidth.

Perceptual loudness

Perception of the loudness of a sound is highly subjective, and hence difficult to measure, but an often-used approximation is that a 10 dB increase in

Table 3.4: Recall and precision for four piano extracts, using 30 ms segments

Test audio	Recall	Precision
Dream	87%	100%
Alice	94%	68%
Swan Lake 1	62%	25%
Swan Lake 2	86%	54%

Table 3.5: Recall and precision for four piano extracts, using 20 ms segments

Test audio	Recall	Precision
Dream	100%	84%
Alice	100%	50%
Swan Lake 1	68%	19%
Swan Lake 2	97%	44%

intensity corresponds to a doubling of loudness (Pickles 2012).

This relationship between the perceived loudness, L, of a sound and its intensity, I, can be represented by Stevens' law, $L = kI^a$, where k is a constant and the exponent a is $\log_{10}(2)$. Although somewhat controversial in its simplicity, this law provides a useful concept for improving the averaging algorithm, as it tells us that $\log(L) \propto \log(I)$.

By returning the mean of the log of the band's responses, rather than simply the mean, from `getSegAvg` and taking the log of the threshold, we can mimic the compression of loudness that occurs in the auditory system.

Note that this idea also appears in the backtracking algorithm in code extract 3.1. This is also the reason that we refer to this idea as the 'mean log' method.

More true positives

When the two changes given above are implemented, the number of true positives in all test audio excerpts does not reach 100%, as detailed in the 'recall' values[1] in Table 3.4.

[1] Precision and recall were defined in Chapter 1, equations 1.9 and 1.10 as Precision = TP/(TP + FN) and Recall = TP/(TP + FN).

The segment size of 30 ms was chosen somewhat arbitrarily. Reducing the size to 20 ms increases the number of detections, both true and false positives, in several test audio extracts (see Table 3.5). In *Dream* and *Alice*, all onsets are now found. In *Swan Lake* excerpt 2, one onset is missed because a detection is 51 ms early. This is the kind of error the second stage will be tasked with fixing. In *Swan Lake* excerpt 1, a number of onsets in the lowest frequencies are still being missed, suggesting that improvements must be made for frequencies below 75 Hz.

Fewer false positives

This change to the segment size obviously reduces the total time over which the mean has to be increasing for a potential onset to be detected. This results in a rise in false positives and corresponding drop in precision.

Some of these detections are immediately identifiable as false positives when inspecting graphs of the mean log by eye. For example, Figure 3.15 shows part of the mean log of the 392 Hz band response to *Dream*. It correctly identifies a region of increase corresponding to the note beginning at 4.14 s; however, it finds another period of increase which is not an onset.

This pattern — a correct detection followed by an erroneous one during the note — is found repeatedly in the test results.

These can be identified automatically by comparing the mean log value of the last segment in the run with the first. If there is not a significant increase, the potential onset is rejected. In these tests, the value chosen was $\log(2)$.

Employing this as an additional criterion (henceforth referred to as the 'last-first' criterion) for calling `findExactTime()` — along with the requirement for four or more segments to have been increasing and the last one to have exceeded a threshold — reduces the number of false positives before the second stage is utilised.

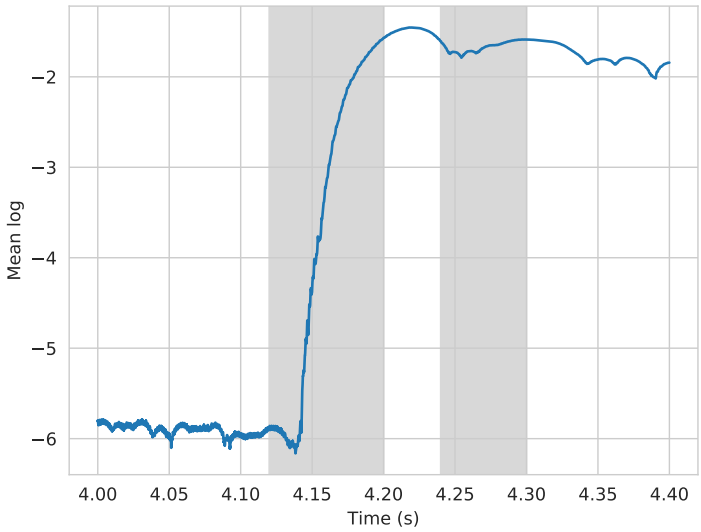


Figure 3.15: In the mean log, shown by the blue line, two runs of three or more increasing segments are found, here denoted by grey regions. The first of these is a true positive; the second a false positive.

It is important to state that this reduces the number of detections that do not correspond to hand-annotated onsets. This does not necessarily mean the detections are erroneous: they occur due to changes in the DetectorBank responses and so may reflect legitimate changes in the input audio. However, the data with which these OnsetDetector prototypes are being tested contains only information about note onsets. Further investigation is required in order to determine whether inclusion of this 'last-first' criterion improves the overall quality of results when intra-note detections can be taken into account. Chapter 4 details the results of rigorous OnsetDetector testing, both with and without the 'last-first' criterion.

In addition to these detections due to small changes in the mean log, inspection of the results shows that the majority of false positives are due to onsets being detected in multiple neighbouring bands: the onset time is correct, but being attributed to the wrong frequency. When these 'spillover' detections are removed from the false positive count, the precision improves dramatically (see Table 3.6).

Table 3.6: Recall and precision (with and without spillover onsets), using 20 ms segments and requiring the mean log to increase by a significant amount.

Test audio	Recall	Precision (all detections)	Precision (w/out spillover)
Dream	100%	100%	100%
Alice	100%	57%	96%
Swan Lake 1	68%	24%	85%
Swan Lake 2	97%	51%	89%

Classifying events as true positives, false positives or spillover detections is not a task for an OnsetDetector, as it cannot access data from other bands. However, a NoteDetector will have data from every OnsetDetector and PitchTracker, so making such classification decisions will be the NoteDetector's responsibility.

3.5 Back to backtracking

The backtracking algorithm introduced in Section 3.1.1 is a useful starting point for developing a second stage, but it must be adapted to incorporate the new ideas for onset detection, such as critical bands and using averaging as stage one.

Line 26 of code extract 3.2 calls the function `findExactTime()`. An implementation of this is provided in code extract 3.4. This expands the original backtracking algorithm, but keeps the same structure: iteratively calculating a 'current' value and mean log of N previous values until this 'previous' mean log is no longer less than 'current'. The mean log method is also used here to reduce a band of k detectors to a single value at every sample of interest (see lines 20–23 of code extract 3.4, for example).

`findExactTime()` begins by setting the sample number at which it will start backtracking and the minimum sample it will backtrack to. If this minimum (`stop`) is reached, the function rejects the detection and returns

`False`. It then calculates initial values for the mean log of the current sample (`current`) and the previous `N` samples (`mean`) where `N` has been set to the number of samples in 75 ms. These values are repeatedly calculated and compared, moving backwards through the data. When it reaches the point at which the mean log of the previous 75 ms is no longer less than the mean log at the current value, it has found the point at which the responses began to increase, so the function verifies the detection by returning `True` and the current sample number.

```
def findExactTime(incStart, incStop):
    # incStart and incStop are the samples at which the run
    # of increasing segments began and ended, respectively

    # backtrack as far as 100ms before incStart
    # sr is sample rate of input
    stop_time = sr * 0.1

    # if 'incStart' is within the first 100ms, we can't
    # go further back than sample 0
    if stop_time > incStart:
        stop = 0
    else:
        stop = incStart - stop_time

    # idx will be current sample number as we backtrack
    idx = incStop

    # get mean log of critical band at current sample
    current = 0
    for k in range(chans):
        current += log(getResultItem(k,idx))
    current /= chans

    # number of samples in 75ms
    N = sr * 0.075

    # calculate mean log over previous 75ms
    mean = 0
    for i in range(idx-N, idx):
        for k in range(chans):
            mean += log(getResultItem(k,i))
    mean /= (chans*N)

    # backtrack, finding mean of prev N samples each time
    while idx > stop+N:
```

```
37
38          # if mean of prev N samples is less than current
39          if mean < current:
40
41              # decrement current index
42              idx -= 1
43
44              # get new 'current' value
45              current = 0
46              for k in range(chans):
47                  current += log(getResultItem(k,idx))
48              current /= chans
49
50              # remove most recent value from mean
51              mean -= (current / N)
52
53              # add new (older) value to mean
54              older = 0
55              for k in range(chans):
56                  older += log(getResultItem(k,idx-N))
57
58              mean += (older / (chans*N))
59
60          # if mean > current, have found the onset
61          else:
62              return True, idx
63
64      # if while loop exits, have not found an onset
65      return False, 0
```

Code Extract 3.4: Backtracking with critical band

The results of testing this algorithm are given in Table 3.7. With the exception of *Swan Lake* excerpt 1, the recall and precision for the test audio extracts is 100% (when spillover detections are not regarded as false positives). However, a significant number of the detections are more than 15 ms from the correct time. With the exception of one detection in *Swan Lake* excerpt 2, all of these are more than 15 ms *after* the correct time, rather than before. This imbalance suggests that the problem may be due to the algorithm, rather than human error in identifying the correct onset times.

This tendency for the returned value to be too late may be because when `current` is greater than `mean`, the value returned is at the end of the N sample block, but in some cases the actual onset may be within this

block. For example, in Figure 3.16, the red line marks the onset, detected at 13.241 s, and the grey region marks the 75 ms segment with which the value is compared. However, the mean log reaches a minimum shortly before the red line. This is where the onset should have been detected.

To compensate for this, when the algorithm finds a potential onset (i.e. the red line in Figure 3.16), it then looks for a minimum in the 10 ms preceding it. Figure 3.17a shows how this improves the onset time in Figure 3.16.

However, it is also important to note that, at some onsets, the preceding mean log values are quite stable: backtracking alone will find the correct time and moving the onset further back will be deleterious.

Therefore, when the local minimum is found, the mean log at this time is compared with the mean log at the time found by backtracking. If these are very similar (deviating by no more than 5% of each other), the local minimum time is ignored and the time found by simply backtracking is returned. Otherwise, the time at the local minimum is returned.

Figure 3.17 shows both scenarios and the (correct) onset time returned in each; Table 3.8 shows that this improves the results for all four piano test pieces, particularly *Dream and Little Dream of Me* and *Swan Lake* excerpt 2.

Backtracking start point

When tested with a more extensive data set (see section 4.1 in the next chapter) than the one used to test the prototypes discussed in this chapter, the OnsetDetector exhibited a tendency to return results tens, or even a hundreds, of milliseconds late. Analysing the state of the OnsetDetector at these points show that the backtracking algorithm was exiting far earlier than the errors that the local minimum was introduced to compensate for, so a different solution was required.

This was found by changing the point from which backtracking begins (i.e. the value given to `findExactTime()` as `incStop` in Code Extract 3.4).

Table 3.7: Results when backtracking is employed: recall and precision (with and without spillover onsets) and the proportion of detections within 15 ms of the correct time.

Test audio	**Recall**	**Precision (all detections)**	**Precision (w/out spill.)**	**Time diff.** $< \pm 15$ ms
Dream	100%	100%	100%	76%
Alice	100%	58%	100%	76%
Swan Lake 1	65%	23%	88%	80%
Swan Lake 2	100%	52%	100%	71%

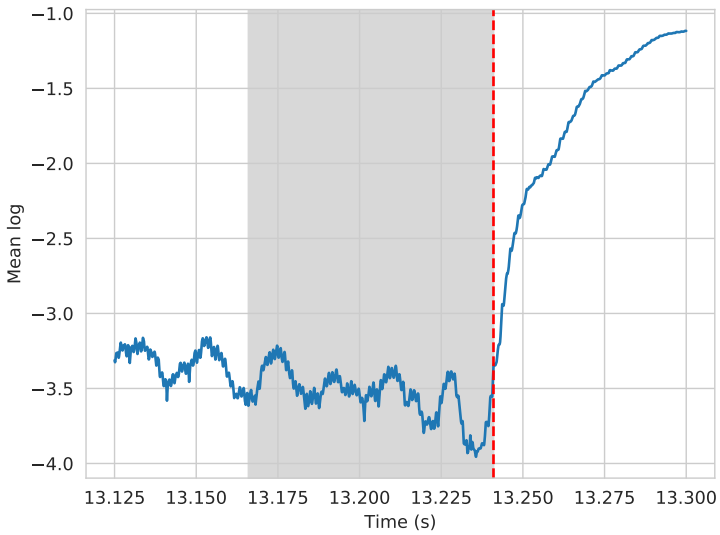
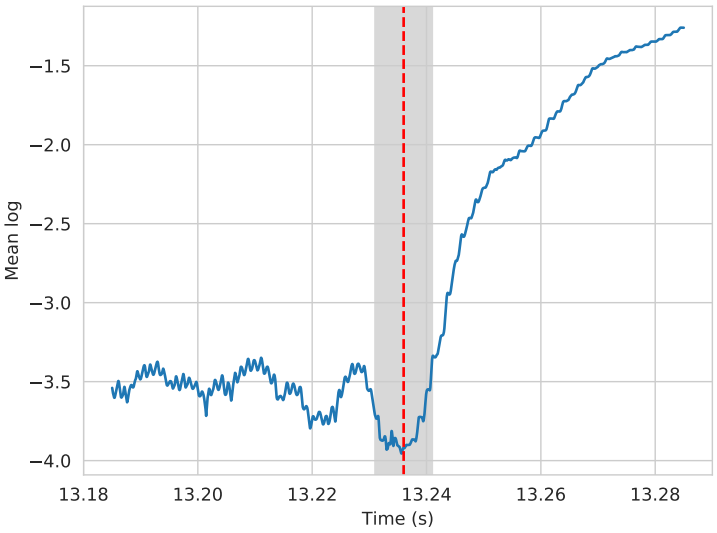


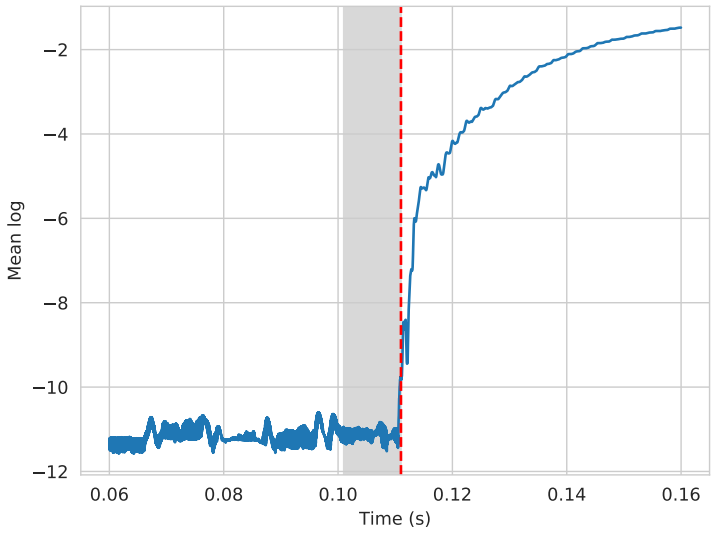
Figure 3.16: Mean log (blue) with detected onset shown in red. The grey region is the 75 ms over which the mean log is compared with the current value. From visual inspection, it seems that the onset should be detected at the local minimum point, a few milliseconds earlier.

Table 3.8: Results when backtracking with local minimum is employed: recall and precision (with and without spillover onsets) and the proportion of detections within 15 ms of the correct time.

Test audio	**Recall**	**Precision (all detections)**	**Precision (w/out spill.)**	**Time diff.** $< \pm 15$ ms
Dream	100%	100%	100%	88%
Alice	100%	58%	100%	82%
Swan Lake 1	65%	23%	88%	86%
Swan Lake 2	100%	52%	100%	89%



(a)



(b)

Figure 3.17: Mean log around notes, with returned onsets marked (dashed red line) and the 10 ms over which the local minimum was found (grey region). In (a), the local minimum provides a more accurate onset time than that seen in Figure 3.16. However, in (b) the mean log at the time found by backtracking alone (right hand edge of grey region) and that at the time of the local minimum are very similar, so the onset time found by backtracking is returned, as the time at the local minimum would be less accurate.

Rather than backtracking from the end of the consecutively increasing run of segments, the OnsetDetector finds the point in this run with the largest segment-to-segment increase and starts backtracking from this sample instead.

This change resulted in a significant improvement in the proportion of onsets that are successfully found by the OnsetDetector. For example, Figure 3.18 shows the mean log response to a trumpet note. In Figure 3.18a, when backtracking from the end of the increasing run of segments, the returned time is 140 ms late; however, Figure 3.18b shows that this is remedied when the new backtracking point is used: the difference between the automatically- and manually-found onsets is now only 6.5 ms. Therefore, this onset is now successfully detected.

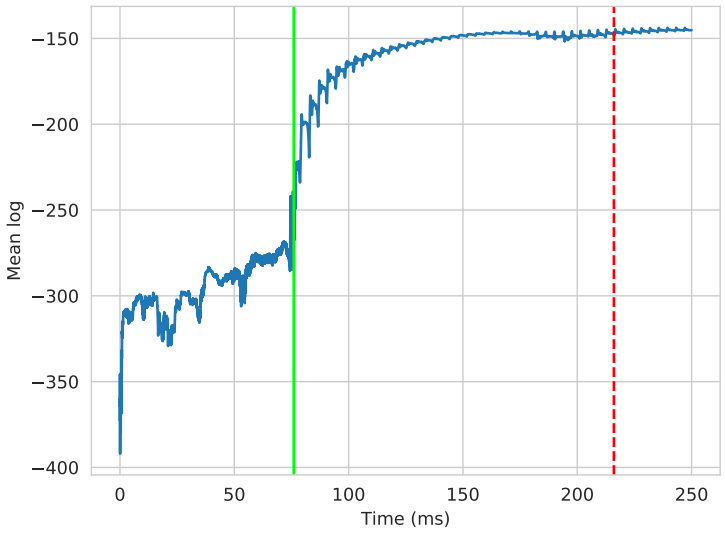
Zero padding

The backtracking algorithm described here requires access to DetectorBank output up to 100 ms before the segment averages began to increase. This may cause problems in situations where the onset is within the first 100 ms of the audio file: there may not be enough data available to backtrack effectively.

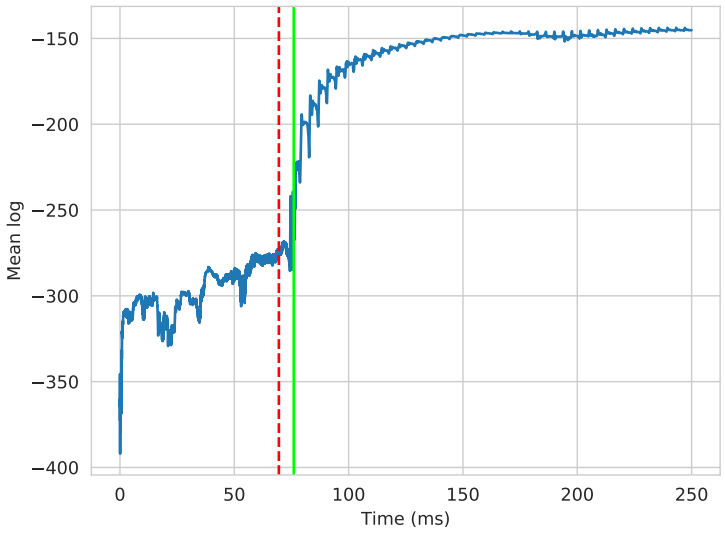
To counter this, a quarter of a second of silence is prepended to the audio buffer before onset detection begins. This offset is then subtracted from the onset times before returning. If backtracking returns a time within this offset period, the time returned will be zero.

An inconvenient consequence of zero padding arises when calculating the mean log of a segment: $\log(0)$ is undefined. In this case, the segment mean log is simply taken to be zero. This ensures that the point at which the response begins to react can still be found.

It also has an impact when deciding whether or not to return the local minimum time as the onset. In its initial implementation, if the local mini-



(a)



(b)

Figure 3.18: The mean log response to a trumpet note, with the manually found onset at 76 ms marked by the green line and the times returned by the OnsetDetector marked with a red dashed line. In (a) the time returned was 216 ms, which is clearly wrong. (b) shows the result when backtracking from the point with the largest segment-to-segment difference: the time returned is now 69.5 ms, just 6.5 ms from the hand-annotated time and therefore well within the range of times that would be perceived as simultaneous.

mum was less than 95% of the value at the time arrived at by backtracking, the time chosen is the local minimum time. However, the scenario may arise where the local minimum is sufficiently smaller because it falls within the zero-padded region of audio, i.e. the local minimum is zero. In this case, the local minimum time should not be used; the time found by backtracking is returned.

```
1  # 'current' is the mean log at the point backtracked to
2  # 'mn' will store the local minimum in the 10ms before this
3  mn = current
4
5  # the onset time is intially set to the current index 'idx'
6  onset = idx
7
8  # number of samples in 10ms
9  M = sr * 0.01
10
11 # iterate through the 10ms before the current point
12 for i in range(idx, idx-M, -1):
13     # calculate the mean log at this sample
14     meanlog = 0
15     for k in range(chans):
16         meanlog += log(getResultItem(k,i))
17     meanlog /= chans
18
19     # if less than previous min, store the value and index
20     if meanlog < mn:
21         mn = meanlog
22         onset = i
23
24 # if the local min time is not is sufficiently
25 # different or the minimum is zero (current/mn is NaN)
26 # use original time, idx, rather than local min time, i
27 if isnan(current/mn) or current/mn >= 0.95:
28     onset = idx
```

Code Extract 3.5: Local minimum search

Code extract 3.5 gives a full implementation of the local minimum algorithm. The local minimum is calculated in lines 12–22, then compared with the `current` value in the `if` statement in lines 27–28. `onset` is the sample number that would then be returned as the precise onset value.

3.6 Final design

Figure 3.19 shows the structure of the software, comprising the following objects:

NoteDetector Top-level object used to find notes.

EventDetector Created by NoteDetector to manage one band's OnsetDetector and PitchTracker.

OnsetDetector Analyses data provided by a band DetectorBank in order to find note onsets.

PitchTracker Calculates the input frequency for notes in the PitchTracker's critical band.

DetectorBank Bank of detectors.

DetectorCache Object which provides up to N segments of DetectorBank output.

The NoteDetector is the high-level object with which the user interacts. It is constructed with an input buffer and associated sample rate; a list of frequencies of interest; the target bandwidth of the detectors; the number of divisions per octave in the input, from which the critical band limits will be calculated; and any non-default parameters to be used when constructing each band's DetectorBank.

As stated in the discussion of Detector spacing, critical bands — as implemented here, rather than as defined in audiology — enable us to search for notes in a range of frequencies, rather than just notes at the given frequency. In practice, a DetectorBank is created for each frequency of interest, with n detectors tuned to cover the range determined by the band.

This DetectorBank provides the data to be analysed by an OnsetDetector and a PitchTracker. Each band has its own DetectorBank, OnsetDetector

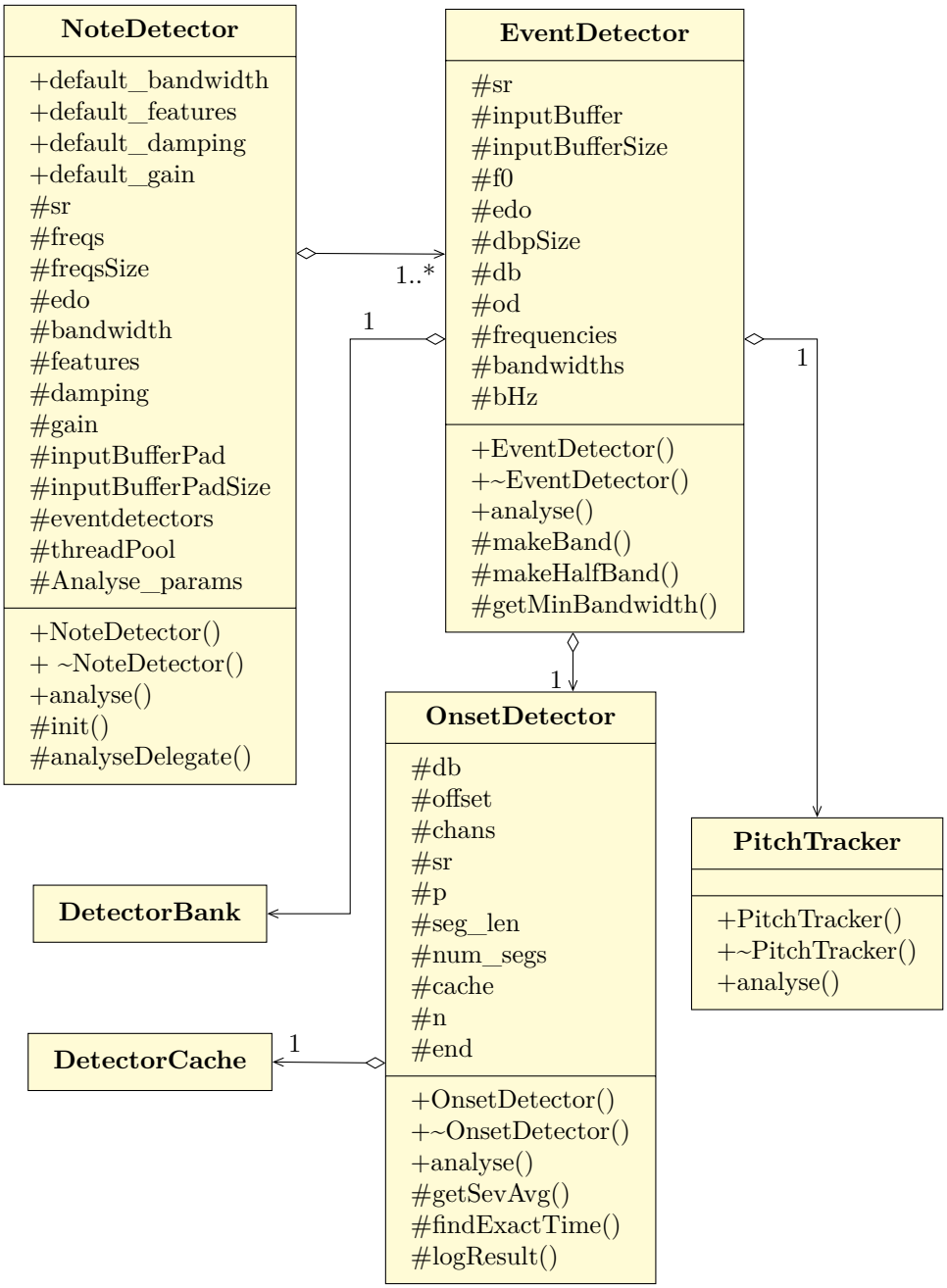


Figure 3.19: UML diagram showing the public (+) and protected (#) members and methods of the NoteDetector, EventDetector and OnsetDetector. The full DetectorBank UML was given in Figure 2.1; the PitchTracker is not implemented in this project, so only a skeleton design is shown here.

and PitchTracker and is independent of the other bands. Therefore, this structure is perfectly suited to multithreading, which speeds up execution considerably.

To manage this exchange of data between objects, another object is created. Given a centre frequency and critical band size, the *EventDetector* constructs a DetectorBank, OnsetDetector and PitchTracker. The EventDetector's `analyse` method runs the OnsetDetector and PitchTracker's `analyse` methods and returns the results to the NoteDetector.

