UNIVERSITY
of
GLASGOW

Development and Exploration of a Timbre Space Representation of Audio

Craig Andrew Nicol

Submitted for the degree of Doctor of Philosophy

University of Glasgow
Department of Computing Science

September, 2005

Abstract

Sound is an important part of the human experience and provides valuable information about the world around us. Auditory human-computer interfaces do not have the same richness of expression and variety as audio in the world, and it has been said that this is primarily due to a lack of reasonable design tools for audio interfaces.

There are a number of good guidelines for audio design and a strong psychoacoustic understanding of how sounds are interpreted. There are also a number of sound manipulation techniques developed for computer music. This research takes these ideas as the basis for an audio interface design system. A proof-of-concept of this system has been developed in order to explore the design possibilities allowed by the new system.

The core of this novel audio design system is the timbre space. This provides a multi-dimensional representation of a sound. Each sound is represented as a path in the timbre space and this path can be manipulated geometrically. Several timbre spaces are compared to determine which amongst them is the best one for audio interface design. The various transformations available in the timbre space are discussed and the perceptual relevance of two novel transformations are explored by encoding "urgency" as a design parameter.

This research demonstrates that the timbre space is a viable option for audio interface design and provides novel features that are not found in current audio design systems. A number of problems with the approach and some suggested solutions are discussed. The timbre space opens up new possibilities for audio designers to explore combinations of sounds and sound design based on perceptual cues rather than synthesiser parameters.

Dedication

To Dad.

1950-2002

Acknowledgments

First of all, my heartfelt and sincere thanks to Mary for her love, encouragement, understanding and assistance.

Thanks to Stephen and Phil for their supervision, inspiration and motivation.

Thanks to David for his help with LaTeX, and thanks to Lorna for the software to run the experiments in Chapter 6.

Thanks to my family and friends for their support. Thanks to everyone who participated in the perception experiment described in Chapter 6.

The Wave writing component of the code was written by Don Cross[1] and was downloaded from his web-site.

Thanks to Ron Kuper of Cakewalk, who has made available the source to the DXi host and has helped in understanding the technical details of the specification.

The Matlab implementation of the EM-Kalman algorithm, on which the EM version of PCA was based, was developed by A-V.I. Rosti [2].

The Matlab implementation of the CQT algorithm used in this research was written by Judy Brown [3] with an improved inverse by Derry Fitzgerald at the Cork Institute of Technology.

The k-d tree implementation used here is from the ANN library by Mount and Anil [4].

This PhD was funded from an EPSRC studentship.

Declaration

The material presented in this thesis is the result of my own research carried out at the Department of Computing Science at the University of Glasgow working under the supervision of Professor Stephen Brewster and Philip Gray. Some of the results presented in Chapters 4 and 5 have previously been presented at CADUI 2004 [5] and ICAD 2004 [6].

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation

Sound is a fundamental part of the human experience, whether presented as speech or otherwise, our lives would be significantly different without sound. From the sound of birds in the morning to the sound of snoring late at night, our everyday lives are full of audio information about the world around us.

There are many occasions where sound is the easiest or only way to obtain information, where footsteps on a hallway are heard long before your eyes are able to see who is at the door, or where mechanics can hear faults in an engine whilst driving allowing them to diagnose a problem. A mechanic can then suggest a resolution without having to dismantle the engine to see inside.

With such a rich sonic environment outside the computer, it seems almost negligent that the audio interfaces we have become used to, on desktop machines running Microsoft Windows and Linux or on mobile phone and other hand-held devices, are static and uninformative, and are often turned off. There is a lot of good research into better audio interfaces, but this research rarely finds its way onto consumer devices. As computing devices get smaller and visual displays start to disappear, the need for alternative streams of information grows.

Audio design is motivated by a number of factors, including accessibility for users who cannot use their vision, due to their eyes being focussed on some other task such as driving, where the available visual display is small or non-existent, as in a number of new hand-held devices including mobile phones and MP3 players, or where the user themselves has vision problems.

The existing literature does offer some hope in this area, particularly in Gaver's development of Auditory Icons [7] in SonicFinder for the Apple Macintosh, which are designed to sound like objects in the real world, and in Blattner *et al.*'s work on Earcons [8], which focuses on sound design using musical principles. These two papers demonstrate auditory interfaces that can reflect information about the underlying computer system through structured and dynamic audio. This means that the interfaces can respond to changes in the underlying information in predictable ways, allowing users to make the same kind of inference from the sound as the mechanic listening to the car engine.

The creation of audio interfaces with this level of usability requires audio design tools and methods suited for interface design rather than musical design. Where musical design is artistic and individual, interface design should be scientific and based on many individuals so that the interface is usable by a range of people. Interface design needs to map the underlying information to audio in a clear and consistent manner to enable users to infer patterns in the information from patterns in the sound. For example, if a metal sound indicates an application in one folder, a metal sound should indicate an application in all folders. Where the sounds can be ordered according to a perceptual axis, for example, "urgent" sounds and "non-urgent" sounds, the more urgent sounds should represent more urgent conditions to reduce confusion in the users.

Some guidelines exist for the design of both Auditory Icons and Earcons, but there is no easy or clear way to design sounds that can be used to develop an Auditory Icon or Earcon interface. This problem was noted by Gaver over 10 years ago [9] and has been mentioned since, by Mynatt *et al.* [10] and by Coleman *et al.* [11], suggesting the design problem remains largely unsolved. The research in this thesis seeks to examine ways to address the need for such a design tool by building a prototype that can be used to explore audio design in a way previously unavailable to audio interface designers.

Currently auditory interfaces are generally designed in isolation from other audio interfaces that may be present. In particular, there is no unifying tool that allows designers to work with both Auditory Icons and Earcons, in order to determine how they might interact and which may be most suitable for the purpose in hand. A unified tool could use Auditory Icons as instruments for Earcons, extending the design range to encompass both systems easily. The lack of such a tool slows progress in creating the interfaces and in understanding how to design sounds in a Human-Computer Interaction (HCI) context.

A better system for audio design and development should take account of research into human perception when developing the interface. Such an interface should enable sounds to be described in terms of the objects that produce them, as with Auditory Icons, or in terms of perceptual response, e.g. to indicate urgency or speed. Current audio design tends to focus on signal or musical properties of a sound. Such a system could help designers find useful sounds for their interfaces and from this a wider set of design guidelines for sonic interactions can be realized and better interfaces can be built as a result. For example, guidelines on the interaction of Auditory Icons and Earcons can be explored.

The research presented in this thesis is centered on a prototype for such an audio design system. This prototype will be designed using timbre spaces (Hourdin *et al.* [12]), based on perception experiments in humans by Grey [13]. A timbre space presents an alternative representation of sounds that allows timbre to be explored directly, allowing object descriptions, perceptual descriptions and musical descriptions to be encoded in a common format.

Sounds will be analysed and loaded into this timbre space where they can be modified before being output via a synthesiser. The timbre space is a useful consideration for such a system as it provides a geometric representation of sound within a multi-dimensional space, allowing sound to be manipulated using generic geometric transforms, and allowing comparison between sounds, both in terms of musical groupings and in terms of finding sounds which do and do not possess perceptual attributes such as urgency.

This research aims to show that timbre spaces provide a new way to study audio design and to explore the opportunities provided by timbre spaces. A number of different timbre spaces are compared to evaluate their suitability for audio design, the output quality of the timbre spaces is evaluated objectively and improvements to this quality are tested. Audio design in the timbre space is investigated by studying the geometric nature of the space and the inclusion of Auditory Icon timbres. Perception of sound is added to the design system through the encoding of urgency in the timbre space. Two separate encodings are tested for their effectiveness in a trial on human listeners.

1.2 General overview of the field

Sound has always been an important part of interacting with the physical world. Sound tells us when someone is coming up behind us, when we need to change our spark plugs or, in the case of a fire alarm, when we should evacuate a building.

Against our rich sonic environment, the computer interface is a poor cousin. Most sounds from our machines are static sounds that do not change to reflect changes in the environment and often have little to do with the events that trigger them.

Gaver noticed this [14] and developed a system of Auditory Icons whose sound was related to the action to which they were connected, and whose properties could be adjusted to reflect changes in the underlying environment.

Since then, many people have developed sonic interfaces to various virtual environments and data sets, but each one is distinct, and despite numerous guidelines defining how each type of interface should be designed, and how people will interpret these, there is no common tool for developing or evaluating these sonic interactions.

In the design of Audio Aura [10] for example, the following guidelines were followed to prevent an "alarm response" in the users:

> "[Background sounds are designed to avoid] sharp attacks, high volume levels, and substantial frequency content in the same range as the human voice (200 - 2,000 Hz)." [10, p3]

The quote is then followed by a note that current audio design often induces this alarm response, intentionally or otherwise.

The computer music community on the other hand has defined many methods for creating and manipulating sounds through a small number of common interfaces. Each of these methods has its own strengths and weaknesses.

The most basic musical interface is the MIDI (*Musical Instrument Digital Interface*) standard, which defines a series of commands that specify musical notes and operations on these such as sustain, pan and instrument used. Short musical segments generated from these commands can be used as Earcons [8].

The biggest problem with MIDI is that the output sound is not guaranteed. Although there has been some recent standardisation, the basic MIDI specification does not guarantee any particular sound is available on the playback device or that the device will reasonably interpret all the commands. In the most extreme cases, MIDI devices will ignore most commands and will only play one note at a time, ignoring any other notes played at the same time.

Programs like Csound [15] are a step above MIDI, and allow the designer to define the instruments as well as the notes, giving the composer complete control over the whole sound of a performance.

Another popular field within computer music concerns the transformations made available by the various Analysis-Synthesis techniques, a review of which was compiled by Masri *et al.* [16]. Analysis-Synthesis allows composers to manipulate sounds directly rather than via the sources that produce them. It allows sounds to be sliced, stretched, reversed and morphed into other sounds.

Since the aims of the MIDI, Csound and Analysis-Synthesis techniques are musical production, little research has been performed to their perceptual relevance or their use as a design tool for non-musical sounds that convey status information. In particular, composers are rarely interested in whether a sound reflects the material an instrument is constructed from, or the size of that instrument, which are important parameters for Auditory Icons. Whilst composers do consider the emotional perception in music, which is reflected in some of the research, they are rarely concerned with the repeatability of emotional response required for interface design, nor are they concerned with generating a set of sounds across a range of emotional responses that could be used in interfaces to signify the relative importance of two events or the evolution of a single event.

The aim of this research is to go a long way to combining the work in these two areas, allowing interface designers access to complex acoustic and musical methods for developing sound through an interface defined in terms of human perception and HCI design.

1.3 Aims and objectives

In summary, the overall goal of the thesis is to see whether timbre spaces can be useful for audio design. This goal can be realised through the following aims:

1. To compare a variety of timbre spaces on a range of measures important for audio design;

2. To implement Earcons and Auditory Icons as models within the timbre space, in order to draw on previous knowledge of audio design;

3. To compare two ways of modelling perceptual knowledge within a timbre space, providing the system with new design possibilities;

4. To discover the advantages and disadvantages of the timbre spaces for audio design compared against existing audio design systems within the computer music and HCI communities.

For this final aim, the alternatives considered include audio design via Auditory Icons and Earcons, and will also consider a variety of synthesis systems that can produce a wide range of timbres but require complex interaction of parameters to do so.

1.4 Typographic conventions

In the rest of this thesis, the following typographic conventions are used. All text in `fixed width font` is a function name, code snippet, or extract from a structured data file. Text in *italic fixed width* indicates an argument that must be replaced with a variable name or defined type. Text in **bold fixed width** indicates a defined name suitable for use as an argument.

Text in *italics* usually represents a new definition, either introduced from a cited reference or created for the purpose of the thesis to describe a new concept. *Italic* and **bold** text is also used to emphasise points within lists, tables and figures.