The nomenclature here is important. The EventDetector does not return notes; it returns *events*, i.e. it reports that there is activity in the band. These do not necessarily have a one-to-one correlation with notes. For example, a single note with vibrato may appear as several events.

In order to classify the events as either note onsets or changes within a note, a broader context must be considered. For example, wide vibrato may only become clear when analysing events found in neighbouring bands. Every EventDetector is independent of the others, so has insufficient data to perform event classification. This task is the NoteDetector's responsibility, as it has access to the results from all the bands. However, an implementation of this is beyond the scope of this project. Currently, the NoteDetector simply returns the times collected by the EventDetector. The PitchTracker is also not currently implemented, although ideas for how it may work will be presented in the Further work section of Chapter 5.

Chapter 4

Testing the OnsetDetector

4.1 Data set

The University of Iowa Electronic Music Studios have a publicly available repository of musical instrument samples (Fritts 1997). A variety of brass, woodwind, string and tuned percussion instruments are sampled, as well as a piano and a guitar. Every note in each instrument's range is sampled. Figure 4.1 shows how many samples fall into each instrument category.

There is no available data concerning whether the performers are professional or amateur musicians.

The playing techniques encompassed a broad range of what is possible with the instruments: string instruments were played both arco and pizzicato; several brass and woodwind instruments were recorded with and without vibrato; the percussion instruments were played with a variety of different techniques and materials — for example, bowing or using mallets with heads made of rubber, yarn or rosewood — and features like damping and sustain were varied. Rolls on marimba and xylophone notes and xylophone glissandi were also recorded.

The wide range of instruments and playing techniques makes this a suitable data set for testing the OnsetDetector.

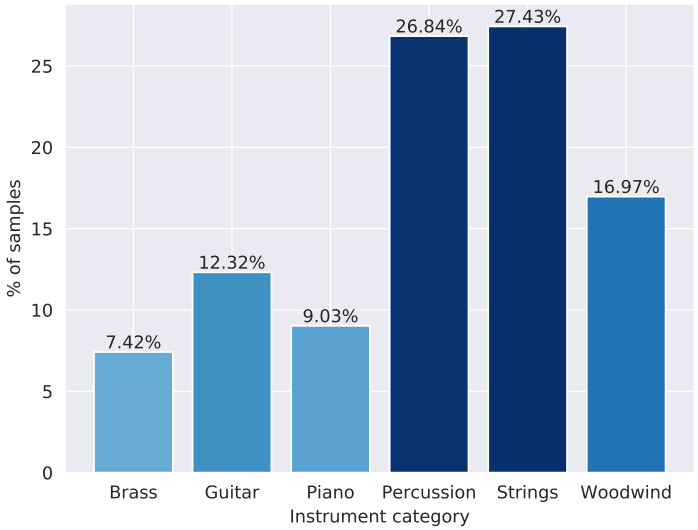


Figure 4.1: The proportion of samples in the data set that fall into each instrument category.

The 'post-2012' brass, percussion, strings and woodwind samples from the repository were used, along with the 'pre-2012' guitar and piano samples.[1] With the exception of the guitar samples, all audio files comprise one note per file. The guitar files were split into individual notes for these tests.

The files were recorded at a sample rate of 44.1 kHz and were released as AIFF (Audio Interchange File Format). Although 44.1 kHz is an acceptable sample rate for this software, the files were resampled to 48 kHz for consistency with previous tests, for example, the investigations in Chapter 2. They were also converted to WAV files.

The guitar and piano samples have versions of each sample at three different dynamic levels: ***pp***, ***mf*** and ***ff***.[2] The other instruments are labelled as having been recorded only at ***ff***, although in practice, there is a great deal of variation in loudness across these samples.

This data set comprises 2860 audio files. The onset time of each note was

[1] All 'post-2012' files and 'pre-2012' guitar files were accessed on 14 June 2018. The 'pre-2012' piano files were downloaded on 22 November 2016.

[2] ***pp***, ***mf*** and ***ff*** mean very soft, moderately loud and very strong, respectively.

Table 4.1: Proportion of onsets in each instrument category. Percussion dominates here because the each onset in the rolls is marked up.

Category	No. of onsets	% of onsets
Brass	212	2.55
Guitar	352	4.23
Piano	260	3.12
Percussion	6237	74.87
Strings	784	9.41
Woodwind	485	5.82

manually marked up by the author, an exercise that took more than three days. 148 of the audio files contain rolls played on marimba and xylophone notes, with an average of 37 beats per roll. The onset of each strike in the rolls were also marked-up. This brought the total number of onsets in the data set up to 8330. The total duration of this audio is over five hours.

Figure 4.1 showed that approximately a quarter of the samples are percussion recordings; however, due to the large number of onsets in the rolls, the majority of the onsets are in the category of percussion, as detailed in Table 4.1.

The names of the audio files in the data set indicate at least the instrument name, dynamics used and note sounding. Playing techniques, the string on which the note is played and whether the recording is in stereo are included, where relevant. The form of the naming convention can be stated as "instrument.[technique.]dynamic.[string.]note.[stereo]", where square brackets indicate optional parameters. In some cases, more than one playing technique may be given, for example when the beater type is specified and the file contains a roll: "Xylophone.rosewood.roll.ff.A4.stereo". The other options are illustrated by file names like "Violin.arco.ff.sulA.E6.stereo", "Horn.ff.C4.stereo", "Piano.pp.B4" and "SopSax.vib.ff.A4.stereo". The only exceptions to this format are the xylophone glissandi, which are named "Xylophone.gliss.down.stereo" and "Xylophone.gliss.up.stereo".

This means that for all files except the glissandi, the frequency to be

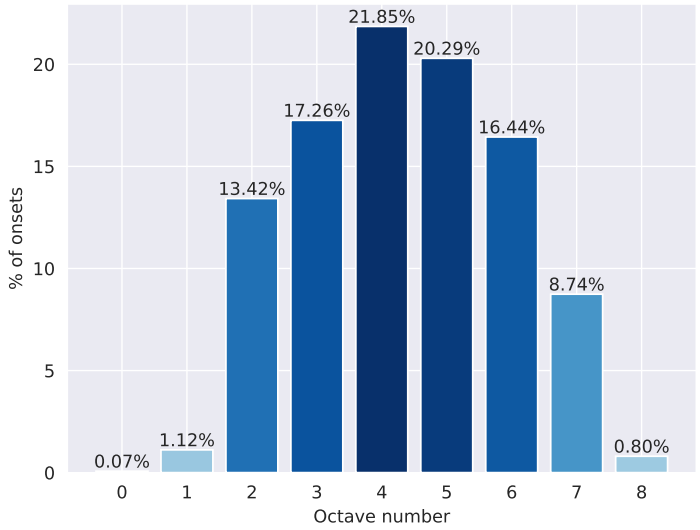


Figure 4.2: Proportion of onsets in each octave in the data set

requested from the OnsetDetector can be derived by parsing the file name to find the note being played. For the ascending and descending glissandi, the frequencies requested corresponded to the lowest and highest notes on the xylophone, respectively. In this case, the lowest note was F4 (349.228 Hz) and the highest was C8 (4186 Hz).

The samples span a wide range of pitches from A0 (27.5 Hz) to C8 (4186 Hz). The distribution of number of onsets per octave is shown in Figure 4.2. As would be expected, most of the onsets are found in the middle octaves, where most instrument's ranges are concentrated and therefore where they overlap. Also, between them, the xylophone and marimba rolls account for two thirds of the onsets in the data set. These instruments have a range of F4–C8 and C2–C7, respectively. In total, 75.8% of the onsets fall within octaves 3–6 (corresponding to fundamental frequencies from 130.813 Hz to 1975.53 Hz); only 2% of the onsets lie in the two lowest and single highest octaves.

4.2 Test setup

The files were analysed using six CPUs simultaneously. This was made possible by using the Python module SCOOP (Scalable COncurrent Operation in Python), which enables parallel computing on a number of hosts (Hold-Geoffroy et al. 2014).

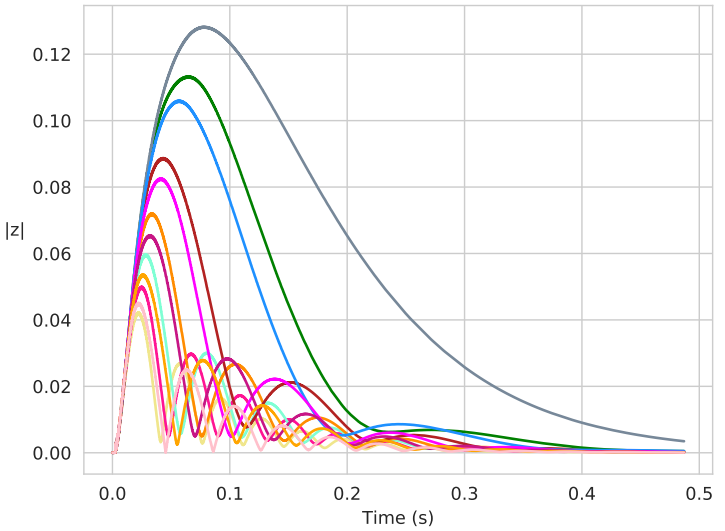
The samples were tested at two damping factors: $1 \cdot 10^{-4}$ and $5 \cdot 10^{-4}$. The former requires a smaller frequency step between detectors in the critical band, therefore more detectors must be used and the time taken to analyse the audio files increases accordingly.

With this setup, it took approximately 90 minutes to find all onsets when the damping was $5 \cdot 10^{-4}$ and seven hours when the damping was $1 \cdot 10^{-4}$. Both of these times are significantly less than the time taken to find the onsets by hand. The total duration of the audio in the data set is five hours: this analysis ran more than three times faster than real time with the higher damping factor, and at 70% of real time at the lower damping factor.

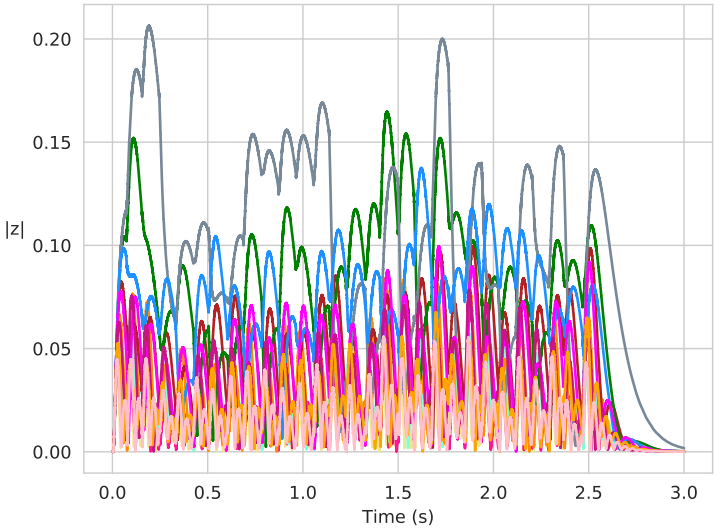
The OnsetDetector uses the DetectorCache to retrieve DetectorBank samples, which means this analysis does not demand a significant amount of memory.

During the OnsetDetector development, detailed in Chapter 3, a feature was added with the aim of finding Fewer false positives by suppressing intra-note detections. This was referred to as the 'last-first' criterion. As discussed when this feature was introduced, these are not necessarily errors; they are detections in response to changes in the frequency components present in any given band. However, the results returned by the OnsetDetector will be compared with the manually determined onset times, which do not include data about intra-note changes.

The only potential exception to this is the xylophone and marimba rolls, for which every note in the roll was marked. Although these are not technically intra-note events — as each onset corresponds to the note being struck



(a) Responses to single note



(b) Responses to roll

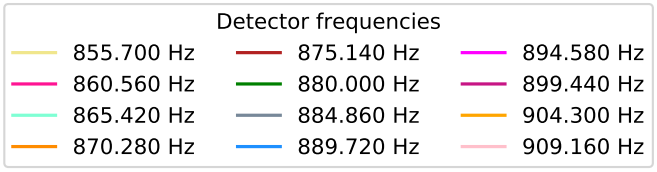


Figure 4.3: Band of responses to a xylophone A5, played both (a) as an individual note, and (b) as a roll. The damping factor used here is $5 \cdot 10^{-4}$, so the band consists of detectors at intervals of 4.86 Hz around 880 Hz.

with the mallet, rather than changes within the note due to features like vibrato — the response characteristics are somewhat similar to what we would expect from a note containing much variation, as new notes begin while the responses are still in relaxation from the previous event. Figure 4.3 shows a band of responses to two xylophone samples. The first is a single note and the second is a roll. The responses to the roll are clearly continually reacting to new events. For this reason, when the 'last-first' criterion is disabled, we would expect to see a high number of false positives — and hence, a low value for precision — for all inputs but the rolls. Indeed, when the OnsetDetector analysed the roll shown in Figure 4.3b under these circumstances, all 29 onsets were successfully found.

4.3 Results

The results of these tests are presented in terms of the precision, recall and F-measure. As stated when evaluating onset detection software in Test results and discussion of Section 1.4.2, the precision is a measure of how many of the automatic detections are correct and the recall tells us what proportion of the hand-annotated onsets have been successfully detected. These can be combined to provide a single value — the F-measure — which describes how well the OnsetDetector performs overall.

For both damping factors tested, the onset detection results are given for the entire data set, then broken down first by instrument category — brass, guitar, piano, percussion, strings and woodwind — then further divided by individual instrument and factors like the octave each sample falls in or technique used to play an instrument. For instruments that were recorded with two contrasting playing styles, like arco and pizzicato strings or the brass and woodwind instruments recorded with and without vibrato, these results are presented side-by-side. Results for percussive instruments are given all together and split into those with rolls and those without.

Percussion rolls are the only group for which we have data points that may be identified by the OnsetDetector when the 'last-first' condition is not enabled. Results found without this feature cannot meaningfully be compared with the rest of the data set, so are not broken down as comprehensively.

Some of the results are also plotted as radar charts, which are a common method of visualising three-dimensional data. One criticism of radar charts is that they exaggerate some results because the area of the triangles generated by the three data points scales by r^2, rather than r. Despite this, they are useful here, as they provide a visual comparison of the extent of data points, either along the same axis or around different axes. They are included here to illustrate the effect of changing one parameter in the test setup, or the difference in results from the same instruments with contrasting playing styles. For each figure, readers are referred to the tables that provide the exact values plotted.

In this chapter, the results are simply presented. Chapter 5 provides a full evaluation and discussion of the information given here.

4.3.1 Low damping

This section presents test results when a damping factor of $1 \cdot 10^{-4}$ is used. The threshold was set to $3 \cdot 10^{-4}$.

Table 4.2 and Figure 4.4 present and compare the precision, recall and F-measure for all samples, with and without the 'last-first' criterion.

Table 4.3 gives the results for each instrument category when the intra-note detections are suppressed; Table 4.4 gives the results when they are not.

In Table 4.5, the percussion results are split into groups depending on whether they contain rolls or single notes, both with and without the 'last-first' feature. Tables 4.6 and 4.7 show the results for each percussion in-

strument on which rolls were recorded, with the 'last-first' criterion enabled and disabled, respectively, and Table 4.8 presents the results for percussion samples that contain one note, with intra-note detection suppression.

Tables 4.9 to 4.11 detail the results for the string instruments when played arco and pizzicato, both for each instrument and taken as a whole.

Table 4.12 presents the results for all brass instruments and Table 4.13 for all woodwind instruments. Table 4.14 presents the results from both these categories where information about vibrato is available.

The radar charts in Figure 4.5 show comparisons of arco and pizzicato strings, and of brass and woodwind samples with and without vibrato.

The guitar and piano samples results are presented, for each of the three dynamic levels at which they were recorded, in Tables 4.15 and 4.16 and radar graphs plotted in Figure 4.6.

Table 4.17 breaks down the results for all samples by octave. Figure 4.2 in Section 4.1 shows the proportion of onsets in the data set that fall in each octave and therefore provides additional context for these results.

Table 4.2: Results of OnsetDetector tests on all samples. Low damping.

'last-first'	**Precision %**	**Recall %**	**F-measure %**
Enabled	76.878	26.7956	39.7399
Disabled	8.80152	77.5402	15.8086

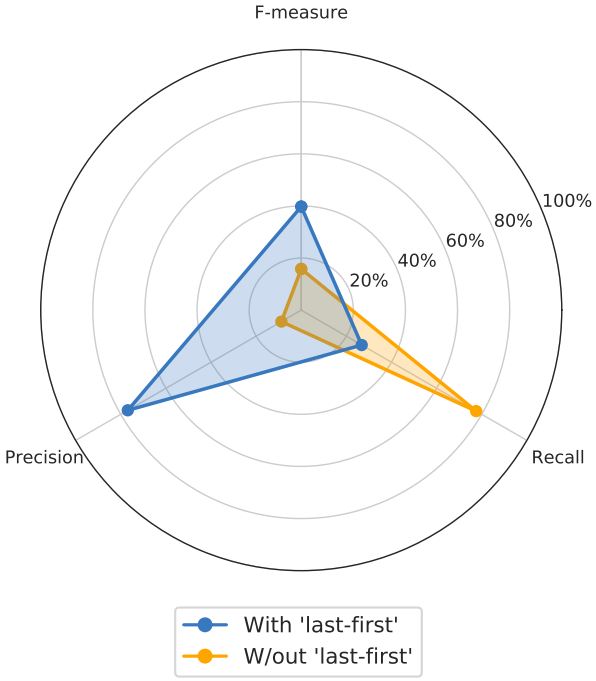


Figure 4.4: The precision, recall and F-measure for the whole data set with the with (blue) and without (orange) the 'last-first' criterion to reduce the number of detections within a note. Low damping.

Table 4.3: Results, by instrument category, with 'last-first' criterion enabled. Low damping.

Instrument Category	**Precision %**	**Recall %**	**F-measure %**
Brass	94.9495	88.6792	91.7073
Guitar	77.748	82.3864	80.0
Piano	100.0	58.8462	74.092
Percussion	94.1558	11.6316	20.7054
Strings	70.5418	79.7194	74.8503
Woodwind	47.8927	51.5464	49.6524

Table 4.4: Results, by instrument category, with 'last-first' criterion disabled. Low damping.

Instrument Category	**Precision %**	**Recall %**	**F-measure %**
Brass	4.41522	88.6792	8.41163
Guitar	3.7037	82.3864	7.08873
Piano	6.70465	58.8462	12.0378
Percussion	18.1268	79.3037	29.5087
Strings	3.18581	80.6122	6.12938
Woodwind	2.10544	51.5464	4.04563

Table 4.5: Precision, P, recall, R, and F-measure, F, for percussion samples, split by those that contain rolls and those that contain a single note, with and without the 'last-first' criterion. Low damping.

Style	**'last-first'**	**P %**	**R %**	**F %**
Rolls	Enabled	98.5149	3.54471	6.84319
Rolls	Disabled	61.9554	77.7699	68.9677
Single note	Enabled	92.6056	84.9758	88.6268
Single note	Disabled	2.85333	93.2149	5.53716

Table 4.6: Precision, recall, and F-measure for percussion samples containing rolls, with the 'last-first' criterion enabled. Low damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Marimba	97.561	3.36889	6.51289
Xylophone (hardrubber)	100.0	3.47044	6.70807
Xylophone (rosewood)	100.0	4.08163	7.84314

Table 4.7: Precision, recall, and F-measure for percussion samples containing rolls, with the 'last-first' criterion disabled. Low damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Marimba	57.601	77.653	66.1406
Xylophone (hardrubber)	66.0455	82.0051	73.1651
Xylophone (rosewood)	75.215	75.5102	75.3623

Table 4.8: Precision, recall, and F-measure for percussion samples containing one onset (this includes the xylophone glissandi), with the 'last-first' criterion enabled. Low damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bells (brass)	88.0952	90.2439	89.1566
Bells (plastic)	95.122	95.122	95.122
Crotale	84.6154	88.0	86.2745
Marimba (cord)	100.0	100.0	100.0
Marimba (deadstroke)	100.0	100.0	100.0
Marimba (rubber)	100.0	96.7213	98.3333
Marimba (yarn)	98.3333	98.3333	98.3333
Thai gong	92.3077	92.3077	92.3077
Vibraphone (bow)	39.1304	42.8571	40.9091
Vibraphone (dampen)	97.4359	90.4762	93.8272
Vibraphone (shortsustain)	100.0	100.0	100.0
Vibraphone (sustain)	100.0	92.6829	96.2025
Xylophone (gliss)	100.0	100.0	100.0
Xylophone (hardrubber)	100.0	61.3636	76.0563
Xylophone (rosewood)	100.0	27.2727	42.8571

Table 4.9: Precision, recall and F-measure for string samples, split according to playing technique: arco or pizzicato. Low damping.

Style	**Precision %**	**Recall %**	**F-measure %**
Arco	62.2	79.9486	69.9663
Pizzicato	81.3472	79.4937	80.4097

Table 4.10: Results for each string instrument, when played arco. Low damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bass	80.0	92.3077	85.7143
Cello	45.7364	62.1053	52.6786
Viola	49.6403	69.0	57.7406
Violin	77.6786	96.6667	86.1386

Table 4.11: Results for each string instrument, when played pizzicato. Low damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bass	83.4862	87.5	85.446
Cello	75.2475	76.0	75.6219
Viola	82.0	82.0	82.0
Violin	85.5263	71.4286	77.8443

Table 4.12: Results for each brass instrument recorded. The trumpet samples were recorded both with and without vibrato. Low damping.

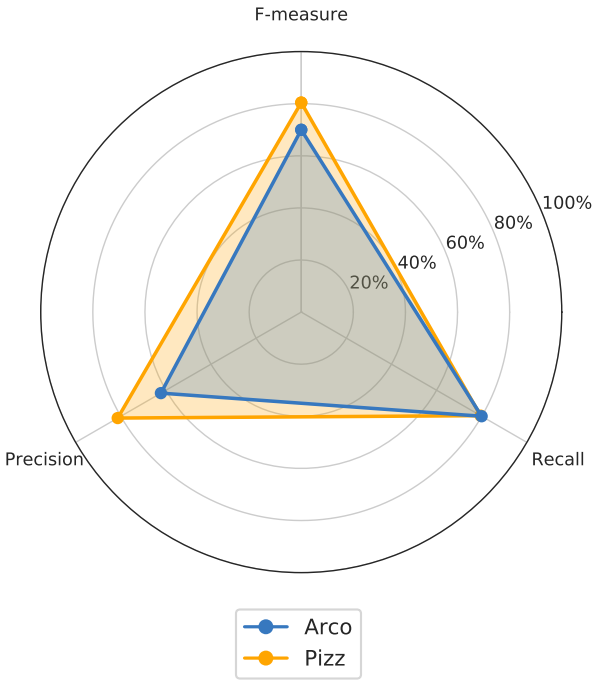
Instrument	Precision %	Recall %	F-measure %
Bass trombone	90.4762	70.3704	79.1667
Horn	89.3617	95.4545	92.3077
Tenor trombone	100.0	100.0	100.0
Trumpet (vibrato)	96.7742	85.7143	90.9091
Trumpet (no vibrato)	97.1429	94.4444	95.7746
Tuba	96.7742	81.0811	88.2353

Table 4.13: Results for each woodwind instrument recorded. Some samples were specifically labelled as being with or without vibrato. Low damping.

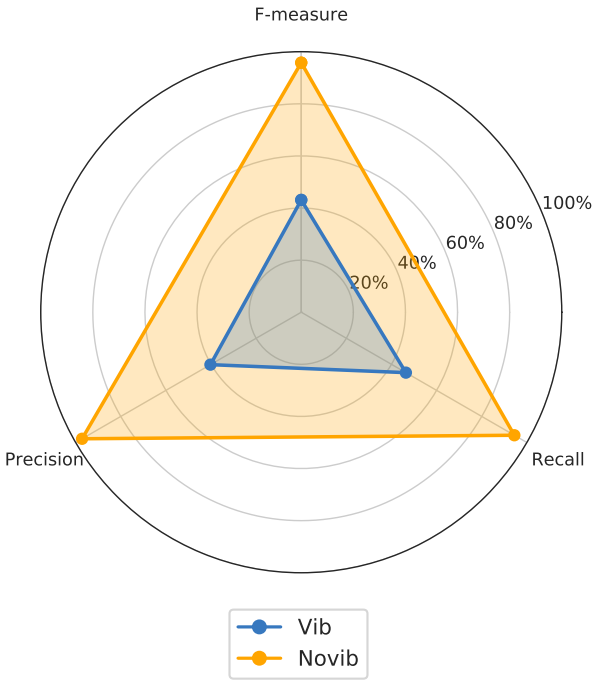
Instrument	Precision %	Recall %	F-measure %
Alto flute (vibrato)	57.7778	72.2222	64.1975
Alto sax. (vibrato)	26.4706	28.125	27.2727
Alto sax. (no vibrato)	31.25	31.25	31.25
Bass clarinet	53.1915	54.3478	53.7634
Bass flute	60.0	71.0526	65.0602
Bassoon	100.0	100.0	100.0
B♭ clarinet	10.6383	10.8696	10.7527
E♭ clarinet	12.8205	12.8205	12.8205
Flute (vibrato)	42.0	55.2632	47.7273
Flute (no vibrato)	69.0476	74.359	71.6049
Oboe	100.0	100.0	100.0
Soprano sax. (vibrato)	21.875	21.875	21.875
Soprano sax. (no vibrato)	32.3529	34.375	33.3333

Table 4.14: Precision, recall and F-measure for woodwind and brass instruments that were recorded with and without vibrato. Low damping.

Style	Precision %	Recall %	F-measure %
Vibrato	40.2516	46.3768	43.0976
No vibrato	97.1429	94.4444	95.7746



(a)



(b)

Figure 4.5: Figures comparing the results presented in (a) Table 4.9 and (b) Table 4.14

Table 4.15: Guitar results at various dynamic levels from very soft (***pp***) to very strong (***ff***). Low damping.

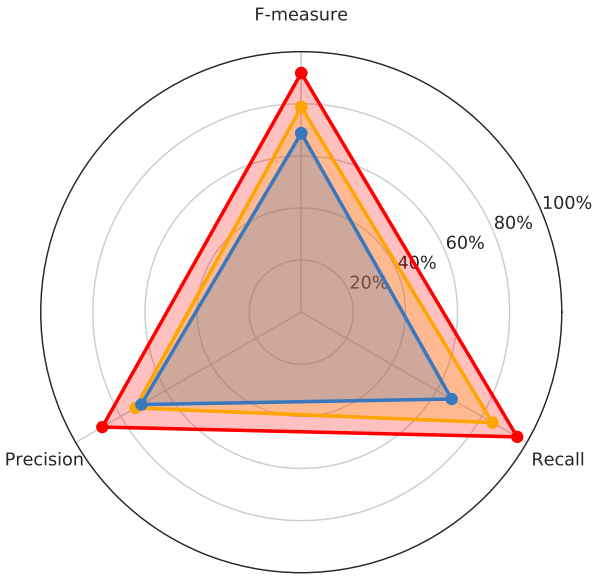
Dynamic	Precision %	Recall %	F-measure %
pp	70.9091	66.6667	68.7225
mf	73.5294	84.7458	78.7402
ff	88.189	95.7265	91.8033

Table 4.16: Piano results at various dynamic levels from very soft (***pp***) to very strong (***ff***). Low damping.

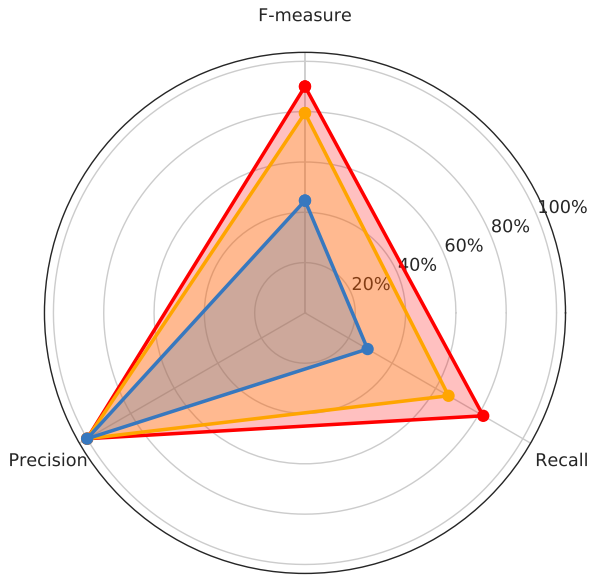
Dynamic	Precision %	Recall %	F-measure %
pp	100.0	28.7356	44.6429
mf	100.0	65.8824	79.4326
ff	100.0	81.8182	90.0

Table 4.17: Precision, recall and F-measure for all samples, split by octave number. Low damping.

Octave no.	Precision %	Recall %	F-measure %
0	0.0	0.0	0.0
1	82.0	44.086	57.3427
2	82.5967	26.7921	40.4601
3	76.4706	35.3064	48.3087
4	76.3871	32.5454	45.6438
5	73.5577	27.1277	39.6373
6	74.3119	17.7632	28.6726
7	89.5349	10.5769	18.9189
8	84.6154	16.6667	27.8481



(a) Guitar



(b) Piano

Figure 4.6: Figures comparing the results at different dynamic levels of (a) guitar samples (Table 4.15) and (b) piano samples (Table 4.16)

4.3.2 High damping

This section presents the results generated when the damping factor is set to $5 \cdot 10^{-4}$. In this case, the required threshold was also $5 \cdot 10^{-4}$. As with the Low damping section, the first three tables here, Tables 4.18 to 4.20, present results for all samples and categories, with and without the 'last-first' condition. The precision, recall and F-measure for all samples are again compared with the graph in Figure 4.7.

The results are then broken down in a similar fashion. First, percussion results are presented, split by whether the samples contain a single note or a roll, and by whether the 'last-first' criterion is enabled (Tables 4.21 to 4.24); then strings played arco or pizzicato (Tables 4.25 to 4.27); woodwind and brass samples with and without vibrato (Tables 4.28 to 4.30); guitar and piano results at each dynamic level (Tables 4.31 and 4.32) and finally by octave (Table 4.33).

Readers are once again referred back to Figure 4.2, which plots the proportion of onsets in each octave and so provides context when results are presented by octave.

The results for string playing technique, vibrato on brass and woodwind instruments and piano and guitar dynamic levels are plotted as radar charts in Figures 4.8 and 4.9.

Table 4.18: Results of OnsetDetector tests on all samples. High damping.

'last-first'	**Precision %**	**Recall %**	**F-measure %**
Enabled	70.5109	39.9472	51.0005
Disabled	8.31118	85.6113	15.1515

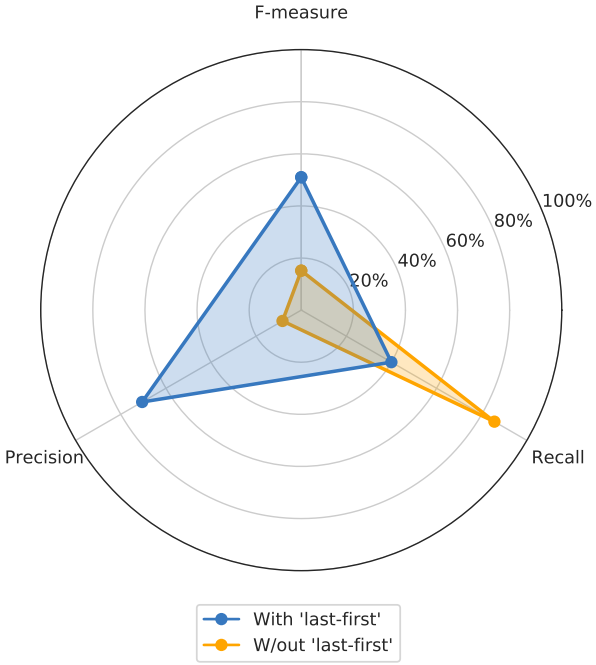


Figure 4.7: The precision, recall and F-measure for the whole data set with the with (blue) and without (orange) the 'last-first' criterion to reduce the number of detections within a note. High damping.

Table 4.19: Results of OnsetDetector test with 'last-first' criterion enabled. High damping.

Instrument Category	**Precision %**	**Recall %**	**F-measure %**
Brass	96.0396	91.5094	93.7198
Guitar	30.4258	83.2386	44.5627
Piano	82.4324	70.3846	75.9336
Percussion	95.1451	27.3544	42.4922
Strings	68.8017	84.949	76.0274
Woodwind	50.0	58.7629	54.0284

Table 4.20: Results of OnsetDetector test with 'last-first' criterion disabled. High damping.

Instrument Category	**Precision %**	**Recall %**	**F-measure %**
Brass	2.80064	91.5094	5.43493
Guitar	2.70872	82.9545	5.24614
Piano	4.82341	70.3846	9.02812
Percussion	16.6278	88.1277	27.977
Strings	3.79041	86.7347	7.26341
Woodwind	2.15232	58.9691	4.15305

Table 4.21: Precision, P, recall, R, and F-measure, F, for percussion samples, split by those that contain rolls and those that contain a single note, with and without the 'last-first' criterion. High damping.

Style	**'last-first'**	**P %**	**R %**	**F %**
Rolls	Enabled	99.1604	21.0367	34.7098
Rolls	Disabled	66.8256	87.3709	75.7295
Single note	Enabled	87.188	84.6527	85.9016
Single note	Disabled	2.28838	94.9919	4.4691

Table 4.22: Precision, recall, and F-measure for percussion samples containing rolls, with the 'last-first' criterion enabled. High damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Marimba	98.4802	18.192	30.7109
Xylophone (hardrubber)	100.0	24.6787	39.5876
Xylophone (rosewood)	100.0	26.7661	42.2291

Table 4.23: Precision, recall, and F-measure for percussion samples containing rolls, with the 'last-first' criterion disabled. High damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Marimba	62.0063	82.4256	70.7726
Xylophone (hardrubber)	70.2048	96.9152	81.4255
Xylophone (rosewood)	79.3599	95.3689	86.631

Table 4.24: Precision, recall, and F-measure for percussion samples containing one onset (this includes the xylophone glissandi), with the 'last-first' criterion enabled. High damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bells (brass)	88.6364	95.122	91.7647
Bells (plastic)	95.122	95.122	95.122
Crotale	40.678	96.0	57.1429
Marimba (cord)	100.0	100.0	100.0
Marimba (deadstroke)	100.0	100.0	100.0
Marimba (rubber)	100.0	96.7213	98.3333
Marimba (yarn)	96.6667	96.6667	96.6667
Thai gong	100.0	100.0	100.0
Vibraphone (bow)	36.7347	42.8571	39.5604
Vibraphone (dampen)	97.2222	83.3333	89.7436
Vibraphone (shortsustain)	100.0	100.0	100.0
Vibraphone (sustain)	100.0	90.2439	94.8718
Xylophone (gliss)	66.6667	100.0	80.0
Xylophone (hardrubber)	100.0	61.3636	76.0563
Xylophone (rosewood)	100.0	22.7273	37.037

Table 4.25: Precision, recall and F-measure for string samples, split according to playing technique: arco or pizzicato. High damping.

Style	**Precision %**	**Recall %**	**F-measure %**
Arco	59.1474	85.6041	69.958
Pizzicato	82.2222	84.3038	83.25

Table 4.26: Results for each string instrument, when played arco. High damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bass	71.5328	94.2308	81.3278
Cello	43.2624	64.2105	51.6949
Viola	56.9536	86.0	68.5259
Violin	65.6716	97.7778	78.5714

Table 4.27: Results for each string instrument, when played pizzicato. High damping.

Instrument	**Precision %**	**Recall %**	**F-measure %**
Bass	78.0488	92.3077	84.5815
Cello	74.5283	79.0	76.699
Viola	91.0891	92.0	91.5423
Violin	88.0	72.5275	79.5181

Table 4.28: Results for each brass instrument recorded. The trumpet samples were recorded both with and without vibrato. High damping.

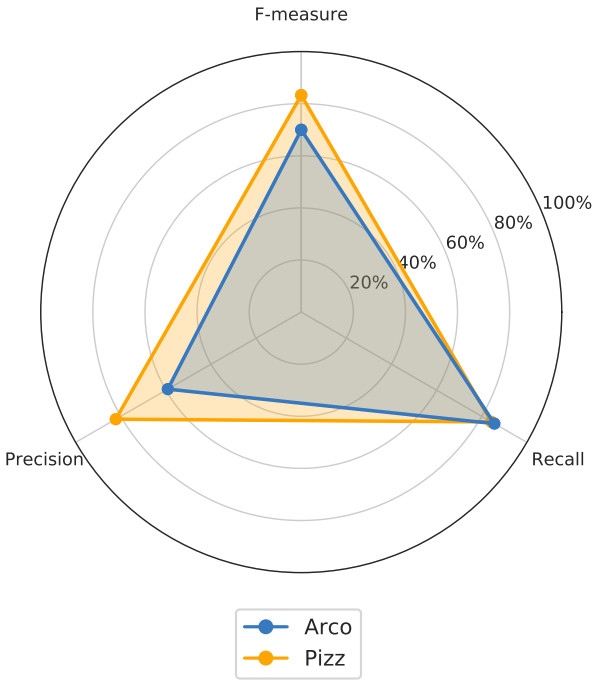
Instrument	Precision %	Recall %	F-measure %
Bass trombone	85.0	62.963	72.3404
Horn	95.4545	95.4545	95.4545
Tenor trombone	100.0	100.0	100.0
Trumpet (vibrato)	100.0	100.0	100.0
Trumpet (no vibrato)	94.5946	97.2222	95.8904
Tuba	96.9697	86.4865	91.4286

Table 4.29: Results for each brass instrument recorded. Some samples were specifically labelled as being with or without vibrato. High damping.

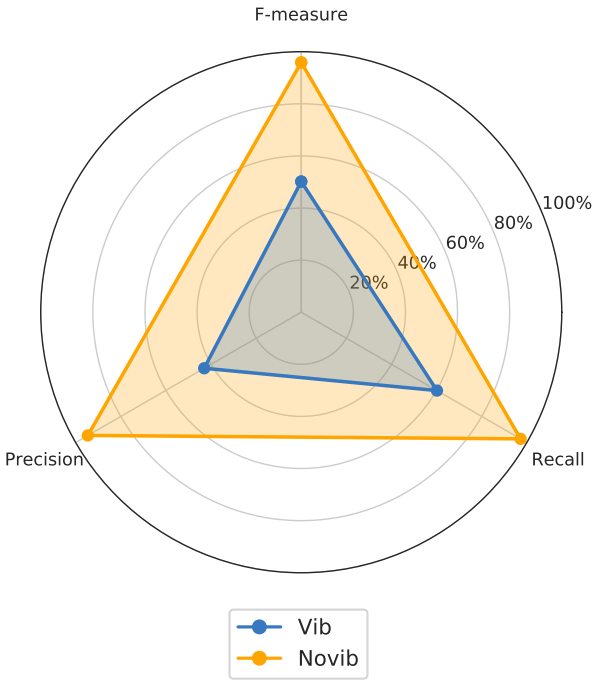
Instrument	Precision %	Recall %	F-measure %
Alto flute (vibrato)	47.3684	75.0	58.0645
Alto sax. (vibrato)	31.4286	34.375	32.8358
Alto sax. (no vibrato)	39.3939	40.625	40.0
Bass clarinet	55.3191	56.5217	55.914
Bass flute	47.3684	71.0526	56.8421
Bassoon	100.0	100.0	100.0
B♭ clarinet	19.1489	19.5652	19.3548
E♭ clarinet	22.5	23.0769	22.7848
Flute (vibrato)	37.6812	68.4211	48.5981
Flute (no vibrato)	66.6667	71.7949	69.1358
Oboe	100.0	100.0	100.0
Soprano sax. (vibrato)	50.0	53.125	51.5152
Soprano sax. (no vibrato)	50.0	53.125	51.5152

Table 4.30: Precision, recall and F-measure for woodwind and brass instruments that were recorded with and without vibrato. High damping.

Style	Precision %	Recall %	F-measure %
Vibrato	43.0052	60.1449	50.1511
No vibrato	94.5946	97.2222	95.8904



(a)



(b)

Figure 4.8: Figures comparing the results presented in (a) Table 4.25 and (b) Table 4.30

Table 4.31: Guitar results at various dynamic levels from very soft (***pp***) to very strong (***ff***). High damping.

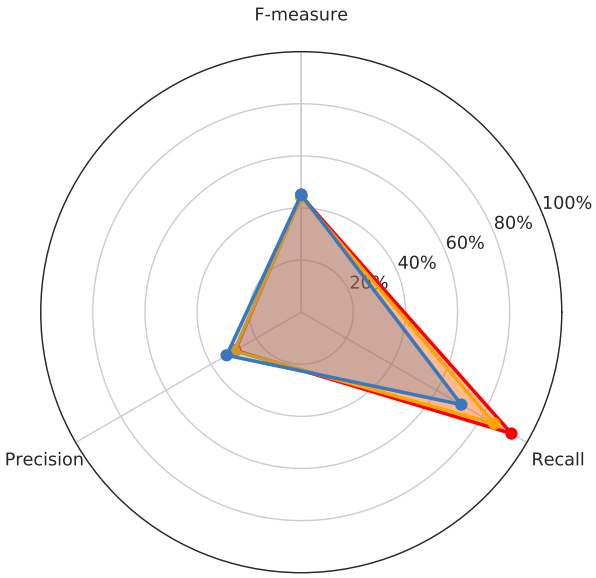
Dynamic	Precision %	Recall %	F-measure %
pp	33.0677	70.9402	45.1087
mf	29.7059	85.5932	44.1048
ff	29.3011	93.1624	44.5808

Table 4.32: Piano results at various dynamic levels from very soft (***pp***) to very strong (***ff***). High damping.

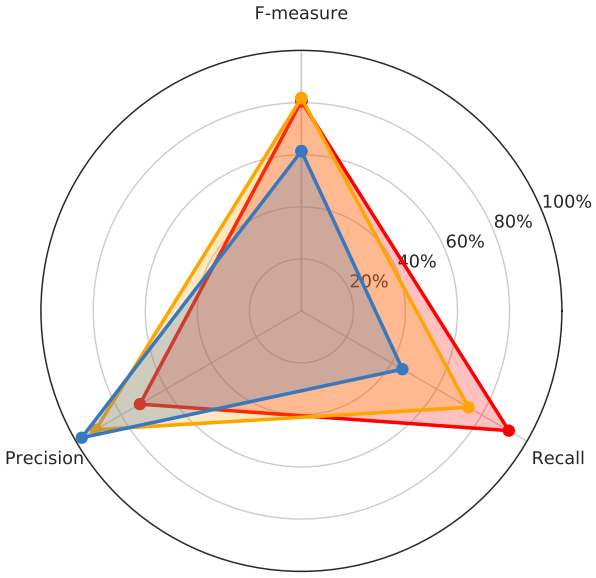
Dynamic	Precision %	Recall %	F-measure %
pp	97.5	44.8276	61.4173
mf	91.3043	74.1176	81.8182
ff	71.6814	92.0455	80.597

Table 4.33: Precision, recall and F-measure for all samples, split by octave number. High damping.

Octave no.	Precision %	Recall %	F-measure %
0	0.0	0.0	0.0
1	78.125	53.7634	63.6943
2	55.1873	34.319	42.3204
3	58.8508	44.2201	50.497
4	78.0836	42.1111	54.7143
5	70.354	37.5887	48.9985
6	84.4311	41.2281	55.4028
7	89.5973	36.6758	52.0468
8	88.4615	34.8485	50.0



(a) Guitar



(b) Piano

Figure 4.9: Figures comparing the results at different dynamic levels of (a) guitar samples (Table 4.31) and (b) piano samples (Table 4.32)

Chapter 5

Discussion

5.1 Comparison with MIREX 2018 results

Figure 5.1 compares the results of the OnsetDetector with the results from the ten algorithms tested for the 2018 MIREX onset detection challenge.[1] Although these tests use a different data set, comparing them can provide a benchmark for how well the OnsetDetector performs on different types of instrument.

Of the various sample categories designated by MIREX, five were determined to be sufficiently similar to those used to generate the results in Chapter 4: arco strings, pizzicato strings, brass, woodwind and percussion. MIREX has two percussion categories: solo bars and bells and solo drums. The former is used here and compared with the results for single note percussion samples.

All the results used in this comparison are those found when the 'last-first' criterion was enabled. As will be discussed in full in Section 5.2.1, this feature suppresses detections that occur when a detector is already responding, so the (true positive) results correspond to note onsets, not

[1] MIREX was introduced in Chapter 1, Section 1.4.2. The onset detection results from 2018 are available at https://nema.lis.illinois.edu/nema_out/mirex2018/results/aod/index.html.

intra-note events. This is also the reason for using not including rolls in the percussion samples, as these samples comprise repeated strikes of the same note in quick succession, so require the version of the OnsetDetector that does not suppress these events in order to produce meaningful results.

Table 5.1 lists the MIREX categories and their equivalents in the data set used here, along with the table numbers where the results used in this comparison can be found.

The OnsetDetector consistently performs at least as well as the various algorithms tested by MIREX, with results for arco strings and brass samples comparing particularly favourably. Table 5.2 gives the mean and standard deviation of the F-measures of the algorithms tested by MIREX, along with the F-measures returned by the OnsetDetector at both damping levels. The OnsetDetector results for arco strings and brass are higher than the mean of the corresponding MIREX algorithms by one and two standard deviations, respectively. The pizzicato strings and woodwind results are below the mean, within one standard deviation. Although the OnsetDetector performed well on the percussion samples, second only to the results from the brass samples, the F-measures are up to three standard deviations below the MIREX mean.

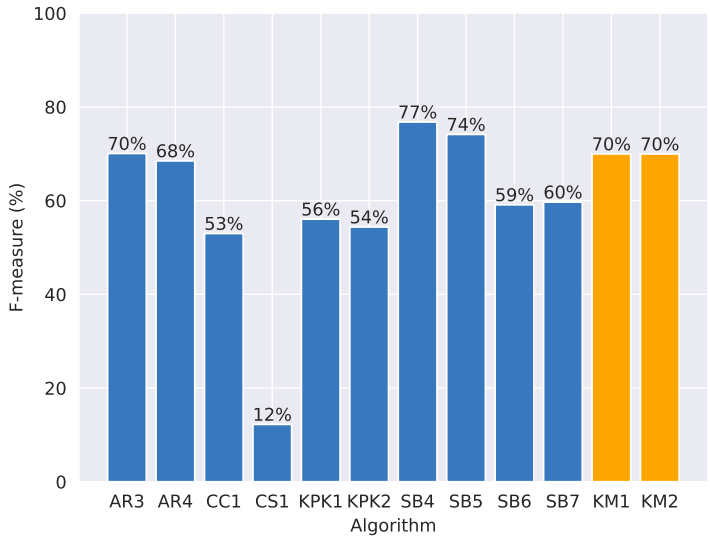
The rest of this chapter analyses the results to identify the strengths and weaknesses of the OnsetDetector algorithm or shortcomings in the data set (Section 5.2) and suggests various improvements that could be made to the OnsetDetector, as well as how to integrate it into a NoteDetector capable of detecting both time and frequency data from an audio signal (Section 5.3).

Table 5.1: MIREX sample classes and the corresponding sample categories and tables for which results are compared.

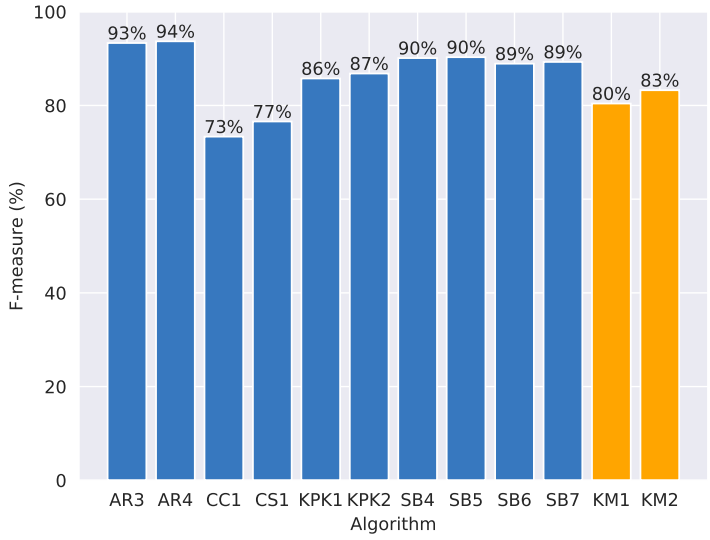
MIREX class	Category	Tables
Solo Bars And Bells	Percussion (single note)	4.5, 4.21
Solo Brass	Brass	4.3, 4.19
Solo Plucked Strings	Pizzicato strings	4.9, 4.25
Solo Sustained Strings	Arco strings	4.9, 4.25
Solo Winds	Woodwind	4.3, 4.19

Table 5.2: The MIREX mean and standard deviation (SD) of the F-measure for each sample category and the low and high damping F-measures from the corresponding OnsetDetector tests (KM1 and KM2, respectively).

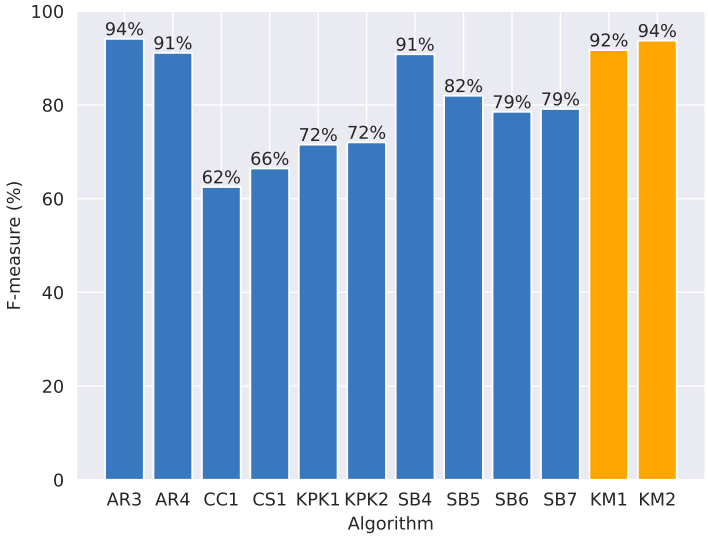
Category	Mean %	SD %	KM1 %	KM2 %
Percussion	94.2717	3.90792	88.6268	85.9016
Brass	78.8079	10.2869	91.7073	93.7198
Pizzicato strings	86.8107	6.40698	80.4097	83.25
Arco strings	58.3916	17.3406	69.9663	69.958
Woodwind	65.758	20.52	49.6524	54.0284



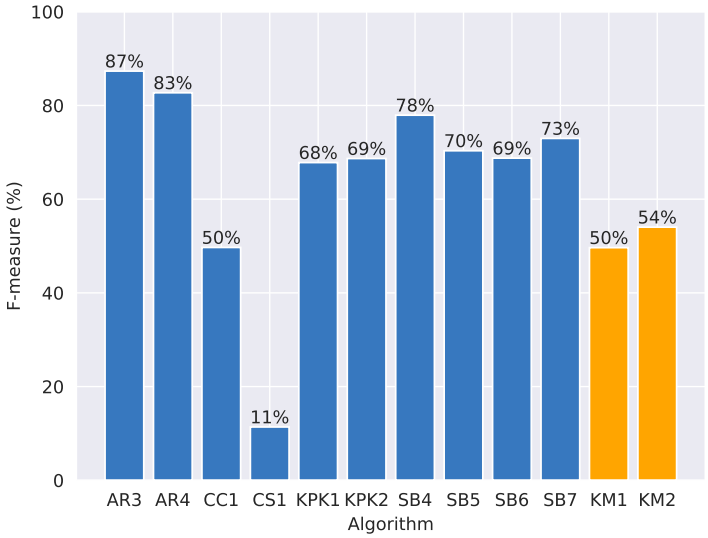
(a) Arco strings



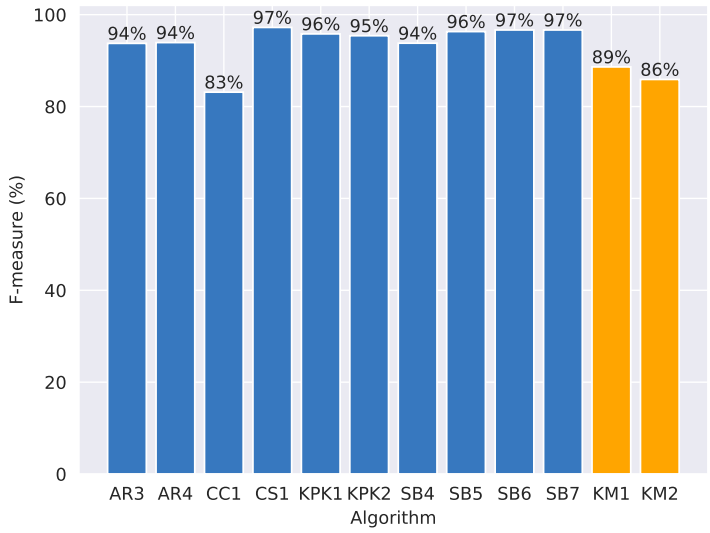
(b) Pizzicato strings



(c) Brass



(d) Woodwind



(e) Percussion

Figure 5.1: Comparing MIREX onset detection F-measures with those presented here. The first ten bars (blue) are the results of the algorithms tested by MIREX. The keys refer to the algorithm makers and are explained on their website. Bars 'KM1' and 'KM2' (orange) represent the low and high damping results returned by the OnsetDetector and presented in Chapter 4. The tables which provide the 'KM1' and 'KM2' results are listed in Table 5.1.

5.2 Analysis of results

5.2.1 Intra-note event detection

It can be seen from a number of tables in Chapter 4 — for example Tables 4.2 and 4.18, which present the results for the whole data set — that results calculated with the 'last-first' criterion enabled have higher precision values than those without, but there is often a corresponding drop in recall.

A low recall value occurs when many onsets are missed by the OnsetDetector; a high precision occurs when there are very few false positives. In both cases, this means there are fewer detections overall. Therefore, we can say that the 'last-first' criterion does indeed suppress detections for both damping factors tested.