1.5 Structure of the thesis

Chapter 2 discusses the current research in sound and audio interfaces, and contains a brief discussion of human perception and audio interface research. Chapter 3 discusses the theory and practice of timbre spaces, including what they are, how to construct them and how to extend them for audio design. This chapter also discusses some of the problems with timbre spaces compared with other representations and how those problems can be addressed. Chapter 4 surveys the technology and design behind the implementation of the system developed for this thesis, and discusses the problems encountered with the approach as well as the solutions implemented to overcome them.

A pilot study to compare timbre spaces was performed. The results of this pilot were used to run an experiment looking at perception in timbre space whilst data was being generated to compare a wider range of timbre spaces. Chapter 5 discusses the results of the experiments to decide which analysis techniques and parameters should be used for the timbre space and whether the concept of timbre spaces is suitable for audio interface design. Chapter 6 discusses experiments on perception in timbre space in which volunteers were asked to rate a set of sounds for 'urgency' in order to determine the perceptual relevance of a specific manipulation in the timbre space.

Chapter 7 discusses the results of these experiments and the overall suitability of the timbre space as a means to manipulate audio and perception of audio and then discusses possible future directions to improve and extend audio design using timbre spaces or related models.

The appendices contain important pieces of the current implementation. Appendix A is an IDL header file describing the components that make up the system. Appendix B contains an annotated file that demonstrates how sounds and transformations can be encoded for the timbre space and audio synthesis parts of the interface. The materials for the major experiments performed during this research are provided in Appendix C and Appendix D. The sounds generated during this research are available on the author's website (`http://www.dcs.gla.ac.uk/~can`), which is also provided on CD with this thesis, and described in Appendix E.

Chapter 2. Audio perception, interfaces and synthesis

2.1 Introduction

This chapter discusses some of the current research in the fields of audio perception, audio interfaces and sound synthesis. The focus here is on tools or experiments that have been or could be applied to the production of sound in an interface design context. The tools and experiments that relate to the timbre spaces used in the actual system covered by this thesis are presented in the next chapter.

This is a multi-disciplinary review, where research from psychology, computer music and Human-Computer Interaction (*HCI*) has been utilised in order to illustrate the design complexity required to produce a wide range of output sounds through a simple interface. The review of each discipline covers a number of well-established works that are referenced throughout that field, but there are very few studies that have combined the work from all these fields, despite the advantages of doing so. The HCI field has discussed ways to create and improve existing audio interfaces whilst the music community has had a lot of focus on the analysis of sounds for the purposes of manipulation and generation of novel sounds. The psychology literature seeks to discover not only how we hear but also the reactions we have to those sounds, how we describe them and what influence certain sounds have on our behaviour.

In this work, the design space from computer music is contrasted with studies of perception from psychology in order to improve understanding of the design of audio interfaces. The ultimate goal is to allow audio interface designers full access to the work in the other fields via an audio interface design system, although this work seeks only to show that such collaboration can be mediated through the timbre space concept and understands that the ultimate goal requires input from many researchers across all three fields to refine and extend the concept presented here.

2.1.1 Overview

The chapter opens with a short discussion of how we perceive sound in Section 2.2. The discussion below, and the system itself, focus on the perception of individual sounds. A short discussion of the effects of the environment including the effects of multiple sounds is provided for completeness.

Section 2.3 discusses the history of auditory interface design on the computer desktop, including a discussion of the difficulties of such design, which has been a motivator for the system developed during this research. Various different interface paradigms are described and compared. Although it is possible for the timbre space paradigm to cover most of the interfaces discussed, in order to maintain a sharp focus for the work, the primary interfaces considered are based on auditory representations of icons, for which there are two competing paradigms, one based on environmental sounds, one on abstract musical sounds.

Section 2.4 describes the methods used to generate sounds for these interfaces. As well as covering those currently in use for the interfaces described, there is a discussion on introducing new synthesis methods either to allow sounds to be produced for which there is no other source, or in order to tune the complexity of the synthesis to the available hardware.

2.2 Perception of sound

Perception is a very old topic in psychology. There are many books and studies on how people perceive the world and many texts on how we hear the world. It is impossible to cover here more than just a fraction of the work on human perception of sound so this section will deal with the concepts that have been applied to existing audio interfaces, and will pay little attention to the complexities of multiple simultaneous sounds, for example, the problems of sounds masking others and other auditory illusions, a short overview of which is given in Section 2.2.4. Few current interfaces take these into account, so this section will concentrate on the problems of generating single sounds.

This section concentrates on computer to human audio communication, rather than communication from human to computer or between humans. It also does not consider the possible effects of irritation that are related to certain signals in acoustics, although audio designers should be aware of the issues.

This section will cover what we perceive when presented with an audio signal, and how the human ear interprets the sound. It starts with a discussion on the current knowledge of perception of sound including information on the different ways we can describe the sounds we here. This discussion then leads in to discussion the perception of one particular construct, 'urgency', as an example of how such constructs can be useful for audio interfaces.

2.2.1 What do we hear?

There are four components to the way we hear a single sound: the pitch, the loudness, the duration and the timbre. This is the breakdown provided by Rasch and Plomp [17] with the addition of duration, added by Hourdin *et al.* [12], since duration is important in the construction of complex sounds from single sounds, as in musical composition. In addition, there are environmental effects related to the sound such as its location, reverberation and Doppler shift effects. These are not considered here since these environmental effects do not affect the production of a sound. For example, a violin string will vibrate identically whether in a practice studio or a theatre, although the perception of the violin will change. The smaller practice studio will create more echoes, creating the effect of a richer sound due to the increased presence of harmonics. Provided filters exist to model these environmental effects, a timbre design system does not have to model them.

Where sounds are combined, the differences in pitch, loudness, duration and timbre between the sounds are important as is the temporal pattern of the sounds. The differences can be exploited to create musical effects and are used to help us determine patterns in a series of sounds. This section concentrates on the issues of single sounds and leaves the sound combination components to the discussion of Earcons in Section 2.3.4.

For single sounds, the perception of pitch, loudness and duration are well understood. Pitch is related to the fundamental frequency of a sound, although complex sounds can share the same pitch even though they may only contain the harmonics of the fundamental frequency and not the fundamental frequency itself [17]. Loudness is related to the amplitude of the original signal by the phon scale, covered in Section 2.2.2. The duration of a sound is simply the time between the onset and offset and is easily measured.

Unlike these three measures, the timbre is a complex mix of the harmonics and their decay rates and the objective measurement of timbre has been a problem for at least 130 years. Helmholtz performed a lot of experiments in the 1870's [18], but it has only been with recent advances in signal analysis techniques that timbre has been seriously investigated, principally through the use of the timbre space, as used by Grey [13] and McAdams *et al.* [19] among others.

As there is little objective evaluation of the perceptual nature of timbre, timbre is generally negatively defined in a manner similar to the following:

> "The character or quality of a musical or vocal sound (distinct from its pitch and intensity) depending upon the particular voice or instrument producing it, and distinguishing it from sounds proceeding from other sources; caused by the proportion in which the fundamental tone is combined with the harmonics or overtones" [20, online edition].

Of the four components of single sounds, the pitch and duration are easily understood. The duration is linear in time and the pitch is related to frequency. Timbre and loudness are both more complicated than this, so will be discussed in the following section in greater detail.

2.2.2 Equal-loudness curves

In acoustics, it has been long known that the ear is more sensitive to some frequencies than others. This can be demonstrated when two sine waves with the same amplitude but different frequencies are played together. One will sound louder than the other. If someone is asked to adjust the loudness of one of the sounds until both sounds are equally loud, we can estimate the comparative frequency response of the ear to those frequencies. The latest international standard based on results of such experiments is ISO226 [21].

In acoustics, the input sounds are rated by their Sound Pressure Level (SPL), which is a physical property of the air and easy to measure. This is usually measured in decibels (dB), which is a logarithmic scale, due to the massive variation in pressure levels the ear can detect, covering a dynamic range where the loudest noise we can hear without damage is 12 orders of magnitude greater than the quietest noise we can hear.

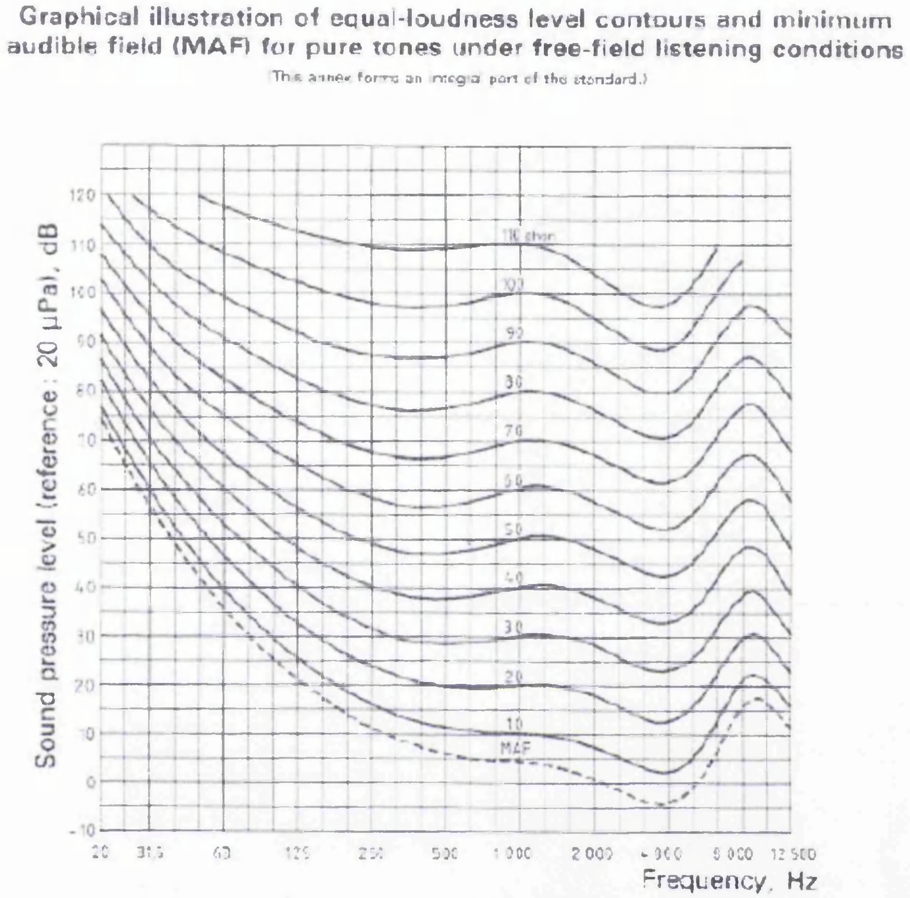


Figure 2.1: The phon curve. This shows lines of equal loudness, which is a perceptual quality. The picture is taken from ISO Standard 226:2003 [21]. The MAF line at the bottom is the Minimum Audible Field, i.e. the quietest sound that is still audible to most people.

As noted earlier, loudness is measured in phons, as demonstrated in Figure 2.1, where 10 phons is defined as the loudness of a 1kHz wave at 10dB SPL and 20 phons is the loudness of a 1kHz wave at 20 dB SPL. The other phon values are constructed according to the experiment above where a 1kHz wave at a certain phon level is played and the SPL of a wave of a different frequency is adjusted until they both have the same loudness. Each line on the graph is a line of equal loudness. Hence a 31.5Hz sound at 65dB is as load as a 8kHz sound at 30dB or a 1kHz at

20dB as they are all on the 20 phon curve.

The phon curves demonstrate the frequency response of the ear to a range of frequencies and SPLs. From the graph we can see that the ear is most sensitive to frequencies in the 3-4kHz range where the graph dips, showing we need little pressure to hear sounds in that region. The ear is least sensitive to low frequency sounds, and this is most noticeable at lower phons where the graph is steepest.