When this setting is disabled, there are many more false positives returned whilst the note is still sounding. As has been stated many times throughout this thesis, detections within a note are not necessarily incorrect: we simply have limited data against which to check them.

One area where we do have relevant data is the percussion samples containing rolls. Comparing Table 4.3 with 4.4 and Table 4.19 with 4.20 shows that the recall values (i.e. the number of onsets successfully found) for most categories of instrument are unaffected, as there is only one note onset in each audio file and hence not much opportunity for false negatives. The percussion recall is much improved by disabling this setting, but this comes at the expense of precision. On this basis, we would expect the best results to be obtained for individual notes when the 'last-first' criterion enabled and when it is disabled for rolls.

This is borne out by the data in Tables 4.5 and 4.21, which break down the percussion results by those that comprise rolls and those that comprise single notes. The F-measures in the scenarios described above are 88.6% (single notes, enabled) and 69.0% (rolls, disabled) when the damping factor

is $1 \cdot 10^{-4}$, and 85.9% and 75.7% when the damping factor is raised to $5 \cdot 10^{-4}$. All of these values are higher than the F-measures found when the percussion samples are not split into these categories (see Tables 4.3, 4.4, 4.19 and 4.20).

This increase in recall and drop in precision and F-measure is also seen in Tables 4.2 and 4.18, and illustrated in Figures 4.4 and 4.7 which plot the precision, recall and F-measure for the whole data set in both scenarios. Although the recall of other instrument categories is unaffected by enabling or disabling the 'last-first' condition, the percussion rolls represent such a large proportion of the onsets (per Table 4.1) that the change in percussion results has a highly visible effect on the values for all samples.

Tables 4.6, 4.7, 4.22 and 4.23 present the precision, recall and F-measure for each group of samples that contain rolls — marimba, and xylophone played with rubber or rosewood mallets — at both damping factors. Once again, it can be seen that disabling this feature leads to better results for rolls, with F-measures ranging from 66.1% to 86.6%. When enabled, the precision drops from being in the high nineties or even 100% to being in the range of 57.6–79.4%. Whilst not ideal, the higher end of this range is still a reasonable value for precision and is comparable with some of the onset detection methods evaluated in the most recent MIREX tests.

Consideration of all this suggests that an implementation of the Onset-Detector without the 'last-first' condition may yield promising results when tested on a data set which contains information about intra-note events. This may raise recall values without the consequent drop in precision and therefore improve the F-measure across the board.

5.2.2 Instrument category

Analysis of results for the remaining instrument categories will proceed by considering only the results generated when the 'last-first' criterion was en-

abled (presented in Tables 4.3 and 4.19), for the reasons discussed above: namely, that disabling it has no effect on the recall and a highly deleterious effect on the precision and F-measure (as can be seen in Tables 4.4 and 4.20).

For most of the categories of instrument tested — the brass, string and woodwind instruments, as well as the piano — changing the damping factor did not have a significant effect on the results, as can be seen by comparing Tables 4.3 and 4.19. Only the percussion recall and guitar precision measurements vary. Differences which occur when the damping factor is changed will be discussed in Section 5.2.4; this section will focus on changes within an instrument category that can be seen at both damping factors tested.

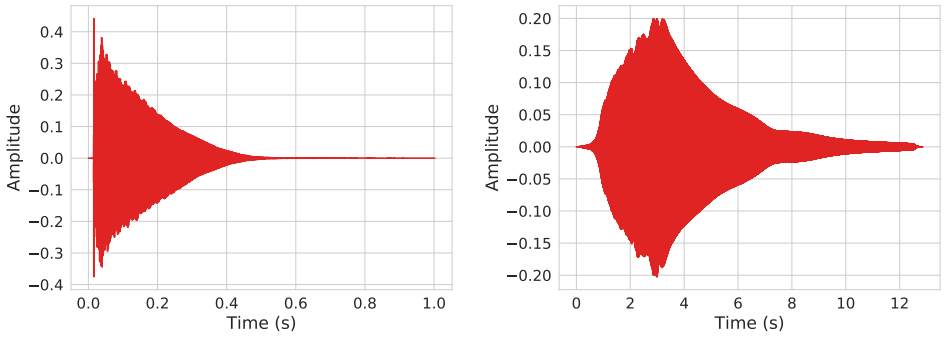
Percussion

The percussion samples returned generally good results, with the majority of F-measures greater than 85%, and a number of instruments with F-measures of 100%, as can be seen in Tables 4.8 and 4.24, which present the results for each percussion sample that contains a single onset. This includes the xylophone glissandi.

For the reasons discussed in Section 5.2.1, the OnsetDetector in the form used to analyse these samples does not perform well on rolls. The results for rolls can be found in Tables 4.6, 4.7, 4.22 and 4.23. Tables 4.5 and 4.21 split the percussion results by sample content. The "Single note, Enabled" rows provide total results for the samples that will be discussed in this section. The precision, recall and F-measure values in this case range from 84.6% to 92.6%.

One may expect onset detection in percussive instruments to be simpler task than for other types of instrument, as the onsets of notes sounded by striking an instrument are quite clearly defined: there is one instantaneous excitation event, which leads to a short attack.[2] All the percussion samples

[2] ADSR — attack, decay, sustain, release — envelopes were introduced in Section 1.2.1.



(a) (b)

Figure 5.2: An A4 played on a vibraphone, (a) by striking (then damping) the note and (b) by bowing. When simply striking the note, the attack is very short, but when bowed, it is drawn out over approximately two seconds. This is longer than the entire duration of the sample in (a).

are played in this manner, apart from one set of vibraphone samples, in which the notes were bowed. In these samples, the bow scrapes across the bar, resulting in an extended attack. A longer attack period may lead to uncertainty when determining the onset time, both for an algorithm and when marking up an audio file by hand, as there are more samples which could potentially represent the onset. Figure 5.2 shows an A4 from both struck and bowed vibraphone data sets. In Figure 5.2a, the note is simply struck with a mallet; in Figure 5.2b, the vibraphone is bowed. The difference in attack characteristics is clear: in Figure 5.2b, the amplitude of the note envelope increases very slowly. In fact, the attack lasts for longer than the entire note in Figure 5.2a.

Onset detection of the bowed vibraphone samples returned an F-measure of about 40%, far lower than the other vibraphone categories, which had F-measures ranging from 90–100%. The extended attack period may be the source of the poor results for the bowed vibraphone for one of two reasons: either the OnsetDetector could not reliably determine the onset times or there was a high degree of human error when marking up the audio files. Figure 5.3 shows both the audio waveform and the mean log of the first

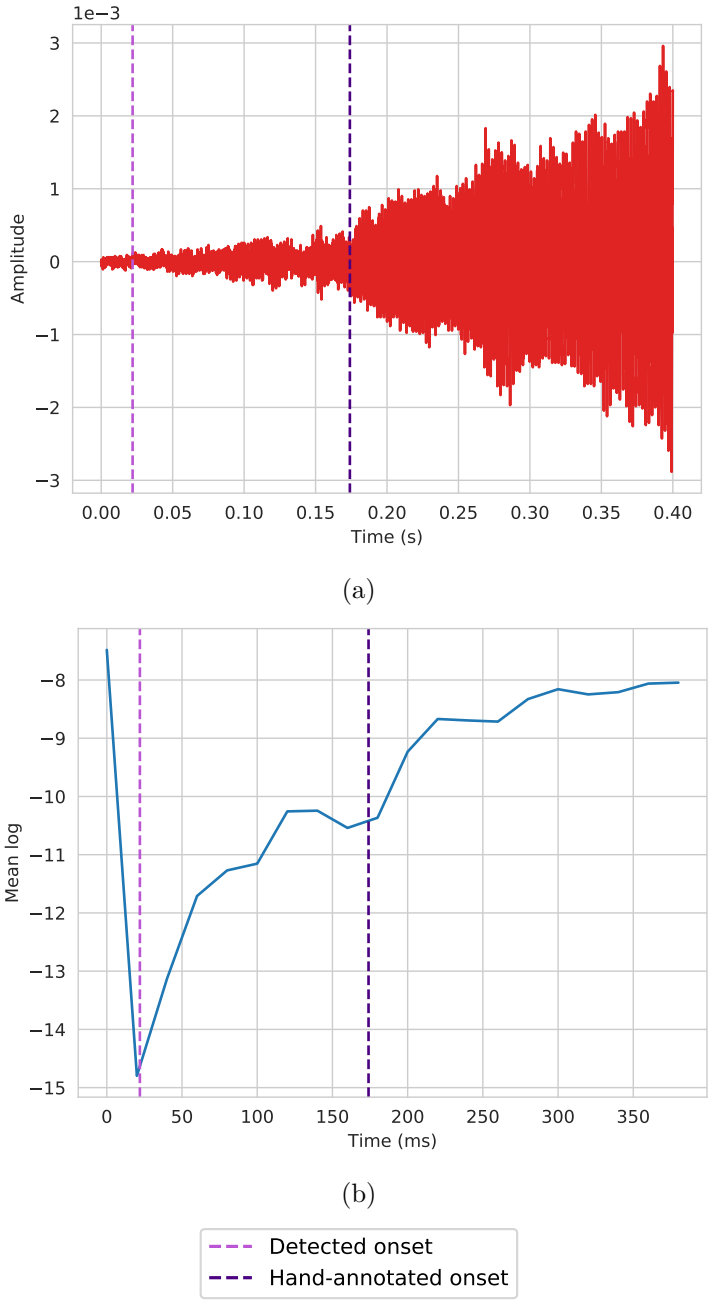


Figure 5.3: (a) audio and (b) mean log of each 20 ms segment of the first 400 ms of the bowed vibraphone A4 sample. The damping factor used is $5 \cdot 10^{-4}$. The light purple dashed line marked the time returned by the OnsetDetector, 152 ms before the time found when manually marking up the audio (marked with the dark purple line).

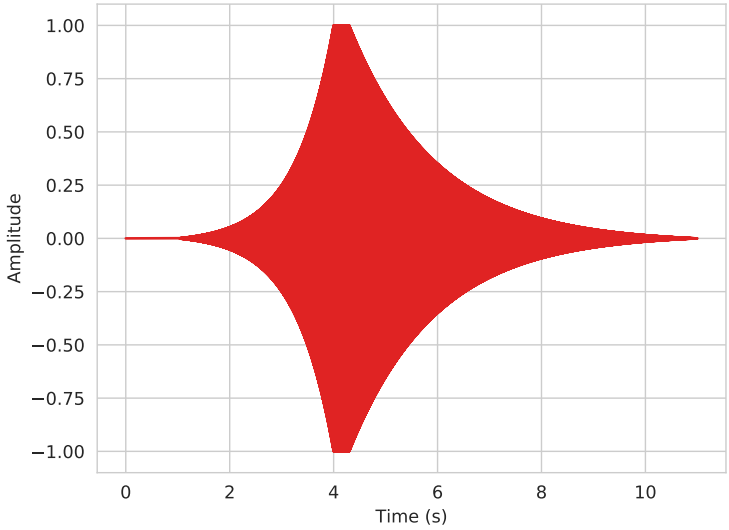


Figure 5.4: Tone generated at 440 Hz, $f_s = 48$ kHz, with an envelope shape mimicking Figure 5.2b. The tone here starts to sound after one second.

400 ms of the bowed vibraphone A4 sample. The time returned by the OnsetDetector is 22 ms, far earlier than the hand-annotated time of 174 ms. The mean log shown here was calculated with a damping factor of $5 \cdot 10^{-4}$ and is very similar to the that generated by a lower damping factor. The onset was manually found by listening to the audio, slowing it down and zooming in on the waveform in both time and amplitude. Comparison of Figure 5.3a with Figure 5.2b shows that the amplitude at the beginning of the waveform is less than 1.5% of its eventual maximum. Consequently, the exact location of the onset was difficult to ascertain, either aurally or visually.

In order to test the likelihood of human error in the markup process, sine tones were generated at all fundamental frequencies from 27.5 Hz (A0) to 4186 Hz (C8), then faded in and out to mimic the envelope seen in Figure 5.2b. One second of silence was inserted at the beginning. Therefore, the exact onset time of the tone is known.

The waveform generated at 440 Hz can be seen in Figure 5.4. Although

the shape of the attack does not exactly match that of the bowed vibraphone samples, in the first 400 ms of the tone, the amplitude reaches approximately 1.5% of its maximum value, like the bowed vibraphone sample shown above.

Figure 5.5 shows the waveform and mean log for this 400 ms window. The shapes of both plots are broadly similar to those seen in Figure 5.3.

At both damping factors ($1 \cdot 10^{-4}$ and $5 \cdot 10^{-4}$), for all fundamental frequencies from 27.5 Hz (A0) to 4186 Hz (C8), the OnsetDetector returned times which were 20–30 ms after the onset of the tone. These delays represent the time taken for the mean log to go from zero to its minimum and, as none was more than 30 ms, all detections are within the window of times that can be considered true positives. Therefore, these tests suggest that the low precision, recall and F-measure for bowed vibraphone samples are due to human error when marking up the audio files, rather than algorithmic error.

The xylophone samples have 100% precision in all categories, except the glissandi at high damping. There are only two glissandi, therefore only two onsets, so a single false positive has a large effect on the precision here.

The xylophone was recorded using two different beater materials, rosewood and rubber, both of which returned sub-optimal recall values (27.3 and 61.4% at the lower damping factor, respectively, and 22.7 and 61.4% at the higher), although the precision in all cases was 100%.[3] These recall values seem incongruous, especially when compared with the marimba, as these instruments are very similar: both comprise tuned wooden blocks which are struck with a mallet. However, the marimba results were significantly better, with recall values that never sank below 96.7%.

The main difference between the marimba and xylophone samples — and indeed between xylophone samples played with rosewood mallets, as opposed to rubber — is the onset times of the notes. The onsets of all

[3]Only single note samples are under consideration here, as the rolls were discussed in Section 5.2.1. Although, even for rolls on the xylophone, the precision was 100% across the board. As one would expect, recall is where they rate poorly, due to the large number of false negatives, as intra-note detections are being suppressed.

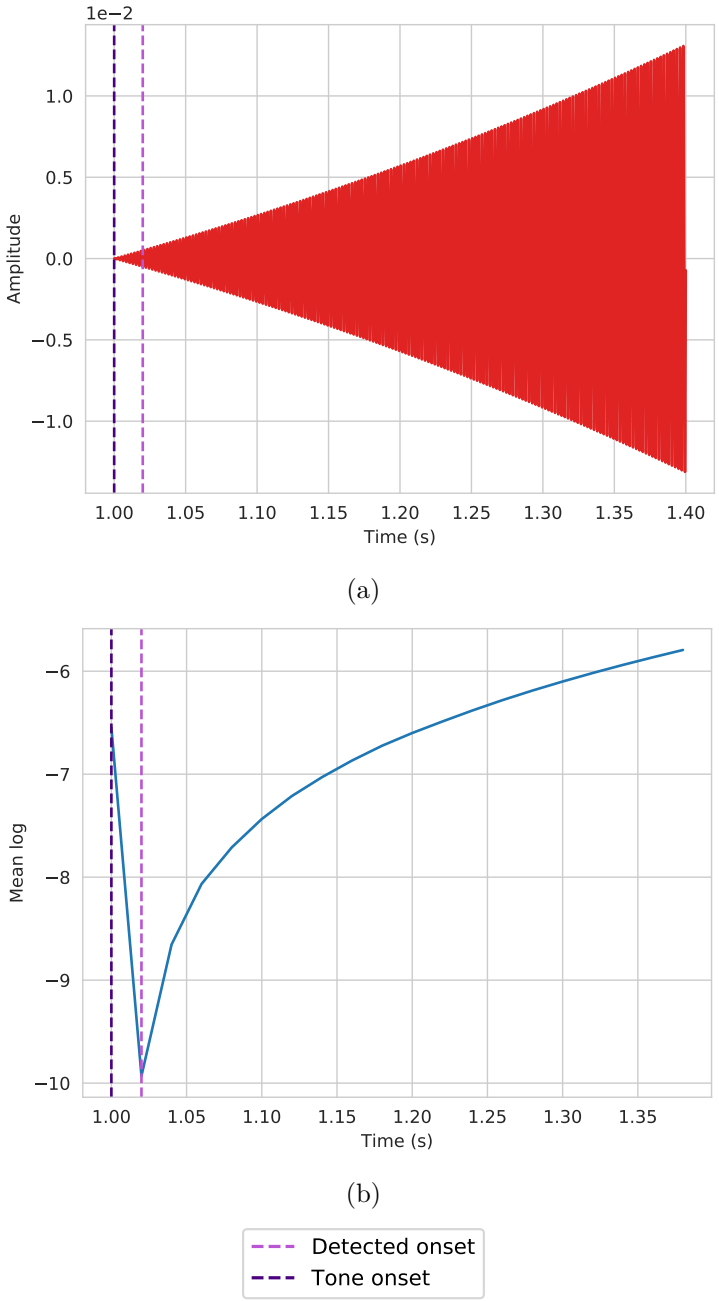


Figure 5.5: (a) audio and (b) segment mean log (with high damping level) of the 400 ms after the 440 Hz tone begins to sound. The dark purple line marks the time at which the tone begins; the light purple dashed line shows the time returned by the OnsetDetector, 20 ms after the onset of the tone.

44 rosewood xylophone notes are within the first 7 ms of the audio files. Those that are within the first 3 ms represent all the false negatives at both damping factors, although when $d = 1 \cdot 10^{-4}$, two onsets in this window are successfully found. The rubber samples contain fewer onsets quite so early. 17 onsets fall within the first 3 ms, accounting for all the false negatives at both damping factors. The remaining 27 fall outwith this time, appearing up to 27 ms into the audio files, and are all detected successfully.

During the OnsetDetector development, Zero padding was introduced in order to ensure there were enough samples available for effective backtracking, even when an onset occurs within the first 100 ms of an audio file. This was clearly successful, as early onsets are indeed detected, but these results suggest that, in order to detect events in the first 3 ms, the OnsetDetector must be improved.

Strings

As can be seen in Tables 4.9 and 4.25, the string instruments yielded better results when played pizzicato than arco. This is to be expected, for the same reasons that we would expect generally good results for percussive sounds. However, in this case the difference is not huge, as can be seen from the radar charts in Figures 4.5a and 4.8a. The F-measures for all pizzicato samples were 80.4% and 83.3% for the lower and higher damping factors, respectively. These are only 10–13 percentage points higher than the arco F-measures, which were 70% in both cases.

The overall recall does not change much between the two playing techniques, but the precision is much higher for pizzicato samples. This suggests that technique had little effect on missed detections (i.e. false negatives), but false positives were more likely for arco strings.

Tables 4.10, 4.11, 4.26 and 4.27 break down the results for each of the four string instruments tested: bass, cello, viola and violin. The most dramatic

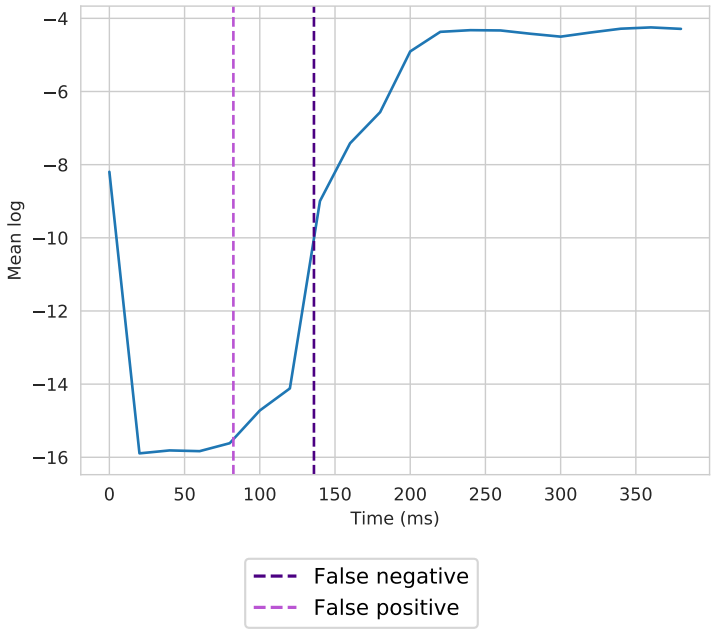


Figure 5.6: A false negative at 136 ms, and a false positive 54 ms before it, shown in the segment mean log. These occurred when analysing the C5 on a cello A string. A 'false positive–false negative' pair like this is typical of the cello and violin results.

changes between arco and pizzicato are seen in the cello and viola results, with F-measures jumping up by 24 percentage points, on average. These jumps are due to increases in both precision and recall by an average of 32 and 12 percentage points, respectively.

The bass and violin results are relatively stable, with F-measures consistently in the range from 78–86%. Somewhat unexpectedly, for a lower damping factor, the violin had a better F-measure when played arco than pizzicato. This is because the recall was much higher (96.7%, rather than 71.4%), although the precision was higher for when played pizzicato, indicating that there are still more false positives when analysing arco samples.

The reason for the large difference in results for each instrument played arco cannot be due to performance at high or low frequencies, as the bass and violin occupy the lowest and highest extents of the range.

Table 5.3: The mean difference between the hand-annotated time and the automatic time for true positives in the string instrument results, at both damping factors.

	$d = 1 \cdot 10^{-4}$, **Mean difference**	$d = 5 \cdot 10^{-4}$, **Mean difference**
Bass	−32 ms	−27.4 ms
Cello	−33 ms	−40.75 ms
Viola	−23.86 ms	−34.43 ms
Violin	−2.9 ms	−0.5 ms

Unlike the bowed vibraphone discussed above, the onset times of the notes in these samples are not ambiguous, so the false negatives are not due to human error. From inspection of its state at these points, it seems that the OnsetDetector is often backtracking too far, resulting in a false positive more than 50 ms before the a corresponding false negative. An example of this is shown in Figure 5.6, which plots the segment mean log of a cello note, the hand-annotated onset at 136 ms which was missed by the OnsetDetector and the erroneous detection at 82 ms. The time difference between these two values is 54 ms.

The true positives in the string results are, on average, approximately 9 ms earlier than the hand annotated onsets, compared with less than half a millisecond for the data set as a whole. The mean differences for each string instrument are given in Table 5.3. This shows that there is a clear tendency for even the correct results (i.e. those that were detected within 50 ms) to be early, with the exception of the violin samples, which were much closer to the manually found time.

One possible explanation for this is a particular interaction between a bow and string, first documented by Helmholtz (1885), in which the bow alternates between clinging to the string and becoming detached, then clinging again. This stick-slip pattern repeats at the same rate as the vibration of the string; i.e. when an A4 is played, the string vibrates at 440 Hz, and the bow is sticking to the string, then slipping, 440 times per second. This

has become known as “Helmholtz motion”. Much research into the mechanics of bowed strings has been carried out since then. Percival (2013) provides a summary of this, including work into the attack of bowed notes and Helmholtz motion. The situation in which Helmholtz motion is established immediately is described as a “perfect attack”. This is estimated to occur in 20–50% of notes played by professional musicians. In the case that a perfect attack is not achieved, the note will still be perceived as acceptable if Helmholtz motion begins within 50 ms.

It may be that Helmholtz motion is established relatively late in the viola and cello samples, and to a lesser extent in the bass, but not in the violin. This may mean that for a significant time at the beginning, the string is essentially being driven by white noise, rather than a coupled resonant system, and therefore the amplitude of vibration is low, perhaps imperceptible to humans, but enough to trigger a response from the corresponding detector.

This hypothesis is backed up by Figure 5.6, where it is clear that the OnsetDetector has backtracked to the approximate point at which the mean log begins to increase, although the actual note onset — the time of which was derived from visual and aural inspection of the waveform — occurs during the period when the mean log is increasing.

Additional false positives may be due to the performer’s intonation changing over the course of the note. The ‘last-first’ criterion was disabled for these tests, so false positives resulting from changes like this are less likely to occur, but are still possible. While the bow is pulling against the string, the intonation can change due to movement in the finger stopping the string or changes in string tensions arising from bowing. Figure 5.7 plots the responses of each detector in the critical band to a cello D4. The damping factor here is $5 \cdot 10^{-4}$, so the detector frequencies in the band are 4.86 Hz apart, per the investigations in Chapter 2, Section 2.2.7. The note onset,

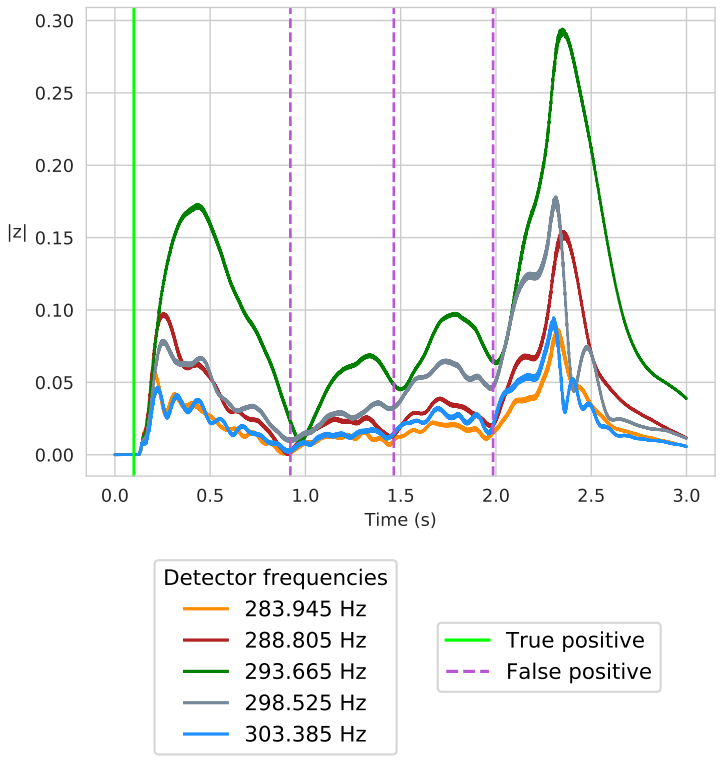


Figure 5.7: Responses of the critical band around D4 (293.665 Hz), when the damping factor is $5 \cdot 10^{-4}$. The input is the cello D4 sample, played on the A string. The bright green line marks the correctly found onset; the dashed light purple line marks additional, false positive, results from the OnsetDetector. Although these do not mark the onset of the note, it can be seen that each false detection correspond to changes in magnitude of the responses the band.

successfully found by the OnsetDetector, is marked by the lime green line, and the dashed light purple lines show the additional times returned by the OnsetDetector. Although these are false positives, it can be seen that they correspond to clear changes in the responses.