As the graph demonstrates, the response of the ear varies with frequency and with sound pressure in a non-linear fashion making the modelling of loudness a complex process. Any audio design system has to take such complexities into account. Fortunately, the experiment describes a method for uncovering the complexities by asking participants to rate pairs of sounds on one parameter (i.e. loudness) from which the graph is obtained. Provided there are methods to record such complex relationships, the design system merely needs to use the relationship to map between the perceptual concept of loudness and the sound pressure level of the signal.

2.2.3 Perception of phase

The loudness and frequency of the harmonics of a signal are not a complete description of the timbre of the sound, since the analysed data can also vary in phase. The phase of a periodic signal refers to how far through a cycle that signal is. For sinusoidal waves, the phase after one wavelength is 2π, which is identical to the phase at the start of the cycle (i.e. a phase of zero). Where there are multiple frequencies present in a sound, not only will their wavelengths differ, but their initial phase may differ. When these frequencies are represented as a set of sine waves, this phase difference can be observed as a difference in initial amplitude and direction for each frequency. If a sound is sampled at several regular intervals, the relative phase of its component frequencies will vary with the wavelength and the initial phase of each frequency.

Where the initial amplitude of a sine wave is non-zero (i.e. its phase is a non-integer multiple of π and is non-zero), audible clicks can be detected in the sound due to the limitations of sound-producing hardware. Other than such artifacts, the sound of a sine wave is independent of its phase. A cosine wave, which has a phase difference of $\frac{\pi}{2}$ from a sine wave, will sound identical to a sine wave at the same frequency. However, it has been suggested that phase information is essential in properly reconstructing the timbre of a multi-frequency sound [22]. In contrast, Risset and Wessel [23] discuss Ohm's acoustic law by saying "if the Fourier representations [see Section 3.3.1] of two sounds have the same pattern of harmonic amplitudes but have different patterns of phase relationships, a listener will be unable to perceive a difference between the two sounds, even though the two sounds may have very different waveforms" [23, p114]. They note that this insensitivity to phase only exists for periodic tones and the phase distortion in speech transmitted over telephone cables, where the frequencies propagate at different speeds, can make the resultant speech unintelligible. Hence, even if the production of a sound is carefully controlled, phase changes caused by the transmission of a sound can dramatically affect its timbre.

2.2.4 Situated audio

Audio effects such as masking and environmental effects such as reverberation affect our ability to distinguish individual sounds and timbres. Masking effects, whilst important, only tend to have an effect where multiple sound sources are used, which is important for the development of complex audio interfaces, but does not affect the properties of any individual sound. Since individual sounds are the focus of this research, masking effects are left as a consideration for further work. Further information on this phenomenon can be found in Wegel [24], which describes what is heard as the frequency and loudness of a tone is varied against a fixed reference tone.

Environmental effects are easily added to a single sound source after it has been generated. This is reasonable since the vibration of a string does not depend on the global environment, such as the room size or the height of the orchestra in relation to its audience, yet there is a difference in perceived timbre if a violin is played in a concert hall rather than in a small practice room, and this difference is far greater than the local environmental changes that can affect the string such as heat and moisture. These local environmental changes can be modelled during synthesis, but the global environmental changes, since they affect all instruments, are more easily modelled outside the synthesiser as a post-processing step.

To understand the changes brought about by the environment on our listening experience, it is necessary to consider how humans perceive the timbres of the sounds in their environment, and how this perception changes within the environment and between different sound sources.

2.2.5 Analytical listening

Analytical listening is the process of breaking a sound down into its constituent components in order to compare different sounds and how they are produced. It is part of the information processing view of perception and it is closely related to the field of signal analysis (see Section 3.3.1), since the analysis techniques are based on theories covered by the analytical listening view of human perception. Work on the cochleagram and correlogram described in that section and the experiments that led to their development suggest that the ear does perform some sort of frequency analysis since different nerve endings along different parts of the ear resonate at different frequencies.

In the analytical listening model, humans decode sounds into a sonograph style representation that breaks the sound into its constituent frequencies over time. The analytical model is supported by the fact that humans can perceive timbre through changes that do not overly affect the sonograph, such as pitch shifting and environmental distortion. Work has shown that the ear itself is able not only to detect these frequencies, but also other musical components of a sound, such as loudness and rhythm, without requiring conscious effort [25].

The identification of objects from their audio representation in the analytical listening model is achieved by applying inference to this representation with respect to stored memories. Patterns of sounds in memory are referenced in order to fit the new sound into existing experience. The newer field of ecological acoustics seeks to uncover what we consciously perceive when we hear, rather than just trying to understand the unconscious biological processes underlying hearing and memory. Marin and Perry discussed the effects of memory in relation to analytical listening [26]. Their study of patients with a variety of neurological conditions affecting memory and musical ability is shown to demonstrate the importance of memory on musical perception, but this is independent of the memory of separate timbres, such as used to name a violin or to identify a speaking voice.

2.2.6 Ecological listening

Gibson defined ecological perception [27] as an alternative to the analytical perception used to describe vision. Gibson viewed the visual perception of objects as part of the ecology of the viewer. Gibson argued that the analytical approach was useful for mathematical descriptions of the world and understanding the biological processes concerned with interacting with the world, but that considering objects in abstract terms such as wavelengths of light and planes of objects is insufficient for describing how we actually perceive the world in terms of objects and textures.

Moreover, whilst the traditional view holds that memory and inference are required to turn a retinal image into a perceived object, the ecological view suggests that the optical system perceives the invariants in the environment (such as edges between objects), and the memory processes work on this representation.

Audio design at the analytical level is easily accomplished since the construction of the sound is clearly defined in terms of frequencies, loudness and pitch. However, various experiments, such as Mynatt [28] and Vanderveer [29] have shown that people hear the sources of sound in preference to the properties of sound, a process described as *ecological listening* in reference to Gibson's work on ecological perception.

Gaver applied the concept of ecological perception to hearing [14], and argued that when people hear the sound of a slamming door, they will describe hearing 'a slamming door' in preference to 'a short, dull, loud thudding sound', or 'a quickly-decaying, low-pitched, high-intensity sinusoidal sound with few harmonics.'

These three descriptions can be thought of as a hierarchy of levels of description of the sound. These levels are defined here as: the *object level* (describing properties of the source of the sound - **'Breaking Glass'**), the *perceptual level* (describing what it is about the sound that helps us determine what the source is - **'Repeated high-pitched, randomly occurring contact sounds'**) and the *signal level* (describing how the sound can be synthesised - **'Randomly repeated, quickly-decaying sinusoidal sounds of high pitch with many higher harmonics**).

As Mynatt and Vanderveer both showed, most users will be working on the object level. Gaver [14] demonstrated this by using the term *everyday sounds* for description at the object level, and the term *musical sounds* for description at the perceptual or signal level. Ecological listening concerns everyday sounds and the object level whilst analytical listening is concerned with musical sounds and the perceptual and signal levels of description.

2.2.7 Ecological design

Gaver [14], Mynatt [28] and Vanderveer [29] all argue that listeners prefer to describe sounds at the perceptual or everyday level rather than at the signal levels. Due to this bias towards everyday sounds, this research seeks to allow users to modify sounds in terms of material properties as well as in terms of perceptual qualities. The object level encapsulates a set of concepts in the perceptual level, such as the relationship between heavier objects and lower pitch [30], allowing further objects, both real and virtual, to be transformed by adjusting perceptual properties.

An ecological audio design system would allow a user to take an existing object and adjust properties of it, so that one could take a door object with properties about rooms and slamming locks and derive from this an airlock object, where the far room could be either a large water expanse or a vacuum, thereby producing perceptually different qualities for opening and closing the door.

For example, a designer working at the perceptual level could be provided with a general sound control panel for creating sounds from auditory properties (e.g. {90% harshness, 30% loudness, 60% bright}). The designer could also be provided with a sound control panel at the object level concerned with object properties. For example, a door object would have properties of the form {Wooden, 20% of size of largest door} where 'Wooden' maps to a certain configuration of harshness and brightness (where wood is low on both scales but a metal jail cell door would be not very harsh and somewhat bright as the ringing can be heard). 'Size' would map to both a change in loudness and a change in pitch.

From the perceptual to the signal level, loudness maps to the amplitude of the synthesised sound via the phon scale, with harshness and brightness relating likewise to the properties of the harmonics of the base frequency and the decay of these. A representation is therefore required to capture the changes in timbre corresponding to perceived changes in harshness and brightness. In order to achieve this, some encoding of the perceptual constructs of "harshness" and "brightness" must be used.

2.2.8 Representing perception of timbres

Humans tend to perceive sound in terms of the object that created it, but as sounds get abstracted, or where those sounds seem to reflect a process of thought, there are many other aspects to the sound that a listener can perceive. For example, given the sound of footsteps, the tempo of the walk, the loudness of each step and the regularity of the rhythm all change as the person walking gets more anxious.

On a more abstract level, Edworthy *et al.* [30] set out to discover what effect changes in a sound made to how we perceive it in terms of a range of adjectives such as 'urgent', 'fast', 'slow', 'dangerous', 'smooth' and 'heavy'. They found consistent changes across participants for which changes in the input signal related to a change in the perceived value of each adjective.

Edworthy *et al.* also performed an experiment that focused on just one adjective - 'urgent' [31]. For this, a number of changes in the signal were devised and an ordering for each one was determined so that a consistent increase in perceived urgency could be predicted. The changes included adjusting the fundamental frequency of the sound, detuning the harmonics of the sound and adjusting the envelope of the sound so that the amplitude was constant throughout or faded in or faded out. Each of these musical properties of the sound was found to affect the perceived urgency in a predictable way such that distinct choices could be made to construct the most urgent and least urgent possible sound from the test suite.

These experiments show that, despite what Mynatt and Vanderveer conclude about humans not hearing at the musical level, there is a lot of perception that relates to the properties of the sound itself and not the object that creates it. In contrast to Gaver [14], Edworthy *et al.* demonstrate that perception can be described on an emotional rather than a musical level, in addition to the object level. This result is to be expected expected in a wider population where musical training is rare, whilst emotional engagement is common. Using these emotional perceptual constructs to identify timbre changes at the signal level allows a designer to construct new timbres using familiar terms that express the intent of the sound in an interaction, so bigger objects can be made to sound heavier and operations on more important files can be made to sound more urgent.

2.2.9 Implications for audio design

Ecological listening states that humans are more likely to describe sounds with reference to the apparent source of that sound as opposed to certain characteristics of that sound. Edworthy *et al.* show that, when prompted, humans can describe sounds at a more abstract level. Blattner *et al.* state that "The timbre of a sound is usually described with adjectives such as ***bright, warm, harsh, hollow, twangy,*** or ***brassy,***" [8, p26] showing that the ecological view does not provide the full picture of what we hear in sounds.

None of the papers report any listener describing a sound purely analytically, at the mathematical level required to synthesise a sound. Gaver [14] discusses the thought process required to extract such information from a sound for developing his Auditory Icons and it seems clear that most listeners are uncomfortable or unable to analyse sounds in the same way. It is at this level however that most current sound synthesis systems work, requiring the audio designer to examine sounds in this unnatural fashion.

An overview of such synthesis systems and their use in existing interfaces follows, in order to demonstrate how this gap has been bridged in the past, but it seems clear that an audio design system that allows manipulation and analysis of sound at the perceptual or object level would open up new avenues of exploration for sound designers currently constrained by working at the signal level.

2.3 Interfaces and design

This section will discuss the current research into auditory interfaces in a desktop environment, i.e. the standard *WIMP* (Windows, Icon, Menu, Pointer) interface found on most modern desktop and laptop machines and a number of hand-held devices including some mobile phones. The design system at the heart of the research is to be developed for this environment, for situations where the expressiveness and flexibility of the sound development is far more important than accurately recreating a sound from the real world. Such situations are cited by Gaver for the file copying Auditory Icon [9] where the expressiveness of pouring is more important than the realistic synthesis of a copying sound. For examples of realistic synthesis in a virtual environment, see van den Doel's work on environmental capture [32, 33] or Karjalainen *et al.*'s work on creating virtual stringed instruments [34, 35].

In a typical auditory interface, the sounds reflect the current state of the system. In some cases, this is used to provide feedback and information on user actions, as in the original work on Earcons [8] and Auditory Icons [14]. Others, such as Conversy [36] and Mynatt *et al.* [37], use sound as a non-intrusive way of providing status information on background tasks.

Audio is a two-way medium and can be used both to communicate to the user and to gain information from the user. Each direction of communication covers a wide range of techniques and challenges. This chapter will focus on the use of audio output in user interfaces, which has been the focus of work on Earcons [8] and Auditory Icons [14]. The audio outputs in these cases are generally used alongside non-audio input devices such as the traditional keyboard and mouse or gesture and haptic interfaces. Use of audio for both input and output can lead to feedback problems when the two channels can not be reliably distinguished by the user or the computer. Some discussion of the techniques used in audio analysis, which is one stage in the audio input cycle, is discussed in the next chapter, but the overall problems of audio inputs are not covered.

As an interface medium, audio shares a problem with the traditional graphical and command-line interfaces in that how the interface is perceived depends upon a number of factors including the background experience of the user and other information presented to the user either concurrently with or preceding the audio signal. This section discusses the different interface methods, following a brief exploration of this context sensitivity.

2.3.1 Cultural independence

Interfaces depend on context to be understood, since many sounds are hard to understand without context. A continuous sequence of two beeps and a pause, for example, can easily refer to an error that prevents a computer from booting up, such as a memory error, but can also be used to indicate an incoming phone call. The context of the interface often includes a lot of cultural context including language, custom and personal experience. For audio interfaces, the use of speech is an obvious case where a certain cultural background is essential for understanding the interface, and the interface must be customised for a new audience with a new language. Other cultural differences include abilities such as perfect pitch [38], where some people can reliably detect identical frequencies and others cannot, although training can improve this [38] and the ability of doctors, through training, to be able to detect certain illnesses via a stethoscope due to specific differences from normal patterns. There are a few culturally independent sounds that mean the same across all cultures, such as environmental sounds like wind and rain, or changes in sounds related to internal processes, such as increasing heart beat, and hence tempo, indicating an increase in exertion or fear, although the everyday sound description of such a change in a sound will be context dependent.

2.3.2 Speech

For many tasks, speech is the only audio output method that is viable. For example, for telephone-based banking interfaces, Web and e-mail access for blind users, or any other situation where the primary information is textual.

Speech can be a problematic interface if there is a lot of information, as shown by the confusion that occurs when many people are having a conversation. In Baldis [39], a group discussion is played to a subject who is then asked comprehension and person identification questions. The subjects are often confused about who is talking, even when one person is talking without interruption.

Although there are methods for increasing the usable range of speech outputs and increasing the comprehension, such as spatialising the sound as described in Baldis' paper, speech is ultimately less scalable than simpler audio interfaces, such as Conversy's use of multiple audio streams to convey workload [36], or the use of audio in computer games where several environmental effects can be present without confusion [40]. Orchestral music demonstrates the ability of the ear to pick out various parts of a complex stream of sounds by grouping the notes for each instrument together [41] but applying this to interfaces has proved trickier [42].

Non-speech audio interfaces are often quicker at presenting information than speech. Using the desktop metaphor, if a sound is associated with a file, there is a quicker sense of how big the file is from a pitched sound than from a speech sample reading out the size of the file as there is a lot more information to process in the longer, more complex speech signal. However, speech is the best way to get the *exact* file size across as it is less ambiguous than non-speech audio.

In general, speech interfaces are fairly advanced, with most text-to-speech synthesisers [43] offering great control over the timbre of the speaker, allowing complex speech to be generated from a simple text file and a small set of control parameters. Text-to-speech synthesis can help an interface to be culturally independent since a wide variety of languages can be handled by translating a text file, although the voice model must be able to handle the variety of syllables in the target language. This cultural independence is not as great a problem for iconic graphical output or non-speech audio outputs where a carefully designed set of icons can be used by a range of users across different cultures.

Whilst a generic audio interface design system could allow access to speech-based interfaces, particularly in controlling the timbre of the speaking voice, the leading text-to-speech synthesisers already offer advanced control of timbre, which is hard to replicate in a generic system that covers a wider range of timbres. This research, therefore, will not consider speech interfaces in the experiments performed in order to concentrate on more general principles of audio design. An audio design system that allows parameterisation of timbre that can be customised to specific domains could be used to generate speaking timbres for a text-to-speech engine. An example of such a customised timbre space for audio has been published by Terasawa *et al.* [44] during the final months of this research.

2.3.3 Auditory Icons

Auditory Icons [9] were devised by Gaver during his work on the SonicFinder interface [7]. They were developed according to ecological perception, as discussed in Section 2.2.6. They allow audio design at the everyday level. They are auditory representations motivated by real-world sounds. Gaver discusses a variety of sounds designed to resemble tapping, scraping, pouring and other real-world actions. The tapping sounds are used to represent the act of clicking on an icon, the scraping to represent dragging an icon over the desktop and the pouring is used as a progress indicator.

Gaver's sounds are produced by a mathematical combination of sinusoidal functions, where impact sounds and pouring sounds are base functions, and a bouncing sound is represented by repeated application of the impact function with varying parameters. Alternative synthesis methods for Auditory Icons have been developed by Keller and Traux [45] who used Granular Synthesis, and by Rocchesso *et al.* [46] who studied simplified Auditory Icons based on the simplest representation required to signify a particular type of event.

The sounds are parameterised such that different file types are associated with sounds of taps on different materials and the pitch of the sound represents the file size. Gaver describes this as an iconic relation between the sounds and the actions on the files:

> "iconic mappings are based on physical causation - the display entities look or sound like the things they represent" [7, p87]

Gaver points out later in the paper that there are many processes in the computer world that have no real-world analogue. For these processes, a choice of auditory icon is hard. The examples given in the paper are a disk-write error, which has nothing equivalent outside the computer, and copying files, where the sound of a photocopier provides little scope for parameterisation.

In the copying example, a pouring sound is used as a metaphor for copying files. The sound of a container filling up from another container can be readily related by a user to the concept of a file being copied from one place to another. Despite the fact that there is this metaphorical relation between the sound and the process, there is an iconic relationship between the sound and its source (i.e. the virtual containers). This iconic relationship provides a simple set of parameters for adjusting size, which is a concept common to both the sound source and the process the sound represents.

> "Metaphorical mappings make use of similarities between the entities or relationships of their domains" [7, p87]

One significant problem with Auditory Icons is that it is non-trivial to define new parameters, such as mapping the file size to how hollow the tap object sounds or adding a drum skin material to the tapping icon. Even where the mapping between a perceptual description of a sound and the system state it represents is obvious, it is rarely easy to modify the sound signal according to this perceptual description as standard sound editors operate at the signal rather than the perceptual level, making parameters such as 'object size' inaccessible without prior expert knowledge.

Although techniques such as Analysis-Synthesis, discussed in Section 3.3.1, allow more control over the sound than editing at the signal level, they are still hard to design and often very specialised to a single task in the same way as Gaver's mathematical descriptions of the icons. Gaver has said that more design tools are needed [14]. A system for audio design that can capture the Auditory Icon work would allow designers access to the environmental sounds they have produced.

2.3.4 Earcons

In contrast to Auditory Icons, earcons do not necessarily attempt to describe an event with a real-world sound. Blattner *et al.* [8] defines earcons as "nonverbal audio messages used in the user-computer interface" [8, p13] and defines two levels of earcons, "representational" and "abstract". Auditory Icons are defined as a type of representational earcons, whereas abstract earcons are constructed from short musical structures known as "motives", designed to have simple rhythmic and pitch structure, and hence provide design at the musical level. In common with the other literature listed here, the term *Earcons* is used to refer to abstract earcons, and the term *Auditory Icons* is used to refer to representational earcons.

Earcons are short musical motifs that are abstract representations of the information exposed by the interface. This does not imply a less intuitive interface however since Gaver noted many of the Auditory Icons he uses bear only a metaphorical relation to the task they describe rather than a truly iconic one, the copying sound being a good example of this.

Earcons are constructed as patterns of musical notes, where the instrument, duration, pitch, volume and other attributes are modified according to the state of the system. As Earcons can be made of many notes, the rhythm and tempo of those notes and their relative volume and pitch are also important attributes that are used in Earcon design.

Hierarchical Earcons, as used in the experiments by Brewster *et al.* on the effectiveness of Earcons [47, 48], assign different attributes to different levels of description of the interface. For example, menus are described by the timbre of the sound, and menu items are described by the rhythm of the sound.

Blattner *et al.* describe three ways to combine Earcons. In compound earcons, each motive has a fixed timbre, pitch and duration. Earcons are combined by playing a series of motives in sequence. For example, a menu item could be represented by playing the motive for "menu" followed by the motive for "file" followed by the motive for "open". Inherited Earcons create family groupings for similar events, so all the motives representing items in the "file" menu for example, may share a common rhythm. Complex Earcons are still constructed from sequences of motives, as in the compound case. Where the family groupings are used to construct and distinguish Earcons, they are classed as Hierarchical Earcons, as defined above. Other Earcon transformations described by Blattner *et al.* to generate family grouping include reflecting a motive in time (i.e. playing the notes in reverse order) or in pitch (e.g. turning a descending sequence into an ascending sequence), and adjusting the dynamics of the motive. They note that the perceptual significance of these transforms has not been validated.

The problems inherent in speech interfaces, where the sound can get extremely complex, apply equally to arbitrarily long Earcons due to this richness of space. It is not as great a restriction in Earcons however, as longer Earcons can be formed from the repetition of a few simple motifs, as used by Mynatt *et al.* in the design of Audio Aura [10].

2.3.5 Comparison of Auditory Icons and Earcons

Edworthy and Hards [49] suggest that listeners perform equally well on remembering the meanings of Auditory Icons and abstract sounds such as Earcons, unless there is a period of training where the users are told what each sound represents, when the Auditory Icons work better. This suggests that Auditory Icons are no more intuitive than Earcons. The paper then goes on to agree with Gaver that one of the primary reasons for this is that when an environmental sound such as an Auditory Icon is heard, the listener assigns meaning to the sound based on the object it sounds like rather than the process the designer has associated with that sound.

Earcons allow a more structured sound design space than Auditory Icons since Auditory Icons are independent of each other and can only be parameterised with respect to simple object interactions, such as a scrape. The interaction of multiple Auditory Icons has not received as much attention as multiple Earcons [50] and so Auditory Icon parameterisation is performed in isolation, considering single objects and only rarely considering multiple interactions between objects, such as demonstrated by Gaver's breaking glass icon. Earcons can be parameterised with a wide range of musical features, allowing a single Earcon to present much more information than an Auditory Icon, and multiple interactions grouped by rhythm and tempo are far more common. The parameterisation of an Earcon is also easier than for Auditory Icons as the parameterisations fit closely with musical concepts and are easily manipulated using musical theory applied with music composition tools such as sequencers, some of which offer scripting abilities to transform or generate patterns of notes according to higher-level parameters.

2.3.6 Combining Auditory Icons and Earcons

For Earcons, the atomic unit is the note, and a single earcon is a complex unit formed of one or more notes. Earcons treat timbre as a single dimension, which is a categorical[1] dimension rather than a numerical one.

Earcons use musical concepts such as pitch, rhythm, duration and tempo as further dimensions, modifying the parameters of the atomic units and their relative positions to reflect changes in the underlying process.