Brass

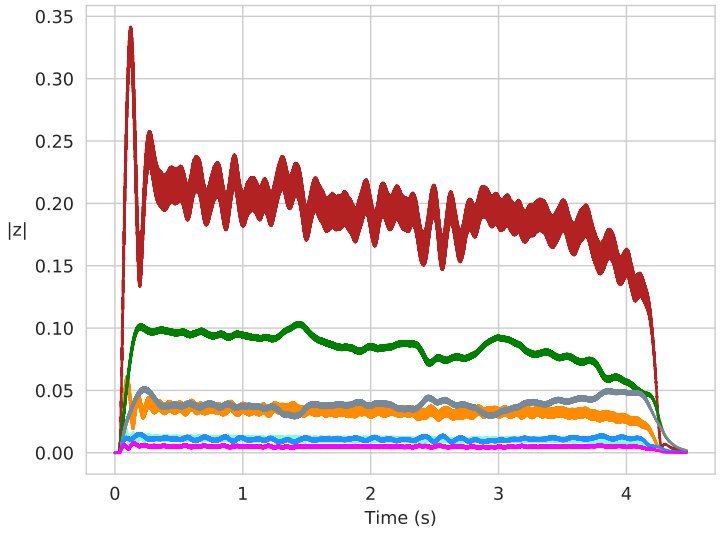
The brass instruments returned good results (see Tables 4.12 and 4.28). The lowest F-measures are given by the bass trombone: 79.2% and 72.3% for low and high damping factors, respectively. In both cases, this is due to a low recall value (70.4% and 63.0%); the bass trombone precision is in line with the other instruments, at 85–100%.

The tenor trombone achieves an F-measure of 100% at both damping factors. The remaining brass instruments have F-measures of 88–100%, indicating that there are very few false positives or negatives in this category.

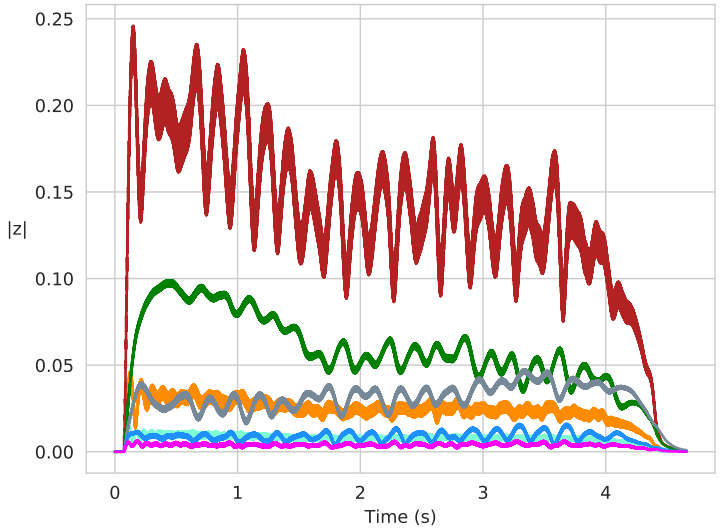
The trumpet was recorded both with and without vibrato; the results with vibrato are very similar to those without. For the higher damping factor, all three measures — precision, recall and F-measure — are all 100% when played with vibrato. This is higher than when played without, although only by a few percentage points: the F-measure without vibrato and with a high damping factor is 95.9%. This suggests that the profiles of the trumpet with and without vibrato do not vary greatly. This can be seen in Figure 5.8, which shows the responses to trumpet A4s, both with and without vibrato. The overall shape of the responses is quite similar in both cases, although there is more oscillation in the responses with vibrato.

Woodwind

The woodwind results, given in Tables 4.13 and 4.29 are much more varied than the brass. The bassoon and oboe both achieve F-measures of 100%, but the results for the other instruments are not as promising.



(a) No vibrato



(b) Vibrato

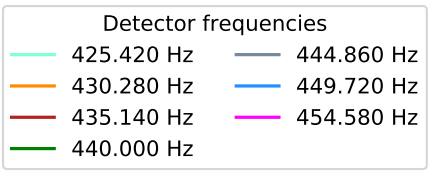


Figure 5.8: A4 trumpet responses, (a) without and (b) with vibrato. The damping factor was set to $5 \cdot 10^{-4}$. Both onsets were successfully found, with no false positives.

For all samples, except the alto flute with vibrato, the precision and recall are very similar, i.e. false positives and false negatives appear at roughly the same rate. From inspection of the results, it seems that when onset detection was not successful, we have a pair of false results: a false negative at the onset time and a false positive detected more than 50 ms before this. This pattern was previously seen in the arco cello and viola samples of the Strings section, and a significant delay in the establishment of Helmholtz motion was hypothesised as the cause. A similar process may be occurring here, as vibration of the reed couples to the vibration of air in the instrument, like the slip-stick rate of the bow couples to the vibration of a string. Therefore, a delay between the onset of the reed being blown and the appearance of resonant vibration may result in a period of time where the note is barely, if at all, audible, yet the DetectorBank is reacting.

At the higher damping factor, the soprano saxophone results were the same with and without vibrato. However, for all other woodwind instruments for which we have samples both with and without vibrato, and at both damping factors tested, F-measures for samples without vibrato were between four and 24 percentage points higher.

Tables 4.14 and 4.30 compare the results for all brass and woodwind instruments, where the presence or absence of vibrato is known. In both tables, the F-measure is approximately twice as high when no vibrato was used. This is due to the woodwind samples, as the results for the trumpet samples did not change significantly when vibrato was used, with F-measures between 90 and 100%.

Piano

As the dynamic level is raised, the changes in the piano results are consistent for both damping factors, which can be seen by comparing the shapes of the radar plots in Figures 4.6b and 4.9b and the values in Tables 4.16 and 4.32.

The precision is 100% for all three dynamic levels at the lower damping factor, i.e. there are no false positives. At the higher damping factor, it is greater than 90% for ***pp*** and ***mf*** samples, but drops to 71.7% at ***ff***.

There is more variation in the recall values. The quietest piano samples had the lowest recall. Going up to the next dynamic level, the recall increases by 37 and 29 percentage points for the lower and higher damping factors, respectively. These increase to 81.8% and 92.0% at the loudest dynamic level. It may be that false negatives are more likely to occur in quieter samples because the segment mean log is not reaching the threshold, which was held constant for all tests at the same damping factor. Conversely, the loudest samples return more false positives (at the higher damping factor), perhaps because the responses are hovering around the threshold level, so can easily tip over and trigger a new detection.

The low recall values for ***pp*** samples reduces the F-measure at this level; however, for samples ***mf*** and louder, the recall and precision are sufficiently high to yield an F-measure of approximately 80–90%.

The idea of adaptive thresholding will be discussed in Section 5.3.1. This feature may help to improve consistency of results at different dynamic levels.

5.2.3 Octaves

At both damping factors, 0% of the onsets in octave 0 were found (see Tables 4.17 and 4.33). This may be because, as shown in Figure 4.2, the data set has very few recordings in this octave; just those of the lowest three notes on a piano — A0, B♭0 and B0 — which have fundamental frequencies from 27.5 to 30.9 Hz. Only the ***ff*** piano samples have all three of these notes; the ***mf*** samples begin at B0 and ***pp*** at B♭0. This means there are only six samples in octave 0. Having so few samples makes it difficult to draw conclusions about the performance at these frequencies. However,

we can study the OnsetDetector state to ascertain whether there may be deficiencies in the algorithm.

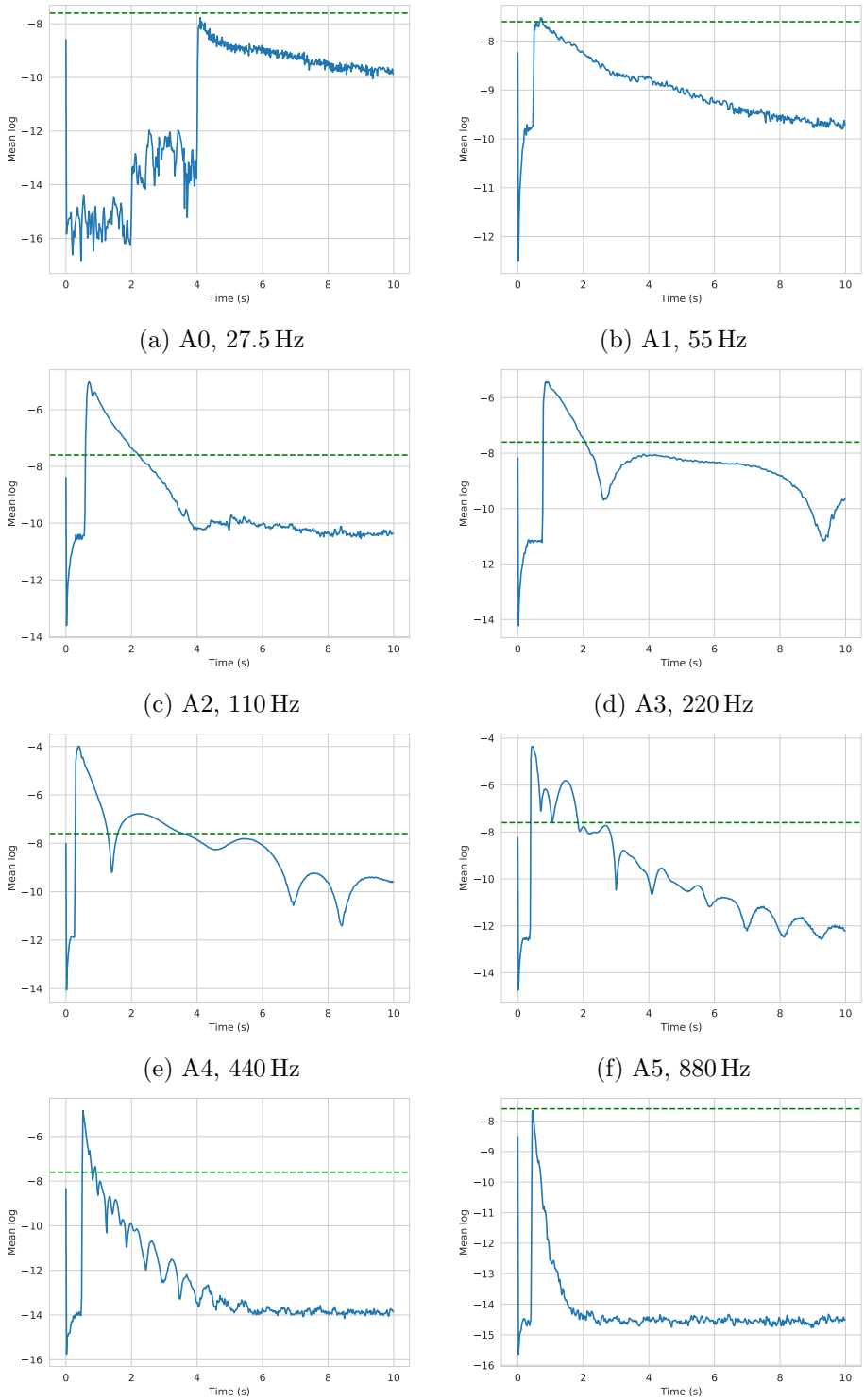
Chapter 2 investigated the DetectorBank response to audio input comprising sine tones at various frequencies. Figure 2.8a showed the DetectorBank output when driven by consecutively increasing tones, at the fundamental frequencies corresponding to the notes from A0 to A1. This, along with Figure 2.2a, show that the responses to tones generated at the lower end of the spectrum are not significantly different from those higher up the scale.

However, one of the test audio extracts used in Chapter 3, *Swan Lake* extract 1, comprised notes at the low end of the scale and consistently had the worst results during OnsetDetector development. Section 3.4.2, More true positives, suggested that improvements needed to be made to more reliable detect low frequencies. Clearly, responses at low frequencies merit further investigation.

Figure 5.9 shows the segment mean logs for 'A's played on the piano in successive octaves. The samples at the highest dynamic level are used here; the ***mf*** and ***pp*** samples follow a similar pattern. The threshold, which was constant for all samples, is marked with a dashed green line. It can be seen that at the lowest octave, A0 (Figure 5.9a), the response at the onset misses the threshold by a whisker: the threshold $= \ln(0.0005) = -7.60$ and the maximum segment mean log is -7.77. In octave 1, the mean log just exceeds the threshold. In octaves 2–6, the threshold is well exceeded; however, in the highest octave, the mean log values drop again.

Adjusting the threshold for OnsetDetectors operating at the highest and lowest frequencies may solve this problem.

At both damping factors, octave 1 yielded the highest recall and F-measure and octave 7 the highest precision. Tables 5.4 and 5.5 show that, when calculated over octaves 1–7, the mean values of the recall and F-



(a) A0, 27.5 Hz

(b) A1, 55 Hz

(c) A2, 110 Hz

(d) A3, 220 Hz

(e) A4, 440 Hz

(f) A5, 880 Hz

(g) A6, 1760 Hz

(h) A7, 3520 Hz

Figure 5.9: Segment mean logs for piano notes increasing by octave, with the threshold marked by a dashed green line.

Table 5.4: Mean and standard deviation of results, calculated from octaves 1–7. Low damping factor.

	Precision %	**Recall %**	**F-measure %**
Mean	79.934	26.358	38.354
SD	5.271	10.309	11.73

Table 5.5: Mean and standard deviation of results, calculated from octaves 1–7. High damping factor.

	Precision %	**Recall %**	**F-measure %**
Mean	75.011	40.594	52.209
SD	12.66	5.980	5.748

measure are higher, and the corresponding standard deviations lower, at $d = 5 \cdot 10^{-4}$. The opposite is true for the precision. At every octave in this range, the recall and F-measure values are higher at the higher damping factor. The precision values are either lower or very similar.

Table 4.17 gives the results broken down by octave at the lower damping factor. Comparison with Table 4.33 shows that the recall varies more from octave to octave at low damping than high, as would be expected from the standard deviations in Tables 5.4 and 5.5.

Although the recall is higher at the higher damping factor, the precision is lower. To put this another way: there are both more true positives and more false positives. This suggests the inconsistency between damping factors may be due to thresholding: at high damping, there were simply more detections. The whole data set comprises 8326 onsets. At a damping factor of $5 \cdot 10^{-4}$, there were 4717 detections, of which 3326 were true positives; at $d = 1 \cdot 10^{-4}$, there were only 2902 detections, of which 2231 were correct.

5.2.4 Results at different damping factors

The results returned by the OnsetDetector for the string, brass, woodwind and piano samples were consistent at both damping factors. However, there

are some situations in which was not the case, including across octaves, as discussed above, the guitar samples and in some of the percussion results.

In almost all cases investigated in Chapter 4, when results for different damping factors varied, even by a small margin, the lower damping factor yielded a better precision, i.e. fewer false positives, and the higher a better recall, i.e. fewer false negatives, which often resulted in a higher F-measure as well. As mentioned above, the tests with a higher damping factor returned more detections, which ultimately meant more true and false positives. In part, this may be due lack of threshold optimisation, particularly when $d = 1 \cdot 10^{-4}$. The low damping tests took approximately 4.5 times longer to run, which meant small tweaks to parameters like the threshold were more expensive, and consequently the threshold for the higher damping level was more finely tuned.

One exception to this is the guitar, the overall results for which can be found in Tables 4.3 and 4.19. The recall did not change much with damping factor (82.4 and 83.2%), but at $d = 1 \cdot 10^{-4}$, the precision was more than twice that at $d = 5 \cdot 10^{-4}$, which in turn raised the F-measure to 80%, as opposed to 44.6%.

Tables 4.15 and 4.31 present the guitar results for the three dynamic levels. Unlike the piano samples, the guitar results were very dissimilar at each damping factor. At the lower damping factor, the precision, recall and, therefore, F-measure increase with the damping level in a fairly uniform manner (see Figure 4.6a). However, Figure 4.9a shows that, at the higher damping factor, although the recall increases in much the same fashion, the precision is static at around 30%, less than half the lowest precision found in Table 4.15. This limited precision affects the F-measure, which is approximately 45% at every dynamic level tested.

Inspection of the guitar audio files, for example the C3 given in Figure 5.10, shows that there are large DC offsets occurring throughout. These

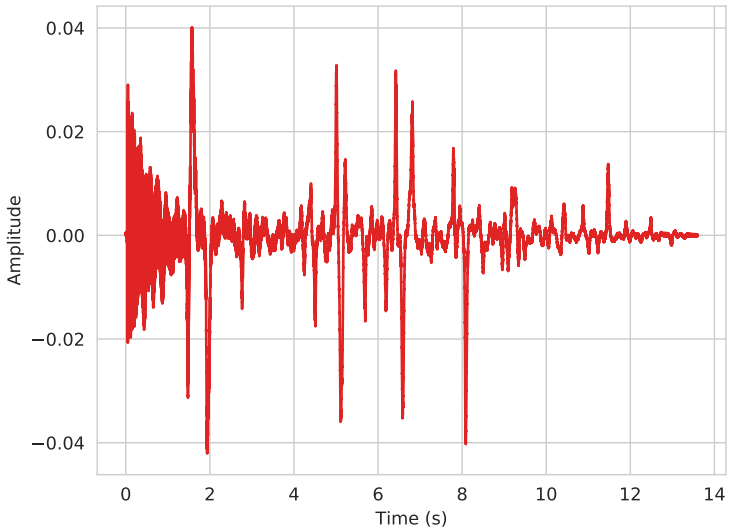
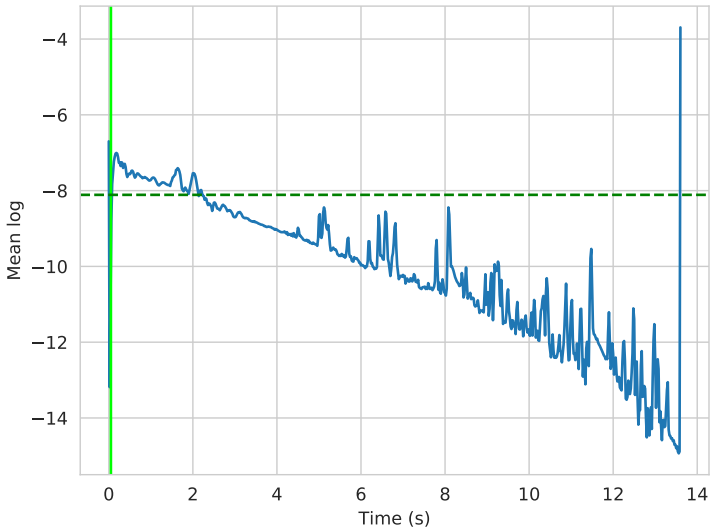


Figure 5.10: Waveform of a C3, played on a guitar A string. The onset of the note occurs at in the first few milliseconds and the note is allowed to ring out for almost 14 seconds. Large variations in amplitude are visible, but not audible, in this period.

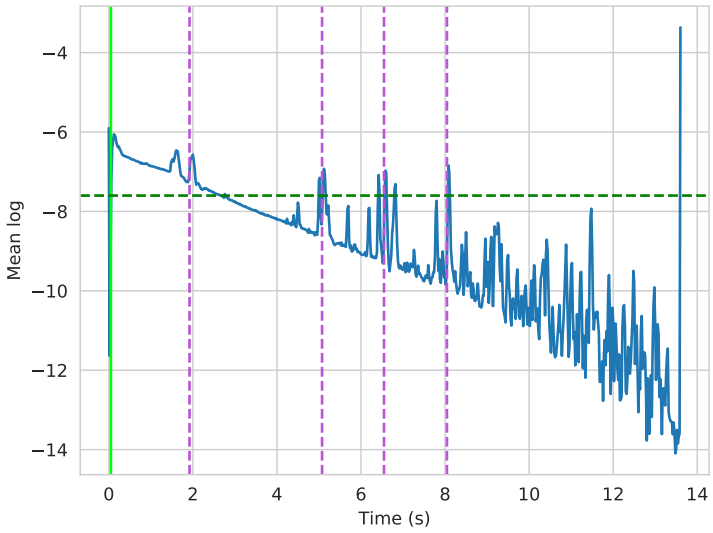
may have arisen due to poorly applied normalisation, which attempted to remove a DC offset, but inadvertently added one instead. At the lower damping factor, these peaks are mostly below the threshold level, but not at the higher damping factor. Figure 5.11 shows the segment mean logs in both cases, along with the threshold and locations of true and false positives returned.[4] From this, it seems that the poor OnsetDetector results for the guitar samples are due to artefacts in the original recordings, rather than any deficiency in the software.

From Tables 4.3 and 4.19, we can see that the percussion results followed the opposite pattern to the guitar: the change in precision was negligible, but the recall and F-measure were more than twice as high at the higher damping

[4]In these graphs, a sudden increase in mean log can be seen at the very end of the sample. These occur because the DetectorCache always returns full segments of values. If samples are requested after the end of the input buffer, the DetectorBank returns zeros. This can result in the final segment largely consisting of zeros. As mentioned when Zero padding was introduced, in this case, the OnsetDetector regards log(0) as equal to 0. If a segment is largely zero, this will drag the mean value up. As this only affects the final segment, it will not cause erroneous detections.



(a) Damping factor: $1 \cdot 10^{-4}$; threshold: $\ln(3 \cdot 10^{-4})$



(b) Damping factor: $5 \cdot 10^{-4}$; threshold: $\ln(5 \cdot 10^{-4})$

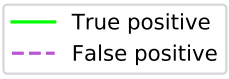


Figure 5.11: Segment mean logs for the guitar C3 shown in Figure 5.10 at both damping factors tested, with the thresholds in each case marked by the dashed green line. The onset at 43 ms is successfully found in both cases, but the analysis at the higher damping factor also returns a number of false positives. In (a) the peaks in the mean log due to the DC offset are lower than the threshold; in (b) several of them are not, and are identified by the OnsetDetector.

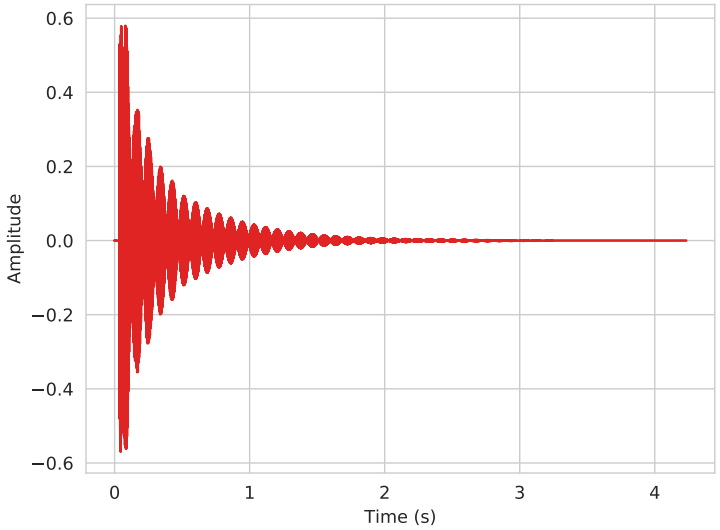
factor. Tables 4.8 and 4.24 show the results for each percussion instrument, most of which were consistent across the damping factors. We will not consider the rolls here; Section 5.2.1 discussed the 'last-first' criterion, which is the main factor determining the rate of correct detections.

The percussion instrument for which the results varied notably were the crotales — a set of small tuned cymbals — which saw a huge drop in precision, from 84.6% to 40.7% when the damping factor was raised. Inspection of the waveforms revealed a large amount of beating in these samples, which is then picked up by the OnsetDetector. Beating arises when there are two similar modes of vibration occurring simultaneously and interfering with each other. The amplitude of the resultant waveform oscillates at a rate of the difference between the two frequencies. Figure 5.12 shows both the waveform and segment mean log for one of the samples. Here, the beating is particularly pronounced, and many of the beats are identified by the OnsetDetector. In fact, this sample alone accounts for almost half of all the false positives in the crotale results at the higher damping factor and all four false positives at the lower. Once again, the disparity in results seems like an issue which is largely caused by the choice of threshold at each damping level.

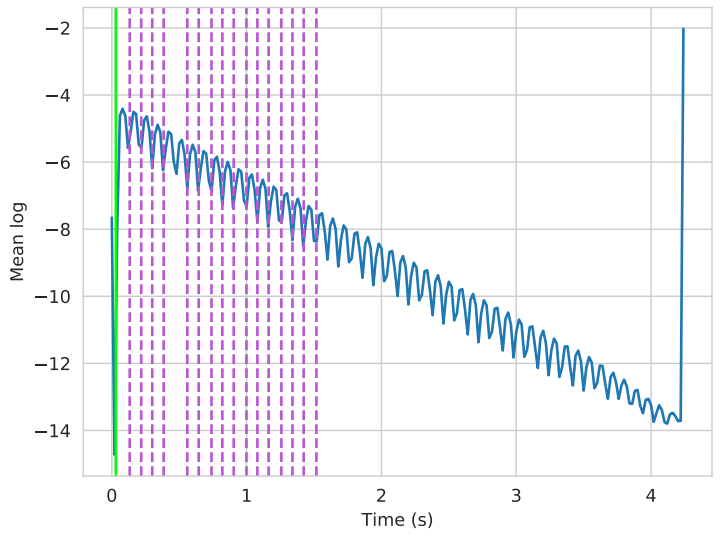
5.3 Further work

5.3.1 OnsetDetector improvements

Many of the problems discussed throughout Section 5.2 arise because of thresholding. Therefore, a method for automatically deriving the threshold — and adjusting it as necessary throughout the process of onset detection — may drastically improve the OnsetDetector results. There is also a great deal of variation in the threshold values used to test the OnsetDetector in Chapters 3 and 4. Determining a suitable threshold for any given audio



(a)



(b)

Figure 5.12: A B7 crotale sounding, beating at a rate of approximately 11.5 Hz. In (a) the waveform is shown, and (b) plots the mean log at the higher damping factor, in which a number of false positives are identified.

file is not a task that should be left to the user; it should be an internal parameter of the OnsetDetector.

In image processing, thresholding is often used to separate an object from the background. In this field, uniform thresholding works in a similar way to thresholding as used in this project: pixels above, below or in a given range of values are selected (Nixon & Aguado 2012). It also encounters similar problems: an appropriate value must be known a priori and it does not account for differing levels of intensity — for example, due to lighting — across an image. Adaptive thresholding algorithms were developed to help solve these problems. These often work by analysing probability distributions or splitting the image into smaller parts and regarding each independently (Dey et al. 2014).

Applying a threshold to DetectorBank output is a different problem. We are not trying to separate foreground and background objects, rather to determine if a response is strong enough to merit further investigation. Backtracking is quite an intensive process, as it requires multiple operations to be carried out at every sample of interest, so we wish to do so only when strictly necessary.

In the switch from a global, uniform threshold to an adaptive one, setting a threshold for each band that can be set and varied as required, rather than one threshold used by every OnsetDetector in the NoteDetector, may also improve the results. Adjusting the threshold for detectors at the lowest frequencies in the musical range was suggested when discussing the results at different Octaves, as the magnitude of the responses to the very lowest frequencies — and sometimes very highest — was often just missing the threshold.

Although it will be a difficult task, developing a better thresholding technique may be the most effective way of reducing the number of false negatives and positives in the OnsetDetector results.