Auditory Icons treat timbre as the combination of many dimensions and rarely use pitch or tempo. Rare examples are given by Gaver's bouncing objects [9] where

[1] i.e. divides the space into categories, without ordering or scale. Some Earcons do however use similarity, where similar instruments are used to represent similar concepts [50].

the temporal proximity of events reflects the springiness and original height of the dropped object, or the ARKola project [51] which uses tempo to signify the amount of work a machine is doing.

For Auditory Icons, the icon itself is the atomic unit, as this is the smallest concept we generally perceive. This is despite the fact that the icon consists of many acoustic units (e.g. its harmonics). This is similar to the way that we can perceive a letter as an atomic object, even though it is made up of many lines or many pixels. We do not notice the underlying structure without extra effort.

The acoustic units of a sound are often the frequencies of a sound, which may vary with time. However, from Gaver's work, the bouncing icon has acoustic units of contact sounds, where each contact sound represents one bounce and the icon is interpreted not as a single contact but as a bouncing object.

As this demonstrates, there are many ways to divide a sound into acoustic units. The bouncing icon can be seen to be divided according to musical listening, whilst the frequencies represent division according to analytical listening. Whilst an automated analysis usually starts by breaking a sound into its constituent frequencies, as discussed in Section 3.3.1, an audio design system should abstract this analysis to break the sound into components closer to those used when a listener deconstructs the sound. Separating events in time is a logical and simple process to automate, but other divisions are harder to compute or to define. One way to examine such constituents of a variety of sounds is to determine what is perceived to be different between two sounds and how that difference can be modelled. This allows us to model, for example, the difference between wooden and metal sounds, as used by Gaver. In order to synthesise at the signal level, the decomposition into frequencies is the most appropriate, particularly as this is a general approach that will work on all sounds. An audio design system hence needs to be able to output a description of the sound suitable for input to a synthesiser.

At the perceptual or object level, an analysis similar to the bouncing icon may be more appropriate. A general audio interface design system must allow this logical division by the designer if it is to work as the designer might expect. It must also be expected that the system will not be able to break down the sound into anything other than frequency and time, or encodings of these, so there will be little scope for a general automation of separation of sounds in the fashion of the bouncing icon into individual bounces, although dedicated analyses for such separation may be possible provided the system is flexible enough to allow a variety of analysis and post-processing options.

By combining the complex atom and timbral manipulation of the Auditory Icons with the complex atom combination and musical manipulation of the Earcon, we can see that a system that allows control over both timbre and musical dimensions will have a much richer design space than either idea on its own. Whether this richer space will provide a more flexible and useful design space is a matter for investigation.

2.3.7 Other audio interactions

Two other areas that could be enhanced by a general system for manipulating a multidimensional sound representation are sonification, discussed in the NSF sonification report [52] and generic 3D visual interfaces.

A quick description of these applications follows, as a cursory examination, with a view to investigating the fields in depth should the system offer enough flexibility to explore them.

Sonification

Sonification, as defined in the NSF (National Science Foundation) sonification report, is *"...the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation."* [52, Section 2, online edition]

Sonification defines a mapping between status variables and sound properties with the intention that a listener can infer properties of the underlying data by listening to the sonic output.

The report argues that sonification is a good solution to the information overload we can experience in a visual environment. Since sound is an inherently dynamic system, it is more suited to input data that is very dynamic, either by having many changing input variables, or when the temporal changes in the data are very complex.

Simple examples of sonification using Auditory Icons and Earcons have been developed by Mynatt *et al.* [37] and Conversy [36], where the background status of many variables is presented in a non-intrusive audio format.

There are also many sonification papers that cover areas traditionally performed in visualisation, one prime example being work by Dombois on earthquake sonification [53]. Dombois describes taking recordings of earthquake data and shifting the wave signals into the auditory domain. The signal is played back faster than real-time and is presented in conjunction with a 3D visualisation. The paper describes the ease of use for analysing the data, as experienced by the author, but no comparative tests were performed.

In Hermann and Ritter's sonification paper [54], the ideas of a parameter map and a perceptual map in sonification are introduced. In their parameter map, each dimension to be sonified is attached to a parameter (or a meta-parameter) of the chosen synthesiser or sonic model, such that changes in that dimension affect a change in the model parameters and hence the sound. In their perceptual map, the sound produced depends not only on the data itself but also upon the interaction, mediated through a haptic device. The perception of the data is hence mapped according to the data itself and to the specific interaction with the haptic device. Their use of the term *perceptual map* hence conflicts with the use of the term within timbre spaces.

3D interfaces

The audio interface work encapsulated by Earcons and Auditory Icons was founded on the 2D graphical desktop metaphor using concepts such as files and folders. Many attempts to move beyond that metaphor are discussed in this section. These utilise 3D environments reminiscent of the world outside the computer.

It is useful to consider 3D environments as they often include support for audio interfaces alongside the 3D graphical interface, particularly since vision only works by line-of-sight whereas sound can be used to indicate actions outside the field of vision, either because it is behind or to the side of the user, or because the sound source is hidden behind other objects in the environment. The environments discussed below generally situate a static sound in the environment and apply effects to it to signify acoustic properties such as reverberation or Doppler effects. This means they can incorporate Auditory Icons and Earcons easily as static sounds in the environment but generally lack the tools required to parameterise the audio according to events in that environment. This restriction forces the audio designer to record a separate audio file or MIDI sequence for each type of interaction, although van den Doel [33] offers a virtual environment without this restriction. A generalised audio design system that allowed easy real-time parameterisation would expand the possibilities for interaction in such spaces and could also be applied to non-graphical 3D environments.

Many 3D Graphical User Interfaces (*GUIs*), such as Rooms [55] and X3D [56], treat their 3-dimensional world as an extension to the standard 2-dimensional WIMP interface, using the same point-and-click metaphors, with an added navigational component.

Rooms uses a 3D extension of the desktop metaphor to the extent of using standard folder terminology and using the Microsoft Windows icons. Although it presents a different set of sounds for the interactions (e.g. door sounds to open a folder rather than click sounds), it covers fundamentally the same audio environment as the desktop. It does not use dynamic sound to aid navigation or to provide feedback. It does not allow any modification of the audio samples used.

The world wide web has gained ground as a possible replacement for the desktop interface by taking existing concepts of icons and menus and extending them using hyperlinks to navigate through documents rather than abstract folders. This evolution can also be seen in 3D worlds inspired by web concepts. One such 3D language, developed along the same lines as the web itself, is X3D.

The X3D specification (formally known as VRML - Virtual Reality Markup Language) has been developed by the Web3D consortium [56]. It is a standard for describing 3D environments and allows events to be attached to any object in the environment. Some of these events trigger sound samples or Earcons. The only sound property that can be changed dynamically is the pitch. This is adjusted by changing the sampling rate, and hence will also adjust the length of the sound when samples are used. The X3D specification also includes a spatialisation component, but otherwise its auditory interface is no more advanced than the desktop. Despite these limitations, Mynatt *et al.* were able to use VRML for their Audio Aura experiments [10] using recorded Auditory Icons, speech and Earcons in an environment based on the building where they worked to present relevant information to the employees who were off-site.

More inventive virtual environments are designed to present the information in ways that can only be done in 3D. This can include locational information, and can also include virtual interactions with objects designed to behave like their real-world counterparts, as studied by van den Doel [33].

The best examples of virtual environments with good sonic interaction are often found in computer games. Andresen [40] discusses augmenting id Software's Quake1 environment with extra auditory feedback to make it accessible to blind users. At the basic level, speech has been added to augment the textual output. In the game itself, many audio cues are used. Aside from the simple object sounds, footsteps and door noises in the original engine, ceiling fans were added to the corridors. These fans, when combined with the occlusion of thick walls, allow users to easily find doors between corridors, particularly as the sound is panned to aid the user in orienting towards this sound. Bumping and scraping sounds are also used with panning to allow the user to detect when they have hit a wall and whether they are sliding along the wall or are stuck. Andresen also makes the important point that adding the sound information into the environment not only makes the game accessible to more people but also gives an extra set of information to existing players which can enhance their sense of immersion in the game.

2.3.8 Summary

Audio environments based on speech and within computer game environments are often far more advanced than non-speech desktop environments where users do most of their interaction [11]. To redress the balance, an audio design system should build on the ideas of Earcons and Auditory Icons, built to work within the desktop metaphor, and extend these with ideas borrowed from other auditory environments discussed here and the work on perception discussed earlier in this chapter. In order to consider all the available possibilities, the next section will look at the variety of output techniques found in the interfaces listed here, as well as some synthesis techniques that have been used in existing audio design work by musicians and composers.

2.4 Synthesis

This section discusses some of the methods available for synthesising a sound within a computer. Sampling (sometimes covered by the term "Wavetable synthesis"), is a common method in modern synthesisers and computer interfaces that has good sound fidelity but poor manipulation potential. The other synthesis models present different balances between ease of control, speed of synthesis and hardware support. After a short discussion of the requirements for a synthesis system, this section reviews the MIDI specification [57], which is the most common synthesis control language and is understood by a large variety of instruments, specialist music devices and computers including hand-held devices and mobile phones.

For this research, the synthesiser used should be capable of replicating the capabilities of Earcons and of Auditory Icons, so needs to be flexible and easy to parameterise in the timbre space. In addition, the following criteria were considered:

- **Fast synthesis.** A truly interactive synthesiser needs to be able to generate output quickly and change that output immediately if the parameters change. An interactive synthesiser is not required for evaluating the quality or capabilities of the audio design system, but is required if that system is then used to generate a production audio interface.

- **Wide timbral range.** The synthesiser should be able to generate sounds across the entire range currently available in Earcons and Auditory Icons, this includes orchestral sounds, percussive sounds and environmental effects such as pouring water. If one synthesiser cannot fill the entire timbral range, multiple synthesisers must be implemented.

- **Easy parameterisation.** The synthesiser needs a representation in the audio design system such that it is a simple matter to turn a sound into a set of parameters.

- **Low data requirements.** Some synthesisers need to store a lot of data themselves to generate high quality sounds. If this data needs to be supplied alongside the parameter data, the storage requirements will be much higher than the storage requirements for other synthesisers. To avoid such large storage requirements, the preferred synthesisers to use alongside an audio design system with its own timbre representation are simple with low data requirements that can rely on the timbre representation to encode the sound complexity. This arrangement provides the most flexibility for the design system.

2.4.1 MIDI

The MIDI (Musical Instrument Digital Interface) specification [57] seeks to define a common protocol for the control of musical instruments. It is commonly used in the design of Earcons [8] and provides a low-level parameterisation that most of the following synthesis techniques have been adapted to.

At its core, the MIDI specification defines a clock signal and a series of time-dependent messages which MIDI-capable devices can process or ignore. Each MIDI device is assigned one or more device numbers and one or more channels, and will respond only to messages that correspond to its assigned device and channels.

Most MIDI instruments accept messages to select a timbre to use for the synthesis, a list of which was added to successors to the MIDI standard. Most MIDI devices also accept commands to play a certain note at a certain pitch for a certain duration. Other common messages include changing the pitch or volume of a playing note, changing the volume of an entire channel, changing the tempo using the MIDI clock and adjusting the balance between the left and right stereo channels.

Any synthesis technique to be considered as a design tool for Earcons must support the MIDI standard, so must allow easy pitch and volume changes, and must be fast enough to respond to a message so that it can be processed at the appropriate time.

2.4.2 Wavetable synthesis

Wavetable synthesis is the term used in the MIDI specification [57] to describe the process of recording a sound wave into memory (the *Wavetable*) and playing it back when needed. The process of recording this sound is often called *sampling*, and some systems use this term to describe the whole process, particularly in systems that offer methods to manipulate the sound in memory. More recent work [58, 59] calls this type of synthesis *fixed-wavetable* and uses wavetable synthesis as a general term encompassing this as well as techniques such as granular synthesis (see Section 2.4.7 below).

Sampling is a simple and quick process as most modern computers have built-in chips (known as Digital to Analog Converters, or DACs) to convert incoming sound into sampled data and sampled data into sound. This ease of use has made it by far the most popular method of sonifying the interface in terms of user reach, with the Microsoft Windows desktop, VRML and hand-held devices all using the technique.

Most sampled sounds are stored in a Pulse-Code Modulated (PCM) format, which stores the data as a list of samples, one sample per channel per time step. The header contains information on how many channels the data has and the sample rate of the file. A number of compression techniques exist, MP3 [60] being the most popular, but the sound must be converted into PCM format in order for the sound card to be able to read the data, and this is how sounds are stored in the wavetable on the sound-card.

The great advantage of sampling is that the sounds are as similar as possible between computers, since each computer will reproduce the same sound from a given file, within the hardware limitations of the device. The recorded sound can be modified in many ways, such as pitch adjustment (with the most advanced equipment able to do this without modifying the length of the sample), filtering (removal of certain frequencies of the sound) and adding echoes and other environmental effects, all of which modifications can also be applied to the output of the following synthesis methods, although these all add a computational cost above the basic playback.

One of the major drawbacks of sampling in an interface is the large size of the files, being several orders of magnitude larger than the parameter definitions for synthesis algorithms. There is also the problem of recording a different sample for every type of interaction we want to simulate. This requires a large amount of storage space and is also labour-intensive in the sound capture stage.

The audio design framework envisaged in this thesis, whilst concerned with parameterised and interactive sounds, should also able to generate sounds for a wavetable or sample based interface to allow it to generate sounds for most modern computing devices, including hand-held devices. This is important as it allows the new interfaces to be tested alongside a familiar environment to see what effect, if any, a particular design has on a system. Where speed is an issue, such as on smaller and older computing devices, this may be the only option. In this case the audio design framework is used to generate a series of static sounds, in much the same way as modern desktop and web icons are developed using vector graphics packages then saved for use as bitmaps to avoid compatibility and rendering speed issues.

2.4.3 Synthesis using oscillators

The simplest form of synthesis for sounds generated entirely within the machine is based on the manipulation of simple waves generated by oscillators. The idea is to take an input wave and modify it through various filters to create a complex output wave. The input wave is usually a simple sine wave, although others such as sawtooth and square waves are also common.[2]

The oscillators are modified by a number of parameters to change the amplitude, frequency or other properties of the wave. This wave can then be summed with or used to modify the parameters of another filtered wave. This process can be repeated with as many oscillators and filters as the synthesiser can handle.

Each parameter of each wave in the process can be shaped by an *envelope*, which defines the shape of one of the attributes of the wave. At each point on the wave, the attribute is multiplied by the value of the envelope. The wave attribute is said to be *convolved* with the envelope. In Figure 2.2 the amplitude of a sine wave has been convolved with the envelope shown.

Unlike wavetable synthesis, an oscillator model can reproduce a sound with much less data as it needs to store only the parameters to create a sound rather than the sound itself. Moreover, these parameters can easily be adjusted in real-time to create effects that are difficult if not impossible to achieve interactively using a wavetable approach. These two advantages over the wavetable approach offer further possibilities to interface designers. The lower data requirements offer a wider

[2] For a general discussion of the types of synthesis available using oscillators, including some not mentioned here due to their unpopularity in user interfaces, refer to Roads [58].

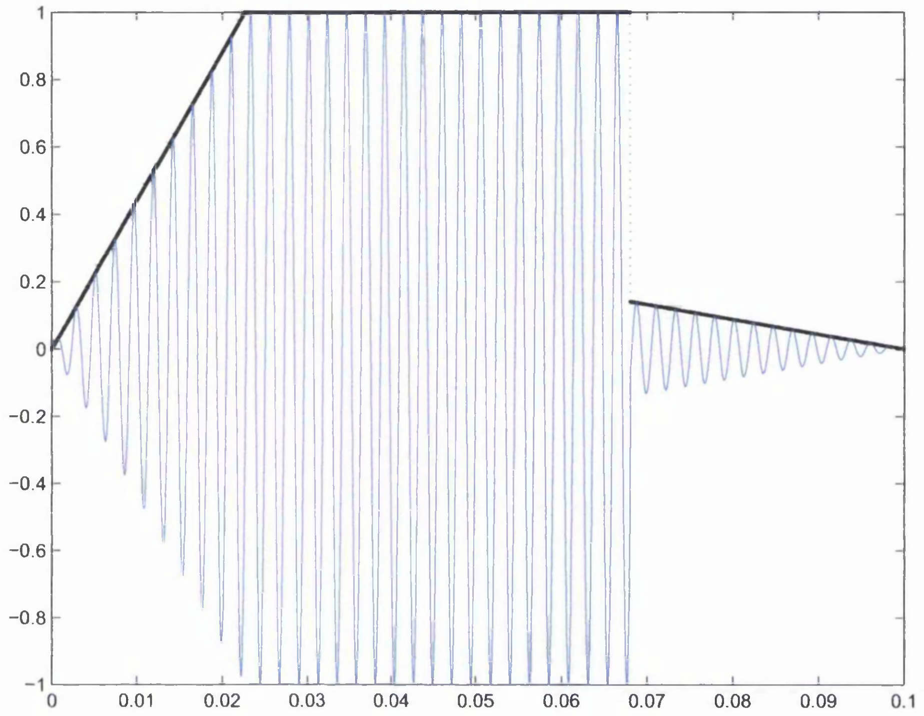


Figure 2.2: A sine wave shaped by an amplitude envelope. The thin line is the wave, the thick line is the envelope.

range of sounds to the designer, whilst the interactive adjustment of parameters allow the sound to change in real-time according to some underlying data. Compared to a wavetable-based synthesis however, each of the techniques listed here has a higher computational cost, which can limit the opportunities for real-time interaction if the processor is busy and there is no supplementary processor, such as a specialised sound chip, to handle the synthesis.

The interactive adjustment of parameters is shared amongst all the techniques listed here, but the additive synthesis and frequency modulation synthesis techniques that follow have the advantage of simplicity. In the additive synthesis, the simplicity is in the parameters themselves, where complex sounds can be described merely by the summation of waves described only by their amplitude, phase and frequency over time. The downside to this approach is the increased computational cost of producing enough waves to model the desired sound. In frequency modulation synthesis, simplicity comes from the small number of parameters required to produce complex sounds due to the way the oscillators are linked. This simplicity comes at the cost of not being able to intuitively derive the parameters for a given sound.

2.4.4 Additive synthesis

Additive synthesis has been used to generate sounds since the first pipe organ was built, [61], and has been used since the earliest days of electrical and electronic synthesis (e.g. Cahill [62]). Additive synthesis is also the basis of Gaver's Auditory Icons [9]. It works by generating parallel streams of input waves, where each wave has its own frequency, phase and amplitude envelopes. The output is the weighted sum of these input waves. Gaver used these to generate contact sounds. The phase and amplitude envelopes are parameterised in order to produce realistic timbres for a variety of real-world contacts.

Where a sound has been analysed using the Fourier transform described later (Section 3.3.1), the additive synthesis approach provides an inverse mapping from the analysis to the sound. This is best demonstrated in the work by Hourdin *et al.* [63] where the resynthesis of a sound via a more compact representation is the aim. This process demonstrates that additive synthesis can recreate arbitrarily complex sounds, as any sound can be analysed.

Additive synthesis, though simple and powerful, places relatively high demands on non-optimised hardware and requires a lot of control parameters, i.e. frequency, phase and amplitude parameters for each wave for each instrument for each time-step. The size of the parameter set can hence become quite large, so many methods to reduce the amount of data stored have been proposed. For example, the Timbre Engine [64], which allows real-time timbre editing, shows that additive synthesis can work in real-time given a well-defined domain of sounds by presenting the user with a restricted parameter set which is then mapped onto a set of synthesiser envelopes, enabling the sound to be described by 8 parameters per partial for the entire sound, where each parameter maps to a number of frequency, phase and amplitude envelopes.

Without specialised hardware, the additive synthesis parameters have to be limited in order to generate sound in a reasonable amount of time within a multi-tasking environment. Therefore, additive synthesis is used here only in a non-real-time setting.

2.4.5 Frequency Modulation (FM) synthesis

On computer systems that have on-board synthesis as well as sampling, the most common synthesis technique is Frequency Modulation (FM) synthesis, developed by Chowning [65]. This has been used for commercial electronic synthesis instruments since Yamaha released its DX7 synthesiser in 1983 based on Chowning's patent. Although some of the newest professional sound cards have Physical Modelling synthesis, it has yet to be found in any significant number of desktop machines, unlike FM synthesis. For modern computers without a built-in FM synthesiser, the process is fast enough to be computed on the main processor during an interactive session.

FM synthesis is based on the following equation:

$$x(t) = sin\,(\omega_c t + I \cdot sin\,(\omega_m t))$$

Where $x(t)$ is the output amplitude at time t. The ω_c component of the equation refers to the angular frequency[3] of the carrier wave, which is the primary wave in

[3] Where angular frequency is equivalent to the frequency in Hz multiplied by 2π

the sound, and is often the fundamental frequency that we hear[4]. The angular component of this wave is varied by the modulating wave whose angular frequency is denoted by ω_m. The scaling term I on the modulating wave is known as the modulating index.

The characteristic FM sound is fairly metallic so the FM output is often filtered in order to produce a more natural sound. The most useful filter for this purpose is the envelope, which is used to soften the onset and termination of the sound by modifying the amplitude. In addition, further sine waves can be multiplied by or added to the output to generate much richer tones [66], although this adds to the complexity of the synthesis.

The classic FM synthesiser can generate much more complex timbres than the additive synthesiser for the same computational cost, but the price of this complexity is paid in having a more complex relationship between the parameters and the output sound. The complexity of recreating sounds using FM synthesis has been explored by Beauchamp and Horner [67]. If this complexity can be harnessed by the timbre representation, an audio interface could be developed that could play many complex sounds that are interactively controlled without placing a large computational burden on the rest of the system. Where hardware exists to perform the synthesis outside the CPU, this could be used to generate a dynamic audio interface with minimal CPU burden.

2.4.6 Subtractive synthesis

In additive synthesis, a wave of a single frequency is created and a sound is produced by summing many of these waves together. Subtractive synthesis reverses

[4] This does not hold for certain values of the modulating frequency where that becomes dominant, or creates auditory illusions. It also does not hold where the modulating index and frequency are large enough that the signal is very noisy

this process by starting with a sound containing all frequencies (i.e. white noise). This white noise is then shaped by a series of filters that remove frequencies from the sound until only the required frequencies are left. This is the technique used by synthesisers such as SYTER [68, 69], developed at the Groupe de Recherches Musicale (GRM) studio in Paris in the late 1970's.

The subtractive synthesis model allows sounds to be produced with a dense frequency content much more quickly than other models. This is useful where the system being modelled is naturally noisy or contains a dense frequency set. One case where this method is commonly used is in speech models where each formant covers a fairly wide frequency band that is densely packed (e.g. Flanagan [70]).

The filter complexity required for this type of synthesis can make parameterisation difficult, but this can be a faster synthesis model than additive synthesis given specialised hardware since far fewer oscillators are required. To avoid the need for specialised hardware, this technique is not considered further.

2.4.7 Granular synthesis

Granular synthesis is a general term for a class of synthesis algorithms that generate complex sounds from small building blocks, often known as *grains*, but also referred to under many other names.[5] Each grain can be anything from a simple pulse of sound to a sample taken out of a longer recording. By combining these grains in various ways, a complex set of timbres can be built up from simple building blocks. As each grain is typically 1-100 ms long, and grains often overlap, there are often many grains in one second, creating a complexity of control that requires abstraction in order to be tractable for a designer.

As an alternative to samples, grains can also be built from wavelets (see Sec-

[5] see Roads [58, Part 2, p168] for a list.

tion 3.3.1) as in Kronland-Martinet *et al.* [71]. The grains are still combined in the same way, but the wavelet approach gives even more control over the timbre of individual grains. A similar approach uses the inverse Short Time Fourier Transform (see Section 3.3.1) to generate sounds from a given Time-Frequency Representation [58].