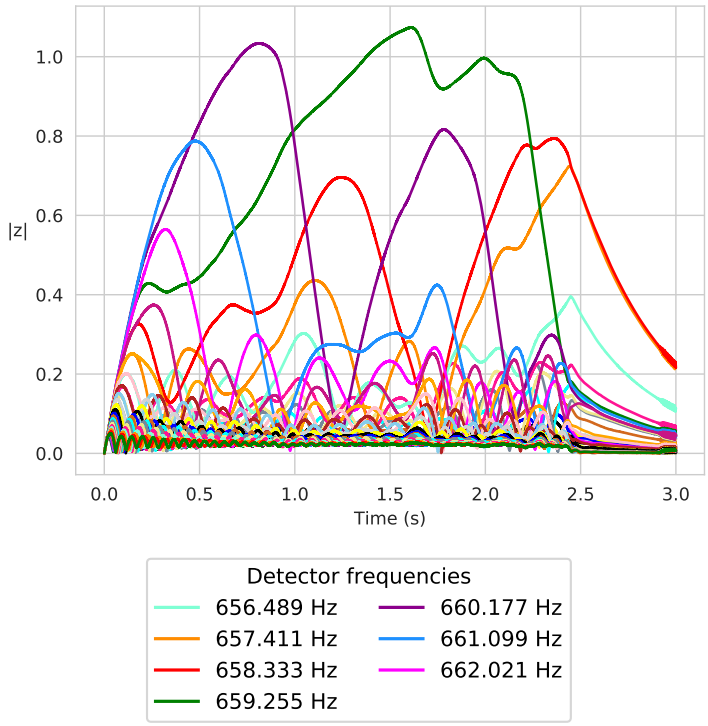


Figure 5.13: Band of responses to an E5, played on a flute. The band consists of 43 detectors around 659.255 Hz; for clarity, only those responses which exceed a magnitude of 0.4 are listed in the legend.

5.3.2 PitchTracker

The concept of critical bands, a key feature of the OnsetDetector, also lends itself to pitch tracking, as the responses in the band are tuned to different frequencies, so changes throughout the note can be seen.

For example, Figure 5.13 shows the responses to an E5 played on a flute. The damping factor used here is $1 \cdot 10^{-4}$ and minimum bandwidth detectors are requested, therefore 43 detectors are created at 0.922 Hz increments, centred around 659.255 Hz, as this is the fundamental frequency obtained from the mathematical expression relating a reference frequency (440 Hz) to the size of a semitone ($2^{1/12}$). In reality, the frequencies produced by a flute depend on both the performer and the instrument itself. It can be seen in Figure 5.13 that several detectors are responding. The detectors cover

the range 656.489 to 662.021 Hz, but only those for which the responses are of significant amplitude are listed in the legend. It can be seen that, at the beginning of the note, the responses at the higher end of this range dominate.

The 660.177 Hz response reaches its peak at approximately 800 ms. After this, the detector at the centre frequency has the largest response. Then, the detectors immediately below this respond most strongly from about 2 seconds until the end of the note. The pitch descends over the course of the note, although only by a few Hertz; to a listener, it does not sound like the note is getting flatter, although a certain amount of vibrato can be heard, especially towards the end.

The relative strength of these responses, and the frequencies they a responding to, could be used to determine an overall frequency at each sample. There are a number of ways this could be achieved. One possibility is to use a weighted average of the frequencies, with the amplitudes of each detector at the current sample set as the weights. This can be refined by using a subset of detectors, rather than the whole band: at every sample, detectors are selected if their characteristic frequency is close to the average frequency calculated at the last sample.

This method cannot be used to calculate the first average frequency value, as there are no previous samples. In this case, the average frequency is found by taking the weighted mean of all detectors in the band. One more improvement can be made to this algorithm: rather than simply setting the weights as the value of the corresponding detector response at the current sample, this value is squared.

This algorithm is used to generate the average frequency of the flute note, shown in Figure 5.14. In this case, the subset of detectors was selected by finding those within a tenth of a semitone above or below the previous sample's average frequency. It follows the expected pitch trajectory, starting

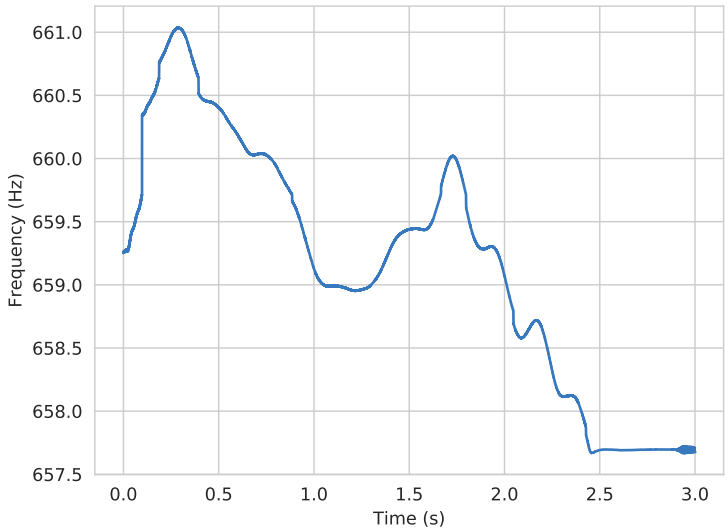


Figure 5.14: Average frequency of responses in Figure 5.13. At each sample, a subset of detectors are used: their frequencies are averages using their $|z|$ values as weights.

above the centre frequency and ending below it.

There is a problem with this technique as described so far: a detector begins to respond when its frequency appears in the input, but it takes time for the response to reach its maximum. This means the weight for this frequency will initially be low, leading to a lag between the frequency appearing and the average frequency arriving at this value.

This can be seen when running this algorithm on a chirp signal. Figures 5.15 and 5.16 show the DetectorBank responses and average frequency, respectively, generated for an input signal which begins at 440 Hz and increases to 466.164 Hz, the fundamentals of an A4 and B♭4, one semitone up. Figure 5.16 also marks the expected frequency, linearly increasing from the start to stop values. The average frequency is mostly lower than the expected, because lower frequency detectors are in relaxation at a higher amplitude when the next detector starts reacting.

It also illustrates another problem: at the first sample, all the detectors are used to calculate the average and the responses amplitudes are all very

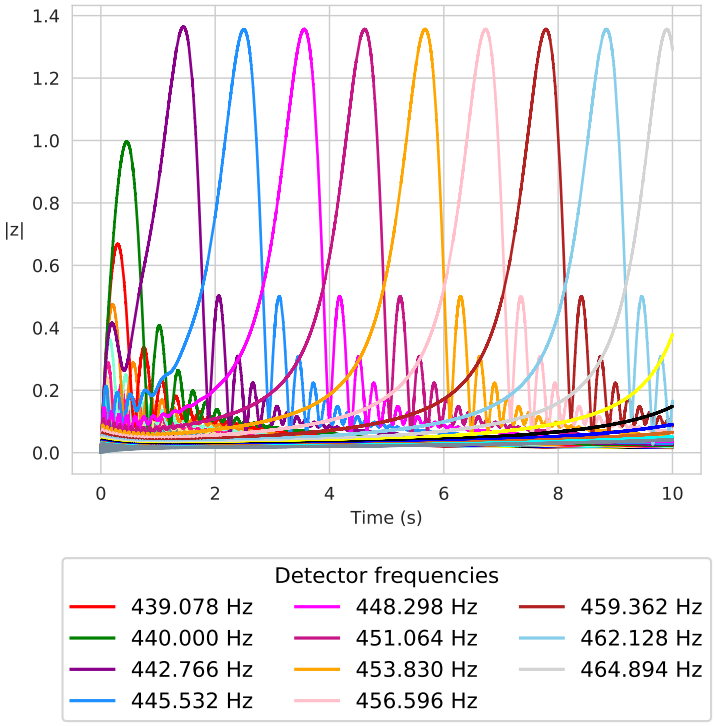


Figure 5.15: DetectorBank responses to a chirp signal, which starts at 440 Hz and ends at 466.164 Hz. Again, only the responses with significantly large amplitude are given the legend.

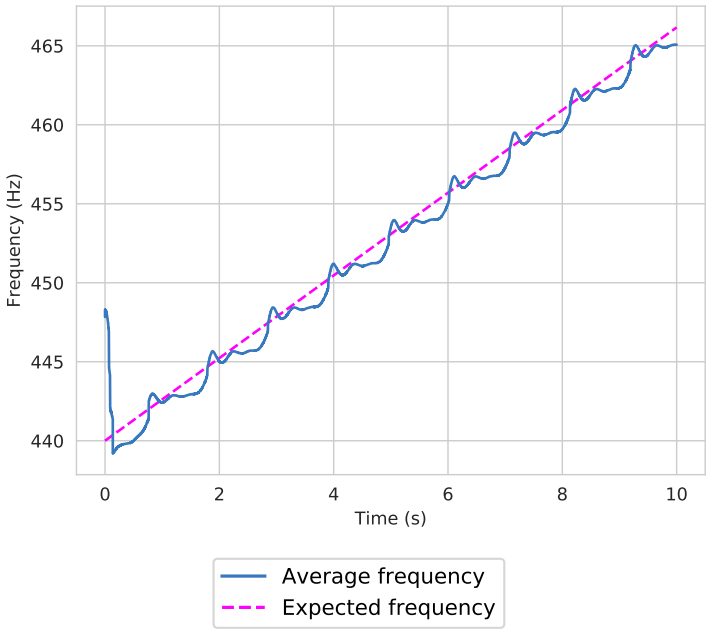


Figure 5.16: Average frequency, and the frequency that would be expected at every sample, calculated using the responses in Figure 5.15.

low. In the case of the data shown in Figure 5.16, the first average frequency value is 447.837 Hz.[5] Disregarding frequency values calculated when the average amplitude is very low may be one way to fix this problem.

The issue of how to calculate the frequency without the time lag will be harder to solve, but will be required for a full implementation of the PitchTracker.

Another potential pitfall is resource usage. It may be possible to design the PitchTracker in such a way that limits the number of samples to be analysed to only those values following a result from the band's OnsetDetector, rather than every sample from the DetectorBank, i.e. only when a note is present in the band. This should keep additional resource requirement to a minimum. Implementing this may require the OnsetDetector to

[5]To adequately cover this chirp, 30 detectors were placed at 0.922 Hz increments from half a semitone below 440 Hz to half a semitone above 466.164 Hz. When all weights are equal, this results in an average of 447.837 Hz.

be extended to also return note offsets; however, these do not have to be exact, so could potentially be determined on the segment level in order to avoid any more intensive sample-by-sample analysis in the OnsetDetector. It would also need the OnsetDetector to pass detections to the EventDetector as and when it found them, so the EventDetector can start and stop the PitchTracker as required.

5.3.3 NoteDetector

Once the PitchTracker is implemented — and improvements made to the OnsetDetector — they can be combined to make a NoteDetector, in the manner shown in the UML diagram in Figure 3.19 at the end of Chapter 3. This design features an object called the EventDetector, which is created to manage the OnsetDetector and PitchTracker for a given band. A NoteDetector searching for n frequencies will therefore consist of n EventDetectors.

It will then be the NoteDetector's responsibility to anlayse the events returned by the EventDetectors and classify them as note onsets, intra-note events or false positives. The NoteDetector will have information from all bands available to it simultaneously — unlike the EventDetectors and their components, which are independent from each other — so will be able to contextualise an event by comparing it with information from other bands at the same time.

This should mean the NoteDetector can analyse polyphonic audio files, as each band's data is independent from the others.

The classification process could be implemented using statistical methods like hidden Markov models (Ghahramani 2001), which have been used before in musicological analysis (Devaney et al. 2011). HMMs can take into consideration information about expected note-to-note changes (from melody trajectories or harmonic and rhythmic sequences) or intra-note changes (like expected pitch changes due to vibrato). Data concerning expected

melodic or rhythmic patterns has been presented in Huron (2006).

If a sufficiently large data set could be accessed, containing both note onsets and information about intra-note events, it may possible to use machine learning to classify events. However, creating such a data set would be a huge undertaking.

5.4 Summary

This project was established with the aim of developing software which is capable of simultaneous time-frequency discrimination. This could have applications in many areas of engineering, but the specific task this software is developed for is the detection of the onset times of notes in musical audio.

Current algorithms attempting to do this use general signal processing techniques, like looking at spectral changes, and are often implemented using short-term Fourier (or similar) transforms, with the result that the time and frequency resolution are constrained by the uncertainty relation, $\Delta t \Delta f \geq 0.5$.

However, the auditory system knows no such bounds. Humans are capable of resolving pitch in millisecond pulses. The just-noticeable difference for changes in frequency is a fraction of one percent. The loudest tolerable sounds are a million times louder than the softest detectable ones. These phenomena occur because of nonlinearities in the auditory system. Small cells in the cochlea — the outer hair cells — have been found to tune, compress and sharpen the response of the auditory system. The mathematical system which describes the response of the outer hair cells — the Hopf bifurcation — was chosen as the engine for the software presented here.

The DetectorBank, a bank of nonlinear tuned resonators in software, each operating at a Hopf bifurcation, succeeded in performing time–frequency analysis to a better resolution than uncertainty principle would suggest is possible, as the output signal does not need to be measured simultane-

ously in both domains. For example, in discussion of the DetectorBank behaviour at Low frequencies, it was seen that a frequency difference of 1.6 Hz was discriminated in as little as 230 ms. This is 82.5 ms faster than the minimum time resolution given by the uncertainty relation. In this case, $\Delta t \Delta f = 0.375$. This was achieved at the highest sample rate tested, 192 kHz, but investigations of the DetectorBank generally focussed on a lower sample rate, 48 kHz. At this value, the simultaneous frequency and time resolutions still exceed the limit, with 1.6 Hz discriminated in 265 ms, therefore $\Delta t \Delta f = 0.433$. Again, these values do not contract the uncertainty principle, as they are not obtained from observing the same signal.

The DetectorBank was then used as the basis for an OnsetDetector. The basic idea was to do onset detection in two stages: the first should identify points where a note may have occurred and the second should then analyse the responses at a greater level of detail to verify or reject the detection.

Several different approaches to stage one were prototyped. The prototypes were judged not only by the accuracy of results, but by the resources required. The vast majority of samples in the input do not correspond to note onsets; we want to determine whether a time period is worth investigating in more detail, without wasting resources.

Once again, a phenomenon from the auditory system was adopted in the design of the software. In audiology, the basilar membrane can be modelled as a series of overlapping filters, known as critical bands. This idea was modified for use here. Given a centre frequency, an OnsetDetector constructs a band of detectors up to half a semitone above and below this, with the spacing between each determined by the detector bandwidth.

The method chosen for stage one of onset detection regards the DetectorBank output in short segments. By taking the mean of the log of each detector's output at every sample, a block containing 20 ms of samples from k detectors can be reduced to a single number. The pattern of these values

('segment mean logs') is then analysed, using two simple rules: (i) has the mean log been increasing for at least N consecutive segments? and (ii) is the end of this increasing run greater than a given threshold? If both of these are true, stage two of the OnsetDetector can then be started.

Stage two involves backtracking to find the point at which the detectors began to respond. This is done by comparing the a value with the mean log of the N preceding values, and iterating backwards throughout the samples until the point at which the response began to rise has been found.

This was then honed, to increase the accuracy of the results returned. In doing so, one more condition was added to stage one of the OnsetDetector: the requirement that the run of values has increased by a significant amount. This condition reduced the number of false positive detections that occur within a note. However, these detections are only classed as errors because there is no data against which to check them. With this condition removed, the OnsetDetector may perform well on a data set that contains information about the changes within notes, as well as their onset times.

The OnsetDetector was tested using a repository of samples of individual notes recorded on a variety of instruments. Although this uses a different data set, these results can be compared with the current state-of-the-art algorithms tested by MIREX. The OnsetDetector results were broadly consistent with the other algorithms across various instrument classes, with the results for brass and arco strings comparing particularly well.

The results of the tests using the University of Iowa repository of musical instrument samples are promising, but more work is needed to improve the onset detection. It may also be the case that new recordings are required for the guitar samples — which exhibited large DC offsets — and the string and woodwind instruments, where it was hypothesised that there may be a significant delay in the establishment of sympathetic resonance between the bow or reed and string or air in the instrument, resulting in low amplitude

vibration which is inaudible to listeners, but could be picked up by the DetectorBank.

Additionally, expanding data set to include information about intra-note events would be required to test the OnsetDetector thoroughly.

Apart from using a more comprehensive data set, improving the thresholding technique may be the most effective way of reducing both false negatives and false positives in the OnsetDetector results.

It may also be possible to create a PitchTracker which operates on the same data as the OnsetDetector, using the method outlined in Section 5.3.2 above.

These two objects — the OnsetDetector and the PitchTracker — can then be brought together to create a NoteDetector capable of detecting not only the onset time and pitch of a note, but continuous pitch data and multiple event times of notes sounding simultaneously at different pitches, at a resolution impossible for algorithms that rely on Fourier transforms or similar techniques.

Appendix A

Code extract syntax

The code extracts in this thesis are written in Python. Python is a high-level language, which — when augmented with comments — should be human-readable. However, for any readers unaware of basic programming syntax, a brief explanation is given here.

Comments

All text that appears after a # in a line is a comment and will be ignored by the program.

Operators

The operator `=` assigns a value to a variable: read it as 'becomes equal to' rather than 'is equal to'. To test whether one variable is equal to another, `==` is used.

The syntax `a += b` is a shorthand for `a = a + b`. Similar shorthand also exists for subtraction (`-=`), division (`/=`) and multiplication (`*=`).

The operators `<`, `<=`, `>` and `>=` mean less than, less than or equal to, greater than and greater than or equal to respectively.

Control flow

`for` and `while` loops will execute a portion of code repeatedly. Code extract A.1 demonstrates a simple `while` loop: the code inside the loop (the indented lines) will keep executing until the condition it is given is false — in this case, when `i` is no longer less than 10.

```
# create a variable 'i', initially 0
i = 0
# while i is less than 10...
while i < 10:
    # ...increment i
    i += 1
    # display the current value of i
    print(i)
```

Code Extract A.1: `while` loop

This example — simply incrementing a variable until it reaches a given value — could also have be written using a `for` loop and the `range` keyword.

```
for i in range(10):
    print(i)
```

Code Extract A.2: `for` loop

Here, `for` creates a variable `i`, which takes a different value every time round the loop (in this case, every value from zero up to 10).

`if` and `else` statements are also used for control flow. If the condition after `if` is not met, its contents are not executed; if an `else` statement is supplied, that will be executed. `elif` is a shorthand for `else if` and is used when an initial `if` condition is not met and you want to test another.

```
if a == b:
    print('a is equal to b')
elif a < b:
    print('a is less than b')
else:
    print('a is greater than b')
```

Code Extract A.3: `if-else` statement

Functions

Lines of code can be grouped together into functions. The syntax for calling a function is `result = func(arg)` where `arg` is an input to the function `func` and the output will be assigned to the variable `result`.

Lists

Square brackets `[]` are used to create and access lists. New values can be added to a list with the `append` method.

```
1 lst = [] # make an empty list
2 # add values to the end of the list
3 for i in range(5):
4     lst.append(i)
5 first = lst[0] # access the first value
6 last = lst[-1] # access the last value
```

Code Extract A.4: Lists

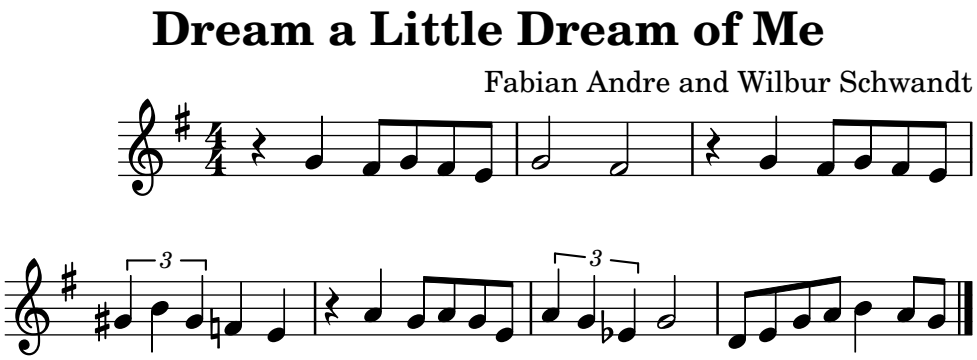
Appendix B

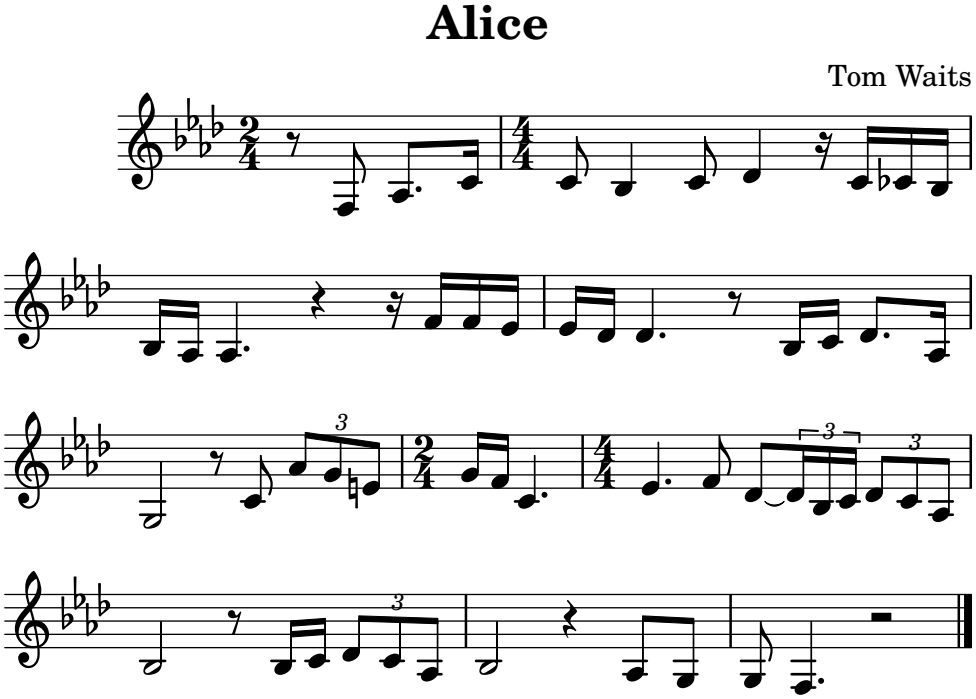
Test audio scores

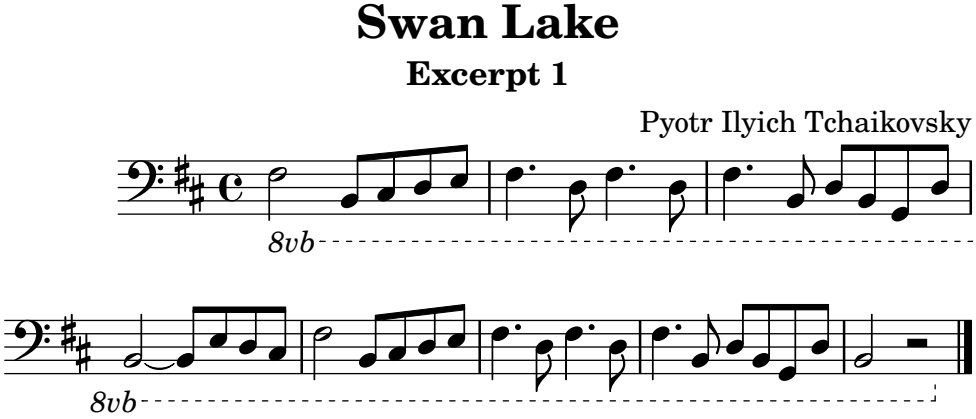
When testing the OnsetDetector (see Chapter 3), the following short musical extracts were used. Details of these extracts can be found in Table 3.1, reproduced here for convenience.

Table B.1: Reproduction of Table 3.1: Test audio details

Title	Instrument	Note range	Freq. range
Dream a Little Dream of Me	Digital piano	D4–B4	293.7 Hz–493.8 Hz
Alice	Digital piano	F3–A♭4	174.6 Hz–415.3 Hz
Swan Lake excerpt 1	Digital piano	G1–F♯2	49.0 Hz–92.5 Hz
Swan Lake excerpt 2	Digital piano	B2–C♯4	123.5 Hz–277.2 Hz
Before All Things	Soprano voice	F4–A♭5	349.2 Hz–830.6 Hz



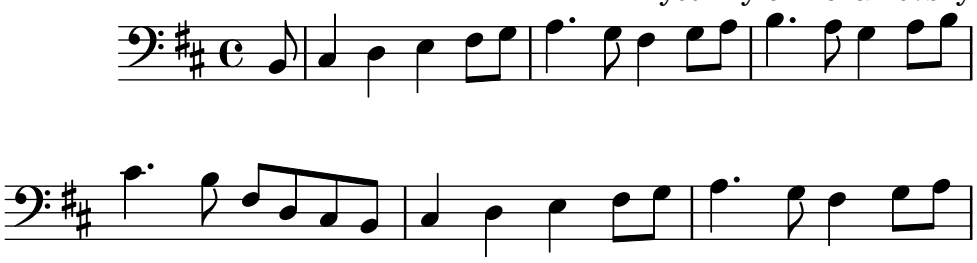

Alice
Tom Waits


Swan Lake
Excerpt 1
Pyotr Ilyich Tchaikovsky
8vb
8vb

Swan Lake

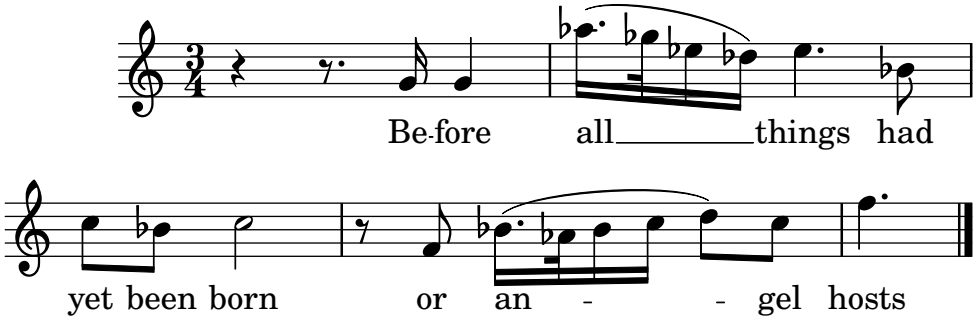
Excerpt 2

Pyotr Ilyich Tchaikovsky



Before all things

Text by Yunus Emre (1238-1320) Graham Hair



Appendix C

Numerical Methods

This chapter presents the two numerical methods provided as options when using the DetectorBank, as detailed in 2.1.2.

C.1 Foruth order Runge-Kutta

The Runge-Kutta method uses approximations to the gradient of the curve to find the solution to a first-order differential equation (James 2011).

The first order Runge-Kutta method (which is the same as Euler's method and the forward-difference approximation) uses only one approximation of the slope.

The widely-used fourth order Runge-Kutta uses four approximations of the curve within a certain step-size (see Figure C.1).

To find the solution to $dy/dt = f(y, t)$, with a known initial condition

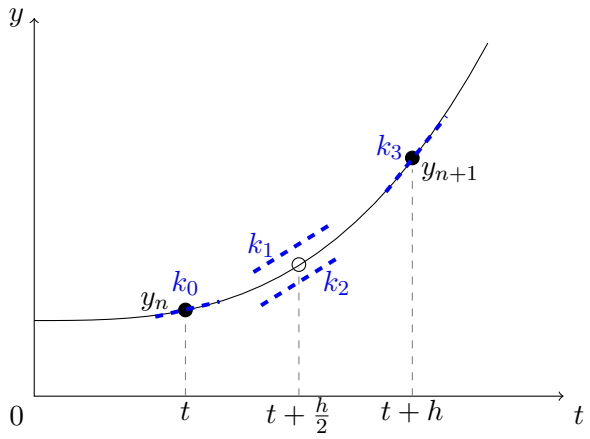


Figure C.1: Fourth Order Runge-Kutta: the solution of dy/dt can be approximated by combining the slopes at four points: k_0, k_1, k_2 and k_3.