Granular synthesis has been used to generate Gaver-style environmental sounds [45] which can be used on the desktop in a similar way, where several streams of grains are played simultaneously, allowing multiple grains to build up to add texture to the timbre and giving control over the temporal distribution and evolution of the sound, in much the same way as Gaver used multiple tapping sounds to generate an icon of a bouncing object.

The sample-based nature of granular synthesis means that any hardware specialised for wavetable synthesis, as most modern sound cards are, will be able to perform some sound manipulation away from the main CPU, reducing the load of the interface on the CPU. The storage cost of such an interface is reduced by utilising the sound-card's on-board memory required for wavetable synthesis.

The parameterisation of Granular Synthesis is difficult [72], and the emulation of target timbres is a non-trivial process. Furthermore, the number of different parameters available makes it tricky to encode within a timbre representation. The size of this data, alongside the possible cost of storing or recreating grains, mean that Granular Synthesis may be added to the design system in the future but its complexity would complicate experiments in the prototype so will not be implemented within this research.

2.4.8 Physical modelling

Physical modelling is a general computing technique that tries to model real-world physics. When used for sound synthesis, the vibrations of objects are modelled and a virtual microphone is placed in the virtual world to generate the output sound.

The most common physical modelling technique in the literature and in use is waveguide modelling, based on the Karplus-Strong algorithm [73], for example, electric guitar models [74], Drum models [75] and environmental models [76]. Other methods of physical modelling are based on physics equations, with van den Doel [32, 33] quantising the space and pre-processing it to generate a sound map of the environment, and Palumbi and Seno [77] using standard string equations for tension and friction and implementing these directly via calculus methods to obtain highly realistic sounds. These other methods are listed here for reference only and will not be used in the final system as the waveguide is the only one that can currently be computed in real-time on a standard desktop computer. However, these synthesisers present a useful set of parameters for audio design and their implementation within the framework allows designers access to a wide range of sounds without requiring samples for the timbral range of each instrument.

2.4.9 Justification for choice of synthesiser

The criteria previously listed for choosing a synthesiser were: fast synthesis, wide timbre range, timbre parameterisation and low data requirements. All the physical models target a specific timbral range and do not generalise to other types of sound, so a physical model audio design system would need to implement all the different synthesis types, with varying synthesis speeds and data requirements, in order to model a wide range of timbres. The physical models do offer a method of control of sound within the timbre space so their parameter set could be used to encode transformations which are then output to other synthesisers.

Wavetable synthesis has fast synthesis and an unlimited timbral range since any sound that can be heard can be recorded. It is hard to generate new sounds sufficiently different from those in the wavetable to be able to parameterise effectively using wavetable synthesis. There are also high data requirements to store the required sounds. Whilst this technique is not useful in the design stage, where parameterisation is limited and factors other than storage space drive the design, an audio interface developed using another synthesis method could be implemented using Wavetable synthesis, allowing such interfaces to be implemented on devices unable to support the design and interaction criteria imposed above.

The Granular Synthesis technique offers an unlimited timbral range but it does require a sound bank of grains to achieve this, the parameterisation of which is a non-trivial problem. This sound bank can be generated from samples or from wavelets or other objects generated as required, offering a trade-off between synthesis speed, timbral range and data requirements. The complex parameterisation over a large number of parameters increases the complexity of the technique in terms of its timbral parameterisation. The simplicity of the additive and FM synthesis techniques precludes the Granular Synthesis approach from being considered further for this research.

Subtractive synthesis also has a complex timbral parameterisation since it requires a series of filters with associated parameters. As few machines have specialised hardware for this technique, it can also be slow.

The FM synthesiser can be simple or complex as it allows a wide range of configurations of its oscillators. Oscillators can be connected either in parallel, as with the additive synthesiser, or in series, and in any combination thereof. The more complex the synthesiser, the longer the synthesis takes, and the more complex the timbral parameterisation, but the wider its timbral range. The only data requirement for the FM synthesiser is to record which configuration is being used. Since the FM synthesiser can be easily tuned in its tradeoff between speed and timbral range, it is an interesting synthesiser choice that allows exploration of the importance of those parameters for audio design.

The additive synthesiser, as the inverse of the Fourier transform (see Section 3.3.1), has the capability of synthesising any sound, so has an infinite timbral range with low data requirements, although it requires many more parameters than the physical models or the FM synthesiser to produce output of similar complexity. If the number of oscillators used is sufficiently limited, the synthesis can be fast at the cost of a smaller timbral range. The definition of timbre as constructed from the interactions of the harmonics of a sound over time suggests it is an ideal match for additive synthesis, leading to a natural timbral parameterisation.

This research will therefore explore the use of the additive synthesiser and the FM synthesiser as output devices since both offer low data requirements and the opportunity to adjust the balance between speed, timbral parameterisation and timbral range so that the full capabilities of the design space can be explored.

2.5 Conclusions

This literature review has surveyed the research in music synthesis that can be applied to the design problems reported for audio interfaces. Auditory Icons and Earcons are two different concepts for developing sounds, and use different synthesis methods. Although a lot of work has been done in each field in isolation, most of the work that looks at both tends to see them as distinct methodologies. A system that brings the two of them together encourages design that uses the best aspects of both, using Auditory Icons and physical models to generate the timbre of a sound whilst using Earcon techniques to combine these icons to form complex spectral and temporal patterns to create a denser information space.

In terms of musical and interface synthesis, there are a large number of techniques for sound generation within the musical fields, from methods based on recorded sounds and granular synthesis to more abstract sound creation such as FM synthesis. In the arena of audio interfaces, the synthesis types have been more limited as it has often been tricky for designers to operate these synthesisers in terms of creating a desired sound. Although additive synthesis and physical modelling have been popular, as they are easier to control, there are fewer examples of granular synthesis and FM synthesis in the literature despite the advantages they are said to offer in creating complex sounds.

A further obstacle to interface designers is created by the complexity of our perception of sound. Many psychologists have created experiments to break down how our perception of sound works, but it is not a trivial matter to incorporate those insights into audio design. Additive synthesis models tend to be used for this but that limits the complexity of the output sound where real-time interaction is required. Most design of timbre is performed at the analytical or signal level, which ecological psychology suggests is an unnatural way of describing the timbres, since the signal description only defines the unconscious part of the hearing process in the ear.

Earcons and Auditory Icons offer design at higher levels of description, respectively the musical level and the object or everyday level. The popularity of these techniques in the field of audio interfaces suggests that designers prefer not to work at the signal level. To aid design, therefore, it is necessary to consider the perceptual and everyday levels of description, which correspond to the language used to describe sounds in ecological listening experiments. Easier creation of object models will also be considered. There is a lot more existing research on musical design than on perceptual and everyday sound design, so musical design is not the main focus of this research.

The field of musical research offers some guidance towards synthesis models that may expand the complexity of sounds without sacrificing real-time interaction. In addition, music research can offer a greater insight into how sounds can be analysed and manipulated. The understanding that arises from this can be used to provide a basis that allows perceptual information about sound to be encoded into interface design. This is explored further in the next chapter.

Chapter 3. Theory of timbre spaces

3.1 Introduction

To address the limited design options currently available for desktop audio interfaces and to create a strong platform for perceptual and other mappings, an audio representation is required. The representation should be able to encode both sounds and transformations and it should be easy to expose these to designers.

The representation chosen for this thesis is the timbre space, which uses a multi-dimensional space to represent both sounds and transformations. In this chapter, the construction, existing uses and potential of timbre spaces are discussed. This chapter will also show that there is a range of possible spaces and transformations and will show some alternatives to timbre spaces and discuss why they are unsuitable for the goals of this research.

Timbre spaces are a group of models used to analyse timbre by representing it as a multi-dimensional space [13, 19, 12, 78]. Each sound has a unique projection in this space. The two main types of timbre space are those generated from experiments on human perception and those generated mathematically by analysing patterns in the sound. As the literature has yet to decide on distinguishing names for these two types, for the purposes of this thesis, the former will henceforth be known as *human timbre spaces* and the latter as *automated timbre spaces.*

In human timbre spaces, each sound tends to be represented by a point in space, which is usually three-dimensional. In the automated spaces, each sound tends to be represented by a path in a multi-dimensional space which can have 8 or more dimensions. Hourdin *et al.* [12] showed that an 8 dimensional space was sufficient to capture 90% of the timbre information in their range of sounds, out of the 80 dimensions in the frequency space of those sounds.

Two common features of both types of space are that sounds which are perceived to be similar will have a similar representation in the timbre space, and the space itself is generated by a Multi-Dimensional Scaling (MDS) analysis, whose purpose is to reduce a high-dimensional data set to one of a lower dimension by filtering out the dimensions which contribute least to the variations in the data. In the human timbre spaces, the MDS analysis is applied to distance ratings of sounds obtained from human listeners. In the automated timbre spaces, the input to the MDS analysis is the output of a signal analysis of the set of input sounds.

This chapter will concentrate on automated timbre spaces, since they have additional features as well as automated generation that make them attractive for building a perceptual map. These features include a path rather than a point based sound representation and a readily available synthesis technique to produce sounds from the space. As some of the concepts used will be based on experiments with human timbre spaces, a short review of these and a comparison of the two types is presented.

In the context of audio design, the extra flexibility of a path based representation allows sounds to be merged across time as well as across timbres, giving a much richer design space. The path representation also allows access to common analysis-synthesis transformations such as time invariant pitch shifting and pitch invariant time stretching, allowing timbres to be adjusted far more flexibly than the point based approach, allowing the designer to squeeze or stretch a range of timbres to have a common duration. A path based approach can also be used to generate looped ambient sounds whose parameters can change without having to restart the sound and without introducing artifacts into the signal, which is difficult if not impossible with a point based representation.

3.1.1 Overview

The motivation for using timbre spaces for audio design is discussed in Section 3.2, where an overview of the problem is presented, followed by a short discussion of why timbre spaces can be used to address the problem posed. In this context a number of constraints are placed on the choice of analysis techniques. To explore audio design and output quality, there should be an easy way to synthesise sounds from the timbre space, either by inverting the stages used to generate the space, or by mapping a synthesiser directly in the space. The space should be able to model a wide range of sounds, including Auditory Icons and the orchestral sounds commonly used for Earcons. The signal analysis and dimensionality reduction techniques are further constrained such that the output of the signal analysis stage must be a suitable input for the dimensionality reduction stage.

These constraints are used to filter the available analysis techniques discussed in Section 3.3 which details the complete timbre space generation process and discusses how signal analysis is used in the automated spaces to transform the set of input sounds into a common form used to generate the similarity matrix. It then builds on the analysis discussion by specifying how timbre spaces are created from these components and comparing different statistical analyses for this process and the different types of timbre space that can be produced as a result. Section 3.4 looks at the existing work on perception and timbre spaces as it relates to the audio design goal.

In Section 3.5, using the timbre space to generate sound is discussed in the context of the synthesisers presented in the previous chapter. As each synthesiser can be represented as a model in the timbre space, the general construction and use of models in timbre spaces is also discussed.

Section 3.6 reviews the literature that has rejected timbre spaces for various reasons and discusses why the rejections are not appropriate to the current problem and how the remaining shortcomings of the technique can be addressed. Timbre spaces are not the only way to provide audio design capabilities and their advantages and disadvantages over other techniques are discussed here.

3.2 Perceptual mapping

Rolland and Pachet [79] and Gaver [14] have demonstrated audio systems that take a complex system (an FM synthesiser and a physical model respectively) and present an abstraction of that system that simplifies its interface to provide greater access to the system. In Rolland and Pachet's case, the simplification allows the user to modify synthesiser parameters in terms of how an instrument sounds rather than how the sound is produced. For Gaver, the abstraction models environmental sounds by converting a set of object properties as perceived in everyday listening to a set of parameters to an additive synthesiser.