$y(t_0) = y_0$, the following slope approximations

$$k_0 = f(y_0,\, t_0) \tag{C.1a}$$

$$k_1 = f\left(y_0 + k_0\frac{h}{2},\, t_0 + \frac{h}{2}\right) \tag{C.1b}$$

$$k_2 = f\left(y_0 + k_1\frac{h}{2},\, t_0 + \frac{h}{2}\right) \tag{C.1c}$$

$$k_3 = f\left(y_0 + k_2 h,\, t_0 + h\right) \tag{C.1d}$$

can be combined to find $y(t_0 + h)$:

$$y(t_0 + h) = y_0 + \frac{h}{6}\left(k_0 + 2k_1 + 2k_2 + k_3\right) \tag{C.2}$$

When calculating $f(y, t)$ in a sampled-time system, the step-size, h, used to find t values is not the same as that used to find y. For t, the smallest time increment possible is one sample, so h becomes two samples. For y, h is the length of time taken for two samples, denoted 2δ.

Therefore, for a discrete-time system with a sampling frequency of $1/\delta$ Hz,

Table C.1: Table of variables in Code Extracts C.1 and C.2 and their equivalent physical value.

Variable	Quantity
`mu`	Distance from bifurcation point, μ
`w0`	Detector frequency, ω_0
`b`	First Lyapunov coefficient, b
`sr`	Sample rate, f_s
`d`	Damping factor

the solution to $dy/dt = f(y, t)$ at sample n is:

$$k_0 = f(y_n,\ n-2) \tag{C.3a}$$

$$k_1 = f(y_n + k_0\delta,\ n-1) \tag{C.3b}$$

$$k_2 = f(y_n + k_1\delta,\ n-1) \tag{C.3c}$$

$$k_3 = f(y_n + k_2 2\delta,\ n) \tag{C.3d}$$

$$\therefore\ y_{n+1} = y_n + \frac{\delta}{3}\Big(k_0 + 2k_1 + 2k_2 + k_3\Big) \tag{C.3e}$$

Code Extract C.1, provides the core of an RK4Detector. The function `dzdt()` calculates the output of the Hopf bifurcation, as given in Equation (1.34), for a given single z value (denoted `u`), where the forcing is the input, `x`, at sample `t`. `dzdt()` is called by the `process()` function, which takes the input array, `x`, and returns an array of the resulting z values, `z`, according to the procedure in Equation (C.3).

The `process()` function also introduces the damping factor, `d`, to the equation. This — and the sample duration, `1/sr` — scale the output. The effect of this will be discussed in Section 2.2.6.

Table C.1 lists the variables included in this Code Extract with the mathematical values they represent.

The complete implementation, which can be accessed in the project's Git repository, provides RK4Detector as a class, where the values of `mu`, `w0`,

`b`, `sr` and `d` are class members available to the `dzdt()` method; in Code Extract C.1 they are assumed to be global variables.

```
def dzdt(u, t):
    # Implement Hopf equation
    return (mu+1j*w0) * u + b * abs(u*u) * u + x[t]

def process(x):

    # Use the NumPy module for Python to make an array
    # for storing the output, initialised with zeros
    # This array is be the same size as the input, len(x),
    # and it will be storing complex numbers
    z = numpy.zeros(len(x), dtype=complex)

    # Start at index 2 because we need two previous
    # x and z values to calculate the next
    # Initially, the previous z values are zero
    for n in range(2,len(z)):

        # Do the four Runge-Kutta steps
        u0 = z[n-2]
        k0 = dzdt(u0, n-2)

        u1 = u0 + k0 * 1/sr
        k1 = dzdt(u1, n-1)

        u2 = u0 + k1 * 1/sr
        k2 = dzdt(u2, n-1)

        u3 = u0 + k2 * 2/sr
        k3 = dzdt(u3, n)

        # Combine the intermediate values to get z
        z[n] = u0 + (k0+2*k1+2*k2+k3) / (3*sr) * (1-d)

    return z
```

Code Extract C.1: Skeleton RK4Detector

C.2 Central difference approximation

The backward-, forward- and central-difference approximations can easily be derived from the Taylor series expansion of a first order derivative (Smith

1965).

The central-difference approximation

$$y'(x) = \frac{1}{2h}\Big(y(x+h) - y(x-h)\Big) \tag{C.4}$$

can be written in discretized form as:

$$y'(x) = \frac{1}{2\delta}\Big(y[n+1] - y[n-1]\Big) \tag{C.5}$$

where δ is the sample time.

Code Extract C.2 provides an implementation of Equation C.5, akin to Code Extract C.1. The variables used are those listed in Table C.1 and the function `dzdt()` is the same in both Code Extracts.

```
def dzdt(u, t):
    # Implement Hopf equation
    return (mu+1j*w0) * u + b * abs(u*u) * u + x[t]

def process(x):

    # Make empty array where output will be stored
    z = numpy.zeros(len(x), dtype=complex)

    # Again, we start at index 2 because we need 2
    # previous samples to calculate the next.
    for n in range(2,len(z)):

        # Get value from Hopf equation
        dz = dzdt(z[n-1], n-1)
        # Put into central difference approximation
        z[n] = (dz * 2/sr + z[n-2]) * (1-d)

    return z
```

Code Extract C.2: Skeleton CDDetector

Appendix D

Multiple simultaneous frequencies

As discussed in Section 1.5.1, residue pitch, combination tones and difference tones are frequencies that are perceived when multiple tones are presented simultaneously, despite being absent from the stimulus.

When a series of harmonics are presented without the fundamental, this missing frequency is often heard by a listener. This is known as a residue pitch. To test whether the DetectorBank is susceptible to this phenomenon, a series of tones were generated and overlaid, starting at 500 Hz and increasing in steps of 250 Hz up to a maximum frequency of 2 kHz. A DetectorBank was constructed with detectors at each of these frequencies, as well as the missing fundamental frequency, 250 Hz.

Figure D.1 shows the responses. The detectors tuned to 500–2000 Hz respond; the detector tuned to 250 Hz does not. Therefore, residue pitches are not detected by the DetectorBank.

When two tones are presented at frequencies f_1 and f_2, combination tones can occur at frequencies given by $f_1 - k(f_2 - f_1)$, where k is an integer. A difference tone, at $f_1 - f_2$, may also be heard.

This was tested by generating tones at 1 kHz and 1.5 kHz. Detectors

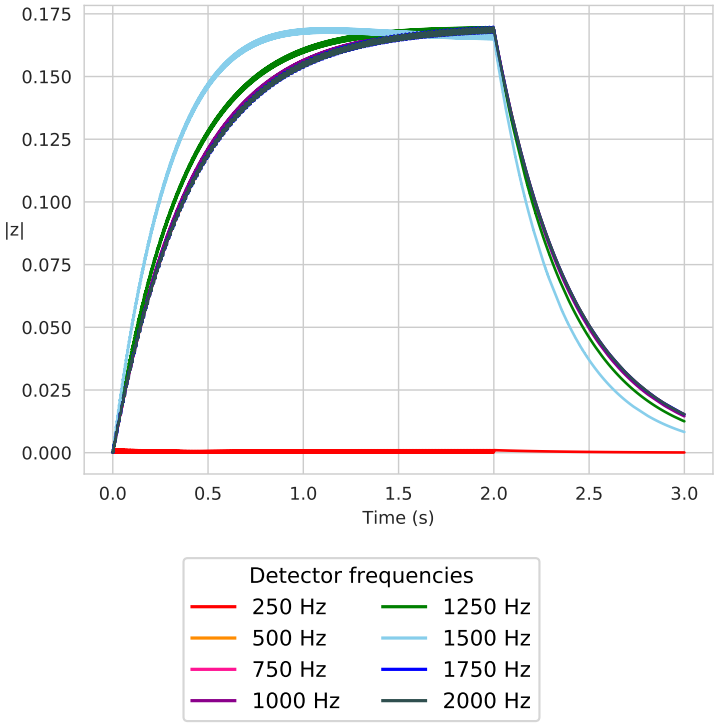


Figure D.1: Responses to tones generated at 500–2000 Hz and presented simultaneously. If the DetectorBank detected the residue pitch, we would expect the detector at 250 Hz to react along with the others. It does not.

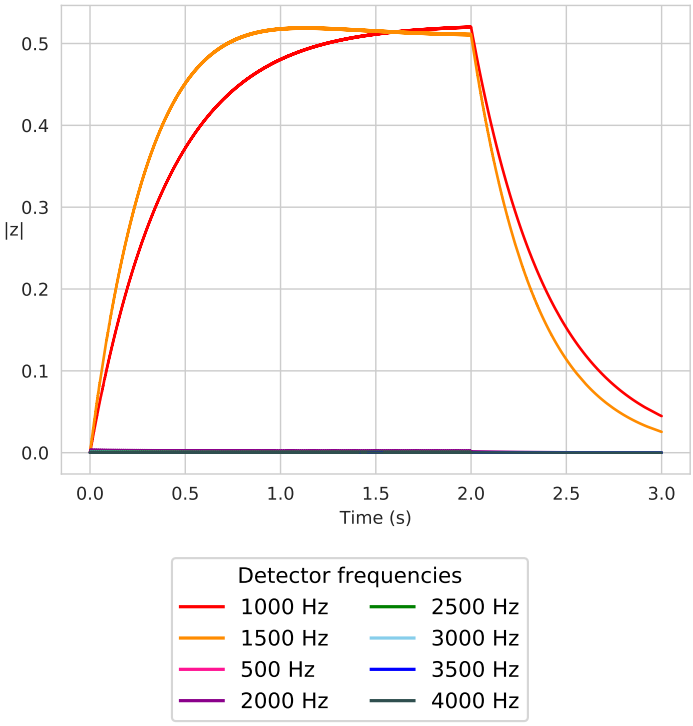


Figure D.2: Responses to tones generated at 1 kHz and 1.5 kHz, with detectors tuned to a range of frequencies found by combining these values. Only the detectors at 1 kHz and 1.5 kHz respond.

were tuned to these frequencies, along with the difference frequency 500 Hz and the combination frequencies for $k = [1 \,..\, 5]$. The results of this are plotted in Figure D.2. The detectors tuned to the frequencies present in the input respond; the others do not.

These tests were carried out using minimum bandwidth detectors; the same results were seen when the bandwidth was increased.

It is clear from these tests that the nonlinearity of the system does not give rise to spurious results at frequencies not present in the stimulus.

Appendix E

Frequency shifting

As stated in Section 2.2.3, frequency-shifted versions of the input signal are generated by creating a double sideband signal and subtracting a version of the signal which has been phase shifted at all frequencies.

The Hilbert transform shifts a signal by $\pi/2$ at all frequencies; therefore it is therefore a useful component in signal processing applications (Van Trees 2001).

This chapter provides a summary of the Hilbert transform theory (Section E.1), derivation (Section E.2), and application to frequency shifting (Section E.3).

E.1 Hilbert transform theory

The spectral characteristics of a real signal are well known: the real part of the spectrum is symmetrical around zero; the imaginary part is anti-symmetrical. For complex signals, no such symmetry conditions exist. Complex signals which have no negative part are known as *analytic signals*.

The Hilbert transform is a fundamental feature of analytic signals, as it provides the relationship between the real and imaginary parts of the signal: the imaginary part is the Hilbert transform of the real part.

E.2 Hilbert transform derivation

E.2.1 Hilbert transform of a real variable

Following the procedures of Debnath (1995), we can say that the Hilbert transform of a function $f(t)$ is

$$f_H(x) = \frac{1}{\pi} \oint_{-\infty}^{\infty} \frac{f(t)}{t-x} \, \mathrm{d}t \tag{E.1}$$

where x is real.

By introducing the function

$$g(x) = \sqrt{\frac{2}{\pi}} \left(-\frac{1}{x} \right)$$

equation (E.1) can be rewritten as

$$f_H(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) g(x-t) \, \mathrm{d}t \tag{E.2}$$

The convolution theorem tells us that convolution in the time domain is equivalent to multiplication in frequency, so applying the Fourier transform to equation (E.2) yields

$$F_H(\omega) = F(\omega) G(\omega) \tag{E.3}$$

where $G(\omega) = j \operatorname{sgn}(\omega)$.

E.2.2 Hilbert transform of a complex variable

Section E.2.1 introduced the Hilbert transform of a real variable x. Derivation of the Hilbert transform in the complex plane follows many of the same

steps, starting from a complex function f_0 of a complex variable $z = x + jy$

$$f_0(z) = \frac{1}{\pi} \oint_{-\infty}^{\infty} \frac{f(t)}{t - z} \, \mathrm{d}t, \quad y > 0 \tag{E.4}$$

This can be expressed as a convolution

$$f_0(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) g(z - t) \, \mathrm{d}t \tag{E.5}$$

where

$$g(z) = \sqrt{\frac{2}{\pi}} \left(-\frac{1}{z} \right)$$

Again, the convolution theorem tells us

$$F_0(\omega) = F(\omega) G(\omega) \tag{E.6}$$

To find $G(\omega)$ we must take the Fourier transform of $g(z - t)$

$$G(\omega) = \sqrt{\frac{2}{\pi}} \int_{-\infty}^{\infty} -\frac{1}{x + jy} e^{-j\omega x} \, \mathrm{d}x \tag{E.7}$$

Given $\mathcal{F}(t - t_0) = F(\omega) e^{-j\omega t_0}$ and $\mathcal{F}(1/\pi t) = -j \operatorname{sgn}(\omega)$, we can say

$$G(\omega) = j \operatorname{sgn}(\omega) e^{-\omega y} \tag{E.8}$$

An analytic signal has no negative part, so the desired frequency response must be

$$H(\omega) = \begin{cases} 0 & \omega < 0 \\ 1 & \omega > 0 \end{cases} \tag{E.9}$$

or equivalently

$$H(\omega) = \frac{1 + \operatorname{sgn}(\omega)}{2} \tag{E.10}$$

Substituting (E.8) and (E.10) into equation (E.6) gives the Fourier trans-

form of (E.4)

$$F_0(\omega) = 2je^{-\omega y}H(\omega)F(\omega) \tag{E.11}$$

E.3 Frequency shifter implementation

The Hilbert transform can be thought of as a filter with the transfer function

$$H(j\omega) = \begin{cases} -j, & \omega > 0 \\ 0, & \omega = 0 \\ j, & \omega < 0 \end{cases} \tag{E.12}$$

Three methods of implementing the Hilbert transform will be discussed here: Fast Fourier transform (FFT), finite impulse response filter (FIR) and infinite impulse response filter (IIR).

E.3.1 Fast Fourier transform

As mentioned in Section E.1, an analytic signal is a complex signal with no negative part. For a given input signal, the analytic signal can be generated by taking the Fourier transform, removing the negative frequencies, then taking the inverse Fourier transform. The real part of the resulting (analytic signal) is the original input signal; the imaginary part is its Hilbert transform.

For example, using the NumPy module in Python, the Hilbert transform of a signal can be found thus:

```
1 h = numpy.fft.fft(signal)
2 h[1:len(signal)//2 + 1] *= 2
3 h[len(signal)//2 + 1:] = 0
4 analytic = numpy.fft.ifft(h)
5 hilbert = analytic.imag
```

Code Extract E.1: Hilbert transform via FFT

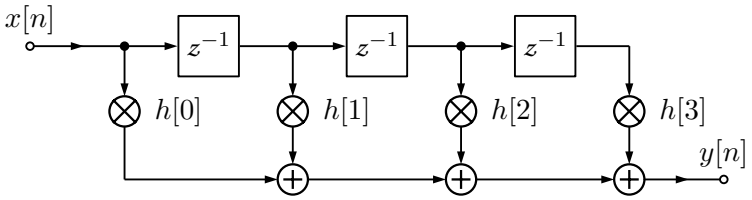


Figure E.1: Four tap FIR filter

Note that the positive frequencies above DC are scaled up by a factor of two, to preserve the total energy in the signal.

Due to the problems associated with entropic uncertainty, discussed in Section 1.4.4 of this thesis, FFT-based implementations are not suitable for realtime processing or other applications which require short buffers of samples. FIR or IIR solutions should be used instead.

E.3.2 FIR filters

As seen in Section E.2, equation (E.1) can be rewritten as

$$f_H(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t)g(x-t)\ \mathrm{d}t \tag{E.13}$$

This is simply a convolution of $f(t)$ and $g(t)$, so can be implemented using an FIR filter, as shown in Figure E.1, with input signal $f(t)$ and convolution kernel which follows the shape of $g(t)$. Setting every other value is set to zero, results in a Type III filter, rather than Type IV (Romero & Dolecek 2012). Figure E.2 shows the kernel generated for an FIR of length 19. A Blackman window has been applied to the kernel.

Although FIR Hilbert transformers are guaranteed to have linear phase and to be stable, this implementation is not perfect: the convolution kernel must be symmetrical around $t = 0$, so there will be a time shift of half the kernel length (samples). Given this, it is desirable to use as short a kernel as possible. Empirical investigation suggests very short kernels give good results. For example, Figure E.3 shows the output of single sideband

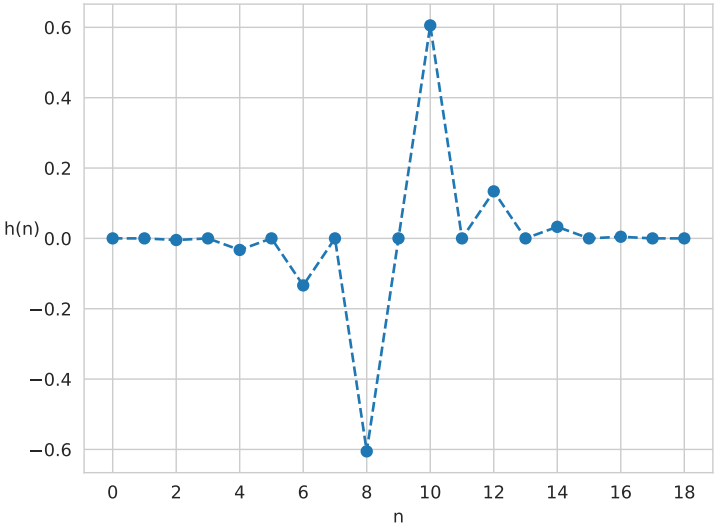
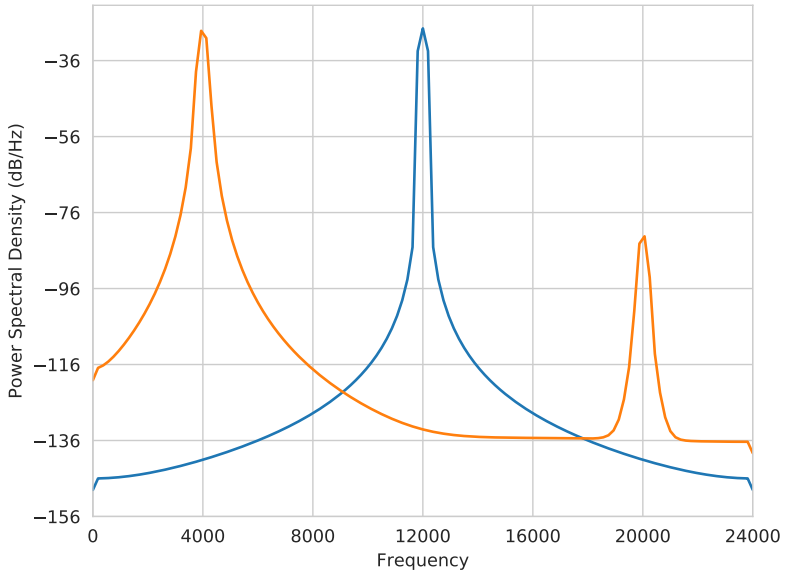


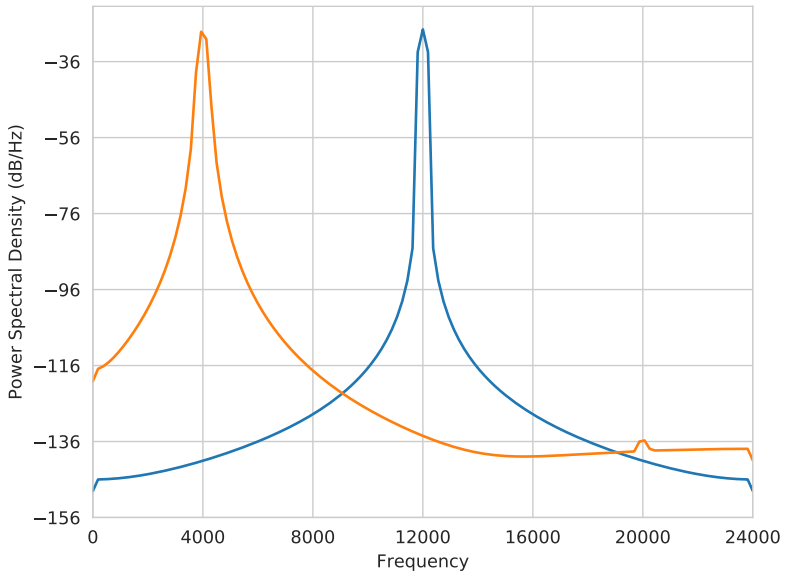
Figure E.2: FIR kernel

modulation, implemented using FIRs. The sample rate used here is 48 kHz and results are shown for (a) 13 tap, and (b) 15 tap FIRs, which correspond to time shifts of 0.15 ms and 0.17 ms respectively. It can be seen that when the kernel length is 13, the spike at 20 kHz (an artifact of sampling, which has also been modulated down) reaches a maximum of 50 dB below the frequencies of interest; raising the kernel length by only two taps eliminates the spike.

Code extract E.2 provides an implementation of the Hilbert transform through FIRs.



(a) 13 tap FIR



(b) 15 tap FIR

Figure E.3: Power spectral density curves showing a signal generated at 12 kHz (blue) and modulated down to 4 kHz (orange), where the FIR Hilbert transformer is used in the frequency shifter.

```python
from math import cos, pi
import numpy as np

def window(n, N):
    """ Return nth from a Blackman window of size N """
    a0 = 0.42
    a1 = 0.5
    a2 = 0.08
    return a0 - a1*cos(2*pi*n/(N-1)) + a2*cos(4*pi*n/(N-1))


def makeKernel(FIRlength):
    """ Return windowed kernel """
    N = 2*((FIRlength+1)//4)
    kernel = [-(cos(n*pi)-1)/(n*pi) * window(m, N)
              for m, n in enumerate(range(-N+1, N, 2))]
    return kernel


def hilbert(x, FIRlength):
    """ Return Hilbert transform of signal x """

    # empty array to be filled
    h = np.zeros(len(x))

    kernel = makeKernel(FIRlength)

    halfklen = FIRlength//2
    # Offset kernel one sample if the 0s aren't in the
    # right place
    koffset = 1 if halfklen == len(kernel) else 0

    # shift by halfken to start from 0 time
    for n in range(halfklen, len(x)+halfklen-1):

        # kmin can't be -ve
        kmin = max(n-(FIRlength-1), 0) + koffset
        # kmax can't be longer than signal
        kmax = min(n, len(x)-1) + koffset

        # sum signal and kernel at current sample number
        for k in range(kmin, kmax-1, 2):
            h[n-halfklen] += x[k] * kernel[(n-k)//2]

    return h
```

Code Extract E.2: Python implementation of a Hilbert transformer via an FIR filter

E.3.3 Infinite Impulse Response filters

IIR filters can also be used to obtain a Hilbert transform Ansari (1987). However, IIR Hilbert transformers perform a phase approximation: exact linear phase is not guaranteed. The advantage of IIR over FIR filters is cost: IIRs only require two delay steps and two multipliers to implement in hardware.

E.3.4 FFT vs FIR Hilbert transformers

The Hilbert transform was used to shift input signals, as discussed in Section 2.2.3. In most circumstances, the FFT and FIR implementations do not noticeable differ in their outputs. However, when testing shifting of very low frequencies, the effects of windowing and the FIR length became apparent.

Figure E.4 shows the responses of five 1 Hz-spaced detectors to a 10 Hz sine wave, generated at 48 kHz and lasting for 1 second. The responses show small oscillations.

If the signal is modulated up to 110 Hz using the fast Fourier transform (Figure E.5a), the responses are smooth, although the shape of the neighbouring responses is somewhat distorted. This is very similar to those obtained when a tone is generated directly at that frequency, which can be seen in many of the figures shown in Chapter 2.

When the same signal is modulated using an FIR filter, as shown in Figure E.5b, the shape of the responses remains the same: the oscillations are still present.

These oscillations are at twice the input frequency, in this case 20 Hz. 1200 samples would be required to represent 20 Hz at sample rate of 48 kHz. The FIR length used here is 19 samples. This means the FIR has no effect on these oscillations.

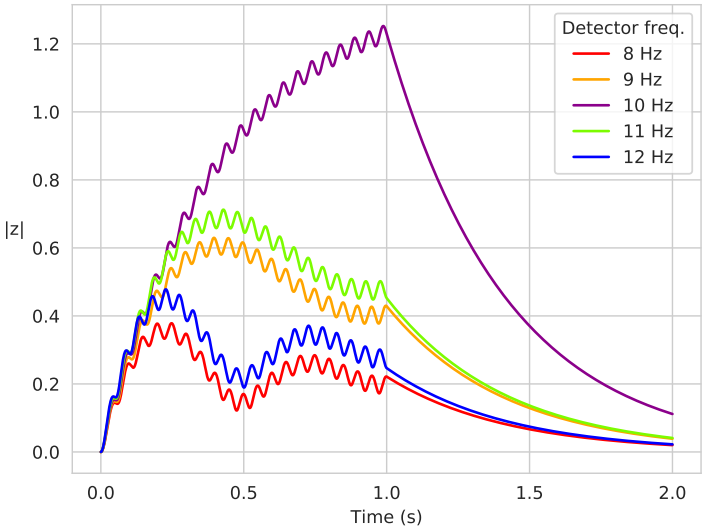
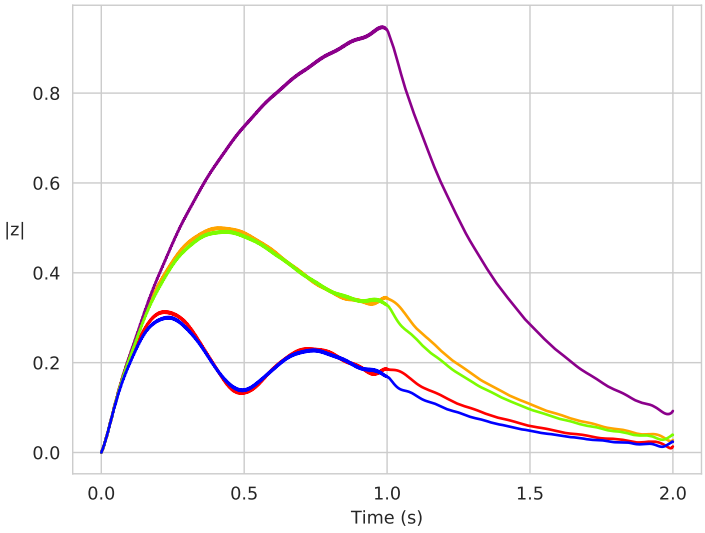
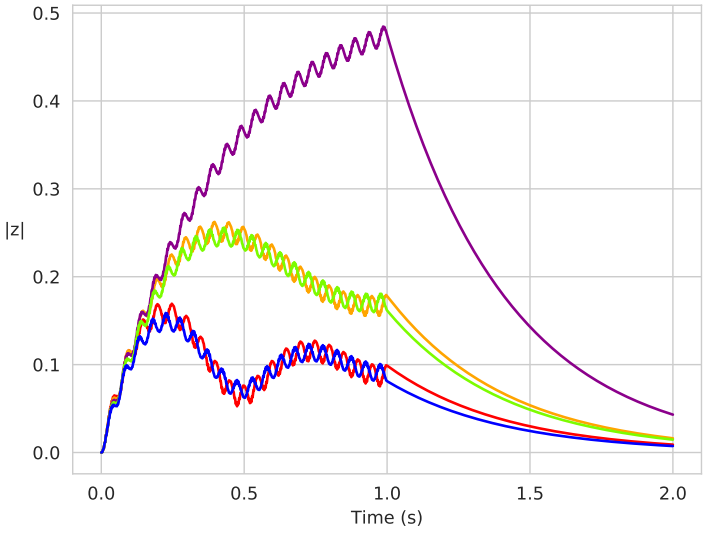


Figure E.4: Unmodulated responses of five 1 Hz-spaced detectors to a 10 Hz sine wave.



(a) FFT



(b) FIR

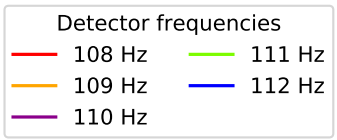


Figure E.5: Responses generated when the signal used in Figure E.4 is shifted up by 100 Hz, implemented with (a) FFT, (b) FIR filter.

Glossary of scientific terms

AbstractDetector Detector base class, derived classes of which provide different methods of solving the Hopf bifurcation. 52

aggregation Term from UML, describing a relationship where a child class can exist independently of the parent, in contrast to a composition. 52

band A *band* in the NoteDetector refers to a set of detectors tuned to frequencies that would be perceived as the same pitch. As described in Section 3.1.2, bands were based on the phenomenon of critical bands in the auditory system, and are sometimes referred to as such in this thesis. 108, 142, 204, 246

bifurcation Changes in a system's topology that occur when a parameter passes through a critical value. 43

cochlea Part of the inner ear in which frequency and time perception occurs. 39, 91

composition Term from UML, describing a relationship where a child class cannot exist without the parent. 245

critical band In audiology, the basilar membrane is often modelled as a series of overlapping bandpass filters. A critical band refers to one of these rectangular filters. This concept was used in the OnsetDetector

development; when the term 'critical band' appears in this thesis, it refers to these OnsetDetector bands, unless otherwise stated. 41, 105, 119, 133, 149, 190, 212, 245

damping factor Damping applied to a detector. 55, 225

detector Object which implements a Hopf bifurcation using either the Runge-Kutta or central difference methods. 51, 97, 142, 245–247

DetectorBank Object for managing multiple detectors, which all take the same input and are tuned to different frequencies. Chapter 2 presents a detailed investigation into the characteristics of the DetectorBank under different conditions. 18, 51, 97, 149, 211, 246, 247

DetectorCache Object which stores, at most, the N most recent samples of DetectorBank output in order to keep memory requirement to a minimum. 53, 98, 149

embarassingly parallel A term for a problem that can easily be split into parallel tasks. 52, 246

frequency Rate of repetition of a pattern in a signal. 8, 247

Hopf bifurcation Term that describes emergence of periodic solutions as a parameter passes a critical point. Section 1.5.2 presents the Hopf bifurcation in detail. 42, 51, 211, 245, 246

multithreading Distributing tasks to be run on multiple CPUs on a processor simultaneosly. This can hugely speed up execution time, particularly if the program is embarassingly parallel. 52, 144

NoteDetector Object for using both an OnsetDetector and a PitchTracker to identify notes in musical audio. A UML diagram showing the structure of a NoteDetector and its components is given in Figure 3.19. 97

Nyquist rate The Nyquist theorem states that to represent a signal of bandwidth B Hz, it must be sampled at a frequency of at least $2B$ Hz. This minimum sampling frequency is known as the Nyquist rate. 21, 31, 95

OnsetDetector Object for finding the onset times of note and intra-note events in musical audio, the design and implementation of which is detailed in Chatper 3. 18, 41, 145, 212, 245, 246

period The duration of one repetition of pattern a signal. For a given frequency, f, the period is $T = 1/f$. 12, 43, 60, 250

search normalisation Method of frequency normalisation employed in the DetectorBank which searches for the input frequency for a given detector's response is optimal and adjusts the detector's characteristic frequency accordingly. 54, 70

support The support of a signal refers to the places at which it is non-zero. 37, 38

topology The study of geometrical properties which are unchanged after undergoing deformation or distortion. 43, 245

UML The Unified Modelling Language, UML, is a graphical system for describing the structure, interactions and uses of objects in software. 98, 245, 246

uncertainty principle Discussed briefly in Section 1.4.4, the uncertainty principle states that it is impossible for a pair of properties of a function to be simultaneously sharply located. In signal processing, the pair of properties are time and frequency: increasing the time resolution, Δt, necessarily reduces the frequency resolution, Δf, and vice versa, in accordance with the relation $\Delta t \Delta f \geq 1/2$. 51, 95, 211

Glossary of musical terms

arco Term for playing a string instrument with a bow. 145, 151

attack Initial period in the waveform of a note, in which the amplitude is increasing rapidly to its maximum. 9, 21, 181

bowed The technique of sounding a note by pulling a bow across the instrument. Most commonly used on string instruments, but vibraphones and saws can also be bowed. 4, 182, 249, 250

EDO This stands for equal divisions of the ocatve, and refers to a scale in which the fundamental frequency of each note increases by a constant factor. In 12 EDO music, this factor is $2^{1/12}$. 23, 32

glissando Musical technique of sliding from one note to another. 8, 145

intonation A musician's pitch accuracy. 5, 190

melisma Feature of vocal music where one syllable is sung over several notes. 13

monophony Monophonic music features no more than one note sounding at a time and is the opposite of polyphonic. 124, 250

octave Relation between notes that corresponds to the doubling of the fundamental frequency. Note names, like A4, refer to both the pitch

class and the octave number, where the octave numbers start from zero. Octave number wrap around at each C. Middle C is C4. 148

ornament Musical technique of 'decorating' notes in a melody, for example trills, mordents and turns. 8

pitch The perception of a note's position in a scale. 8, 249

pitch class Note name from A, B, C, D, E, F, G. 7, 8, 104, 249

pitched A pitched sound contains periodic elements, so has a perceived pitch to which an equivalent note name can be assigned. 5

pizzicato Plucked, as opposed to bowed, strings. 145, 151

polyphony Polyphonic music features multiple note sounding simultaneously. See also monophonic music. 210, 249

roll Percussion technique of repeating a note quickly to produce a sustained sound. 145

transient The portion of sound at the beginning of a note, when the signal is changing rapidly and pitch has not yet been established. 5, 9, 21

unpitched An unpitched sound does not contain periodic elements, so a pitch cannot be assigned. However, unpitched sounds can still perceived as having a relative pitch 'height', i.e. high or low (for example, the toms in a drum kit). 5

vibrato Musical technique of fluctuating the pitch of a sustained note. 8, 145, 151

Discography

Berio, Luciano (2012), *Good Night*, with Wim Van Hasselt, trumpet, on *On The Road*, Channel, CCSSA31811, CD/digital download.

Dixon, Michael H. (2011), *A Hundred Valleys*, Stephen Altoft, trumpet, recorded April 2010, on *The Yasser Collection*, Microtonal Projects, CD.

Dove, Jonathan (2014), *Ariel: No. 2. I boarded the King's ship*, Claire Booth, soprano, on *All You Who Sleep Tonight: Song Cycles*, Naxos, 8.573080, CD/digital download.

Grebtschenko, Alexander (2011), *Bye Bye*, Stephen Altoft, trumpet, recorded July 2011, on *The Yasser Collection*, Microtonal Projects, CD.

Hair, Graham (2013), *Songs from the Turkish*, Frances Morrison-Allen, soprano, private recordings from session of *Music from 3 Continents*, Ravello, RR7877.

Knussen, Oliver (2011), *Four Late Poems and an Epigram of Rainer Maria Rilke*, with Lisa Saffer, soprano, on *Concerto for Orchestra/Océan de terre*, Virgin Classics, 0963542, CD/digital download.

Ligeti, György (2008), *The Big Turtle Fanfare from the South China Sea*, Peter Masseurs, trumpet, on *The Ligeti Project*, Warner Classics & Jazz, 2564 69673-5, CD.

Parsons, Michael (2011), *Melody in 19-division tuning*, Stephen Altoft, trumpet, recorded April 2008, on *The Yasser Collection*, Microtonal Projects, CD.

Reich, Gottfried (1996), *Fanfare Abblasen*, Chip Davis, trumpet, on *Holiday Musik*, American Gramaphone, AG 296-2, CD/digital download.

Bibliography

Addison, P. S. (2002), *The Illustrated Wavelet Transform Handbook*, Taylor & Francis, New York/London. 34, 36

Ansari, R. (1987), 'IIR discrete-time Hilbert transformers', *IEEE Transactions on Acoustics, Speech, and Signal Processing* **35**(8), 1116–1119. 241

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. & Sandler, M. B. (2005), 'A tutorial on onset detection in music signals', *Speech and Audio Processing, IEEE Transactions on* **13**(5), 1035–1047. 21, 22

Bello, J. P. & Sandler, M. (2003), Phase-based note onset detection for music signals, *in* 'Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on', Vol. 5, IEEE, pp. V–441. 22

Benbouzid, M. E. H. (2000), 'A review of induction motors signature analysis as a medium for faults detection', *IEEE Transactions on Industrial Electronics* **47**(5), 984–993. 2

Böck, S. & Widmer, G. (2013*a*), Local group delay based vibrato and tremolo suppression for onset detection., *in* 'ISMIR', Citeseer, pp. 361–366. 22, 28

Böck, S. & Widmer, G. (2013*b*), Maximum filter vibrato suppression for onset detection, *in* 'Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13), Maynooth, Ireland'. 22, 28

Brannan, D. A., Esplen, M. F. & Gray, J. J. (2012), *Geometry*, 2nd edn, Cambridge University Press. 60

Brossier, P. M. (2006), Automatic annotation of musical audio for interactive applications, PhD thesis, Queen Mary University of London. 21, 22, 28

Brown, J. C. (1991), 'Calculation of a constant-Q spectral transform', *The Journal of the Acoustical Society of America* **89**(1), 425–434. 32

Brown, J. C. (1992), 'Musical fundamental frequency tracking using a pattern recognition method', *The Journal of the Acoustical Society of America* **92**(3), 1394–1402. 32

Brown, J. C. & Puckette, M. S. (1992), 'An efficient algorithm for the calculation of a constant Q transform', *The Journal of the Acoustical Society of America* **92**(5), 2698–2701. 33

Cannam, C., Landone, C. & Sandler, M. (2010), Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files, *in* 'Proceedings of the international conference on Multimedia', ACM, pp. 1467–1468. 22

Christodoulou, N. S. (2008), 'Discrete hopf bifurcation for runge–kutta methods', *Applied mathematics and computation* **206**(1), 346–356. 53

Clarke, E. & Cook, N., eds (2004), *Empirical Musicology: Aims, Methods, Prospects*, Oxford University Press, Oxford/New York. 2

Cohen, L. (1995), *Time-frequency analysis*, Prentice Hall, Upper Saddle River, New Jersey. 37

Debnath, L. (1995), *Integral Transforms and Their Applications*, CRC press, Boca Raton, Florida. 234

Dennis, J., Tran, H. D. & Li, H. (2015), 'Generalized hough transform for speech pattern classification', *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **23**(11), 1963–1972. 113

Devaney, J., Mandel, M. I., Ellis, D. P. & Fujinaga, I. (2011), 'Automatically extracting performance data from recordings of trained singers.', *Psychomusicology: Music, Mind and Brain* **21**(1-2), 108. 210

Dey, N., Dutta, S., Dey, G., Chakraborty, S., Ray, R. & Roy, P. (2014), Adaptive thresholding: A comparative study. 204

Duxbury, C., Bello, J. P., Davies, M., Sandler, M. et al. (2003), Complex domain onset detection for musical signals, *in* 'Proc. Digital Audio Effects Workshop (DAFx)', pp. 6–9. 22

Edison, T. A. (1878), 'Improvement in phonograph or speaking machine'. US200521A. 6

Eguíluz, V. M., Ospeck, M., Choe, Y., Hudspeth, A. & Magnasco, M. O. (2000), 'Essential nonlinearities in hearing', *Physical Review Letters* **84**(22), 5232. 40

Feng, Z., Liang, M. & Chu, F. (2013), 'Recent advances in time-frequency analysis methods for machinery fault diagnosis: A review with application examples', *Mechanical Systems and Signal Processing* **38**(1), 165–205. 2

Fontana, F., Järveläinen, H., Papetti, S., Avanzini, F., Klauer, G., Malavolta, L., di Musica, C. & Pollini, C. (2015), Rendering and subjective evaluation of real vs. synthetic vibrotactile cues on a digital piano keyboard, *in* 'Proc. of the Sound and Music Computing Conference'. 14

Foote, J. & Uchihashi, S. (2001), The beat spectrum: A new approach to rhythm analysis, *in* 'IEEE International Conference on Multimedia and Expo, 2001. ICME 2001', IEEE, p. 224. 22

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M. & Gagné, C. (2012), 'DEAP: Evolutionary algorithms made easy', *The Journal of Machine Learning Research* **13**(1), 2171–2175. 76

Fowler, M. (2004), *UML Distilled*, 3rd edn, Addison-Wesley, Boston, Massachusetts. 52

Fritts, L. (1997), 'The University of Iowa Electronic Music Studios musical instrument samples', http://theremin.music.uiowa.edu/MIS.html. Accessed 22 November 2016, 14 June 2018. 145

Fuentes, B., Liutkus, A., Badeau, R. & Richard, G. (2012), Probabilistic model for main melody extraction using constant-Q transform, *in* 'Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on', IEEE, pp. 5357–5360. 32

Gabor, D. (1946), 'Theory of communication', *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering* **93**(26), 429–457. 37, 63

Ganseman, J., Scheunders, P. & D'haes, W. (2008), Using XQuery on MusicXML databases for musicological analysis., *in* 'ISMIR', pp. 433–438. 14

Ghahramani, Z. (2001), 'An introduction to hidden Markov models and Bayesian networks', *International Journal of Pattern Recognition and Artificial Intelligence* **15**(01), 9–42. 210

Ghasemloonia, A. & Khadem, S. E. Z. (2011), 'Gear tooth failure detection by the resonance demodulation technique and the instantaneous power spectrum method–a comparative study', *Shock and Vibration* **18**(3), 503–523. 2

Glendinning, P. (1994), *Stability, instability and chaos: an introduction to*

the theory of nonlinear differential equations, Cambridge University Press. 44

Gold, T. (1948), 'Hearing. II. The physical basis of the action of the cochlea', *Proceedings of the Royal Society of London B: Biological Sciences* **135**(881), 492–498. 41

Gomez, F. & Stoop, R. (2014), 'Mammalian pitch sensation shaped by the cochlear fluid', *Nature Physics* **10**(7), 530–536. 38, 39

Good, M. et al. (2001), MusicXML: An internet-friendly format for sheet music, *in* 'XML Conference and Expo', pp. 3–4. 14

Guckenheimer, J. (2008), 'Singular Hopf bifurcation in systems with two slow variables', *SIAM Journal on Applied Dynamical Systems* **7**(4), 1355–1377. 47

Guckenheimer, J. & Holmes, P. (1983), *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, Vol. 42 of *Applied Mathematical Sciences*, Springer, New York. 44

Hanslick, E. (1986), *On the musically beautiful*, Hackett Publishing Company, Indianapolis, Indiana. 6

Helmholtz, H. L. (1885), *On the Sensations of Tone as a Psychological Basis for the Theory of Music*, 2nd edn, Longmans, Green and Co.
URL: *https://openlibrary.org/books/OL2668992M/On_the_sensations_of_tone_as_a_physiological_basis_for_the_theory_of_music* 189

Hill, M. (2013), 'The uncertainty principle for Fourier transforms on the real line', https://math.uchicago.edu/~may/REU2013/REUPapers/Hill.pdf. Accessed 15 April 2020. 37

Hold-Geoffroy, Y., Gagnon, O. & Parizeau, M. (2014), Once you SCOOP, no need to fork, *in* 'Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment', ACM, p. 60. 149

Holighaus, N., Dorfler, M., Velasco, G. A. & Grill, T. (2013), 'A framework for invertible, real-time constant-Q transforms', *IEEE Transactions on Audio, Speech, and Language Processing* **21**(4), 775–785. 33

Hudspeth, A. (2005), 'How the ear's works work: Mechanoelectrical transduction and amplification by hair cells', *Comptes rendus biologies* **328**(2), 155–162. 42

Hudspeth, A. (2008), 'Making an effort to listen: Mechanical amplification in the ear', *Neuron* **59**(4), 530–545. 42

Hudspeth, A., Jülicher, F. & Martin, P. (2010), 'A critique of the critical cochlea: Hopf—a bifurcation—is better than none', *Journal of Neurophysiology* **104**(3), 1219–1229. 42

Huron, D. B. (2006), *Sweet Anticipation: Music and the Psychology of Expectation*, MIT press, Cambridge, Massachusetts. 6, 211

James, G. (2011), *Advanced Modern Engineering Mathemathics*, 4th edn, Pearson, Harlow, UK. 223

Kemp, D. T. (1978), 'Stimulated acoustic emissions from within the human auditory system', *The Journal of the Acoustical Society of America* **64**(5), 1386–1391. 42

Kemp, D. T. (1979), 'Evidence of mechanical nonlinearity and frequency selective wave amplification in the cochlea', *Archives of Oto-rhino-laryngology* **224**(1–2), 37–45. 42

Kern, A. & Stoop, R. (2003), 'Essential role of couplings between hearing nonlinearities', *Physical review letters* **91**(12), 128101–128101. 42

Kuznetsov, Y. A. (2004), *Elements of Applied Bifurcation Theory*, Vol. 112 of *Applied Mathematical Sciences*, 3rd edn, Springer, New York. 44, 49

Li, B., Smith, J. B. & Fujinaga, I. (2009), Optical audio reconstruction for stereo phonograph records using white light interferometry., *in* 'ISMIR', Citeseer, pp. 627–632. 3

London, J. (2012), *Hearing in time: Psychological aspects of musical meter*, Oxford University Press, Oxford. 2

Majka, M., Martinson, K., Kamisiński, T. & Zieliński, P. (2018), 'Duration discrimination of ultrashort acoustic pulses', *Advances in Acoustics* . 43, 63

Majka, M., Sobieszczyk, P., Gębarowski, R. & Zieliński, P. (2015), 'Hearing overcomes uncertainty relation and measure duration of ultrashort pulses', *Europhysics News* **46**(1), 27–31. 43, 63

Milligan, K. & Bailey, N. (2015), A review of software for note onset detection, *in* 'ANIMUSIC 2015'. 21

Milligan, K., Bailey, N. & Hair, G. (2016), 'A new approach to onset detection: towards an empirical grounding of theoretical and speculative ideologies of musical performance', *Scottish Music Review* **4**(1). 6

Milligan, K., Fiedler, J., & Bailey, N. (2018), 'What is a note, then?'. ANIMUSIC 2018. 8

Moore, B. C. (2012), *An Introduction to the Psychology of Hearing*, 6th edn, Emerald, Bingley, UK. 7, 22, 40

Morehead, P. D. & MacNeil, A. (1992), *Bloomsbury dictionary of music*, Bloomsbury, London. 8

Nanavati, S. P. & Panigrahi, P. K. (2004), 'Wavelet transform: A new mathematical microscope', *Resonance* **9**, 50 – 64. 33

Ning, C.-z. & Haken, H. (1990), 'Detuned lasers and the complex Lorenz equations: subcritical and supercritical Hopf bifurcations', *Physical Review A* **41**(7), 3826–3837. 44

Nixon, M. S. & Aguado, A. S. (2012), *Feature extraction & image processing for computer vision*, 3rd edn, Academic Press. 112, 204

Olson, E. S., Duifhuis, H. & Steele, C. R. (2012), 'Von Békésy and cochlear mechanics', *Hearing Research* **293**(1), 31–43. 41

Percival, G. (2013), Physical Modelling meets Machine Learning: Performing Music with a Virtual String Ensemble, PhD thesis, University of Glasgow. 4, 190

Pertusa, A., Klapuri, A. & Iñesta, J. M. (2005), *Recognition of note onsets in digital music using semitone bands*, Springer, pp. 869–879. 22

Pickles, J. O. (2012), *An Introduction to the Physiology of Hearing*, 4th edn, Emerald Group Publishing Limited, Bingley, UK. 39, 40, 41, 104, 130

Plack, C. (2012), 'Hearing pitch — right place, wrong time?', *Psychologist* **25**(12), 892–894. 41

Proakis, J. G. & Manolakis, D. G. (2007), *Digital Signal Processing: Principles, Algorithms and Applications*, Pearson Prentice Hall, New Jersey. 31

Rabiner, L. R. & Gold, B. (1975), *Theory and Application of Digital Signal Processing*, Prentice-Hall, Inc, Englewood Cliffs, NJ. 68

Rabiner, L. R., Schafer, R. W. & Rader, C. M. (1969*a*), 'The chirp z-transform algorithm', *IEEE transactions on audio and electroacoustics* **17**(2), 86–92. 31

Rabiner, L. R., Schafer, R. W. & Rader, C. M. (1969*b*), 'The chirp z-transform algorithm and its application', *Bell System Technical Journal* **48**(5), 1249–1292. 31

Räisänen, O. (2017), 'Gramophone audio from photograph, revisited', https://www.windytan.com/2017/07/gramophone-audio-from-photograph.html. Accessed 28 March 2018. 3

Reetz, H. & Jongman, A. (2009), *Phonetics: Transcription, production, acoustics, and perception*, Wiley-Blackwell, Chichester, UK. 5

Rhode, W. S. (1971), 'Observations of the vibration of the basilar membrane in squirrel monkeys using the Mössbauer technique', *The Journal of the Acoustical Society of America* **49**(4B), 1218–1231. 41

Rhode, W. S. & Robles, L. (1974), 'Evidence from Mössbauer experiments for nonlinear vibration in the cochlea', *The Journal of the Acoustical Society of America* **55**(3), 588–596. 41

Ricaud, B. & Torrésani, B. (2013), 'A survey of uncertainty principles and some signal processing applications', *Advances in Computational Mathematics* **40**(3), 629–650. 37

Roach, P. (2001), *Phonetics*, Oxford University Press, Oxford. 13

Romero, D. E. T. & Dolecek, G. J. (2012), *Digital FIR Hilbert transformers: Fundamentals and Efficient Design Methods*, Vol. 1, INTECH Open Access Publisher, chapter 19, pp. 445–482. 237

Rosales, R. (2005), 'Hopf bifurcations', http://ocw.mit.edu/courses/mathematics/18-385j-nonlinear-dynamics-and-chaos-fall-2004/lecture-notes/hopfbif.pdf. Accessed 22 February 2016. 44

Ross, A. (2008), *The rest is noise: Listening to the twentieth century*, Fourth Estate, London. 6

Ruggero, M. A. & Rich, N. C. (1991), 'Application of a commercially-manufactured doppler-shift laser velocimeter to the measurement of basilar-membrane vibration', *Hearing research* **51**(2), 215–230. 42

Scheirer, E. & Slaney, M. (1997), Construction and evaluation of a robust multifeature speech/music discriminator, *in* 'Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on', Vol. 2, IEEE, pp. 1331–1334. 22

Smaragdis, P. (2009), 'Relative-pitch tracking of multiple arbitrary sounds', *The Journal of the Acoustical Society of America* **125**(5), 3406–3413. 32

Smith, G. D. (1965), *Numerical Solution of Partial Differential Equations*, Oxford University Press, London/New York. 226

Smoorenburg, G. F. (1972), 'Combination tones and their origin', *The Journal of the Acoustical Society of America* **52**(2B), 615–632. 40

Song, Y., Han, M. & Wei, J. (2005), 'Stability and Hopf bifurcation analysis on a simplified BAM neural network with delays', *Physica D: Nonlinear Phenomena* **200**(3-4), 185–204. 44

Staszewski, W. J., Worden, K. & Tomlinson, G. R. (1997), 'Time–frequency analysis in gearbox fault detection using the Wigner-Ville distribution and pattern recognition', *Mechanical Systems and Signal Processing* **11**(5), 673–692. 2

Stein, J. Y. (2000), *Digital Signal Processing*, Wiley Series in Telecommunications and Signal Processing, John Wiley & Sons, Inc., New York. 19, 30

Stowell, D. & Plumbley, M. (2007), Adaptive whitening for improved real-time audio onset detection, *in* 'Proceedings of the International Computer Music Conference (ICMC07)', Vol. 18. 28

Sundberg, J. (1994), 'Perceptual aspects of singing', *Journal of Voice* **8**(2), 106–122. 29

The Complete MIDI 1.0 Detailed Specification (2014), 3rd edn, The MIDI Manufacturers Association, Los Angeles, CA. 13

The web's signature MIDI collection (2015), http://www.yamahaden.com/midi-files. Accessed 22 January 2019. 14

Van Trees, H. L. (2001), *Detection, estimation, and modulation theory, part I: detection, estimation, and linear modulation theory*, John Wiley & Sons, New York. 68, 233

Velasco, G. A., Holighaus, N., Dörfler, M. & Grill, T. (2011), 'Constructing an invertible constant-Q transform with non-stationary Gabor frames', *Proceedings of DAFX11, Paris* . 33

Wiggins, S. (1990), *Introduction to Applied Nonlinear Dynamical Systems and Chaos*, Vol. 2 of *Texts in Applied Mathematics*, Springer-Verlag, New York. 43

Williams, W. J., Brown, M. L. & Hero, A. O. (1991), Uncertainty, information, and time-frequency distributions, *in* 'Advanced Signal Processing Algorithms, Architectures, and Implementations II', Vol. 1566, International Society for Optics and Photonics, pp. 144–157. 37

Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. (2017), *Data Mining: Practical machine learning tools and techniques*, 4th edn, Morgan Kaufmann, Cambridge, Massachusetts. 23, 105

Wittgenstein, L. (2014), *Tractatus Logico-philosophicus*, Routledge, Oxford. Rev. edn. Routledge & Kegan Paul 1974. 3

Zhang, H., Kang, J., Huang, T., Cong, X., Ma, S. & Huang, H. (2018), 'Hopf

bifurcation, Hopf-Hopf bifurcation, and period-doubling bifurcation in a four-species food web', *Mathematical Problems in Engineering* **2018**. 44

Zhang, Y. & Golubitsky, M. (2011), 'Periodically forced Hopf bifurcation', *SIAM Journal on Applied Dynamical Systems* **10**(4), 1272–1306. 47