Without this abstraction of sound towards the perceptual level, designers are left trying to develop sound at the signal or the musical level. Whilst many practitioners in the field are happy to work at such a level, ecological perception suggests that such an abstraction will open up design possibilities to practitioners who do not wish to work with such low-level structures.

To generalise these abstractions, a mapping is required that connects the human perceptual parameters (e.g. 'urgent' from Edworthy [30]) or object parameters (e.g. 'size' from Gaver [14]) to the synthesiser parameters. For an FM synthesis, these parameters affect the amplitude and frequency of the carrier and modulating waves. In a waveguide system these parameters will alter the configuration of the waveguides, the filters connecting them and the choice of output (e.g. displacement or velocity).

The set of these perceptual parameters will define a 'perceptual map' within the timbre space. It will define a set of points that are associated with perceptual information, such as a certain instrument or an adjective like 'urgent'. This map will be used to define and manipulate sounds according to these associations.

It should be noted that this definition varies slightly from the use of 'perceptual map' in the visualisation literature. Hermann and Ritter [54], for example, use the term to describe perceiving a visualisation through a haptic device. The perceptual map in this thesis is an audio relation to this haptic map, but adds human perceptual parameters alongside object properties as used in that paper.

3.2.1 Methods of mapping

There are two ways in which the mapping can be done. If a reasonable set of perceptual parameters is available, along with descriptions about how they relate to the waveforms, it is a simple matter of mapping those waveform features to the synthesis parameters that produce them.

Where these descriptions are unavailable, experiments may be performed to map those perceptual parameters directly to changes on the synthesiser by asking human participants to rate changes in the parameters. Not all synthesiser changes will result in meaningful changes in the perceptual parameters however so the full set of perceptual parameters may not be covered.

3.2.2 Map from perceptual to synthesis

To map from a perceptual set of parameters to a synthesis engine requires a strong set of perceptual parameters defined as a series of adjectives and the understanding of how those parameters relate to properties of a sound wave.

The advantage of this method is that it has a closer fit to human perception than an approach rooted in what the synthesiser can achieve. It is also easier to build the synthesiser as the perceptual model provides a strong set of requirements that the synthesiser must fit. However, there may be some requirements that it is hard for a synthesiser to meet, and may require a separate synthesiser model for each perceptual quality, with the final synthesiser needing to resolve any conflict where two required qualities of the sound require different manipulations. Trying to map the parameter 'loudness' for example, may be tricky as the phon curve (see Section 2.2.2) that maps loudness to wave amplitude is non-linear and depends on not only the amplitude of the wave, but also its frequency, so trying to create a louder, deeper (i.e. lower-frequency) sound would require the 'louder' parameter and the 'deeper' parameter to interact in a coherent fashion to produce the expected output.

The two ways to gather the adjectives to be used in this mapping are to ask the participants to describe the sounds in free-form, or to ask them to match the sounds to a prescribed list. Both of these techniques were used by Edworthy *et al.* [30]. The former was used to generate a useful list for the latter experiments, but for full effect, this requires the participants to perform musical listening which Gaver claims is less common than everyday listening [14]. This may indicate participants would need training which could introduce bias into the words produced without careful design.

The choice of list can be critical for the designer since different cultures may interpret the words differently. For example, 'urgency' for European listeners is often linked to a rise in pitch whereas Chinese speakers use higher pitches more often and would pick other cues for an urgent sound.

3.2.3 Map from synthesis to perceptual

To create a perceptual map based on the possibilities of the synthesiser requires a flexible synthesiser and a methodology for uncovering the mapping via human testing. This is useful where the output device is fixed or where no strong set of perceptual parameters is available, and so needs to be generated alongside the model. It is a more iterative approach since the synthesiser and mapping will have to be reconsidered after each set of perceptual trials in order to refine the model against an initially unknown mapping. The methodology would be based around asking volunteers to categorise sample sounds chosen to cover the range of synthesiser parameters and also to cover approximations to the perceptual parameters.

The advantage of this approach is that it is more likely to create a computationally efficient synthesis technique. It is less likely to cover every perceptual parameter, however, as the range will be limited by the output space of the chosen synthesiser.

Rolland and Pachet [79] developed a system for programming digital synthesisers without requiring expert knowledge. They describe a method for capturing the knowledge of synthesiser programmers such as "You can make a sound brighter by increasing the low-pass filter's cut-off frequency". They created a set of rules so that sounds can be defined in terms of how these adjectives should be applied to transform the sound. They concentrate on instrument sounds, and description at the perceptual level, and do not extend this up to an everyday listening level. Encoding this type of knowledge in a general platform for audio design, without requiring prior knowledge of the available timbres, provides a novel approach for design.

3.2.4 Choosing the direction for the mapping

Designers should not have to worry about the limitations of a synthesiser, so a mapping that is based on perceptual correlates and applies them to a synthesiser is the preferred option. Whilst it may be the case that not all perceptual parameters will map onto a set of synthesiser parameters, even when a complex, non-linear mapping such as the phon curve exists, the designers should be offered all perceptual parameters and decide for themselves the technical and cultural limitations they must work with.

As the following section will show, an automated process for developing a model that represents some concepts of perception has been developed. The human timbre space has been used as a platform for exploring human perception, although no similar investigations have been performed with automated timbre spaces. This research uses these investigations to construct a mapping in an automated timbre space upon which a perceptual model from Edworthy *et al.* [31] is built to allow intuitive manipulation of sound required.

3.3 Construction of timbre spaces

In 1977 Grey published a paper describing an experiment on perception and timbre [13]. In this experiment, he played a series of sounds to volunteers and asked them to rate how similar the sounds were. Grey then constructed a 3-dimensional space to represent the sounds such that the more similar sounds were rated, the closer together they were in the space. He reported that each axis represented a property of the sound. For example, he related the first axis to the spectral distribution of energy.

Expanding on this work, McAdams *et al.* [80], created a human timbre space model using a weighted distance measure based on assumptions about perception of timbre. This weighted distance was used to model differences between users. The resultant space was also weighted in order to model differences between the stimuli such as pitch and loudness. The combined use of these weights provides a stronger perceptual basis for the timbre space than Grey's model, and also allows discrimination between groups of subjects, which McAdams *et al.* used to differentiate subjects with and without musical training. Most recently, Terasawa *et al.* [44] have built a timbre space for speech where the analysis is constructed such that the resultant axes are perceptually relevant, by carefully choosing the analysis parameters used. Unfortunately, this paper was published too late to influence the choice of timbre space in this work.

Building on the human timbre space work, the two automated timbre spaces described here take a sound and create a Time-Frequency Representation (TFR) of this sound. The TFR is then treated as a set of points in space where each frequency is a dimension and each time-step represents a point, such that a sound is represented by a path in frequency space.

This space is then mapped onto a lower-dimensional space via a dimensionality reduction technique to produce a space related to that developed by Grey. The process of mapping from frequency (or another higher dimensional) space to timbre space is known as *Multi-Dimensional Scaling (MDS) Analysis*, and the tools used are derived from statistical models and machine learning research.

In 1997, Hourdin *et al.* [12] first demonstrated the technique, and compared their space with that of Grey, showing a close correlation between the two. They then used this space to drive a synthesis system [63]. The perceptual relevance of their automated timbre space approach was demonstrated in their first paper where they show that 2 of their 8 axes have a strong correlation with the first 2 axes in Grey's space.

In 1999, Kaminskyj [78] demonstrated that an alternative MDS analysis could be applied to produce a space more suited for separating the timbres of different instruments in order to identify distinct sound sources. In Kaminskyj's space, similar sounds occupy similar paths and regions of the space and this similarity allows his software to identify when sounds present in the timbre space are also present in a provided audio segment. The differences in the two papers show that there are many possible timbre spaces and each one has its own strengths and weaknesses for different tasks. The field is not yet mature enough to compare the strengths of different methods against different uses through the available literature.

The following two sections will look at the automated timbre spaces in more detail, firstly studying a selection of possible signal analysis techniques for producing the TFR used previously in automated timbre space and related fields, then by considering some MDS techniques including those used in human timbre spaces.

3.3.1 Signal analysis

Analysis-Synthesis

Analysis-synthesis is the general name for a class of techniques that analyse an input sound into an intermediate representation, such as a TFR, manipulate the sound via this representation then synthesise a new sound based on the modified representation [81]. The timbre space representation can be considered within this context, and so signal analysis techniques used for analysis-synthesis will be considered here.

Wright *et al.* [81] provide an overview of the current technology in this area. There is also a review of the TFRs and how to use them in a pair of papers by Masri *et al.* [16, 82]. A review of the TFR techniques is given in this Section.

Masri *et al.* describe the technique as analysis-resynthesis, whilst Wright uses the term analysis-synthesis. *Analysis-synthesis* (A/S) is the term used here to emphasise the intention to create novel sounds.

Two examples of uses of A/S from Wright *et al.* are time dilation and morphing. In time dilation, the frequency bands in the TFR are stretched over a longer time than the original signal, producing a sound that is longer than the original but at the same pitch. In morphing, also known as timbral interpolation, the parameters of two sounds are interpolated to create a novel sound containing some properties of each of the original sounds.

Sound Description Interchange Format (SDIF): As part of Wright *et al.*'s paper [81], the various A/S techniques are all compared by converting their TFRs into a general format developed by many of the authors and known as the Sound Description Interchange Format (*SDIF*) [83]. Specifications have been developed over a series of papers [84, 85, 86], culminating in an XML-based tool for interpreting this format.

An SDIF file consists of a number of frames, where each frame is identified by a header that gives its type and size, and where each frame represents an interval in time, most commonly one time step.

Each frame is represented by a matrix whose columns represent parameters and whose rows represent values. For example, in the standard TFR, the columns would represent magnitude and phase and each row would represent one frequency.

One method for encoding a timbre space in this file format could be to create a matrix for each time frame where each column is mapped to a dimension in the space. Each time frame would contain a single row representing a point in the space. Wright *et al.* have demonstrated that their format is flexible enough to store the results of all the analysis techniques listed here.

Analysis techniques

Signal analysis is the first stage in the automated timbre space construction. It is the process by which an input signal is converted into a TFR. There are many different methods for doing this, and good explanations for many of these techniques can be found in Roads [58, Part IV, Chapter 13] and in Masri *et al.* [16]. For the sake of consistency, the terms used here reflect their use in those references as opposed to their use within the individual techniques and papers referenced below.

In the TFR, the frequency axis is separated into a number of *frequency bins*, each of which covers a range of frequencies. This range is known as the *bandwidth* of the bin. In general, there will always be a trade-off between a smaller bandwidth which provides a more accurate frequency representation and a wider bandwidth which provides a more accurate time representation.

The analysis techniques described below include the ones already used in other papers on timbre spaces as well as the standard Fourier transform, which is the most common analysis technique. There is also a short discussion of two other techniques from Roads which may be useful in the development of a timbre space given the goal of designing perceptually relevant sounds.

In order to be useful for this generic audio interface design system prototype, the analysis techniques must possess the following qualities:

1. **Invertible.** For an analysis technique to be invertible, it must capture the entire signal and have an associated synthesis technique. This is useful for determining how much information is lost via the dimensionality reduction stage of the process. These techniques rarely offer perfect reproduction of complex timbres but they must provide some way of synthesising a signal in order to test the prototype.

2. **Generalises over a wide range of timbres.** Many analysis techniques can be optimised according the sound under analysis and some have automated tools to aid the optimisation. Any analysis technique which does this will generate a unique representation for each sound that makes it harder to compare with other sounds, reducing the effectiveness of the dimensionality reduction stage, and reducing the ability of the timbre space to incorporate new sounds.

3. **Suitable for percussive and stationary timbres.** In order to capture information about Auditory Icons, the analysis technique must be able to capture percussive sounds and transients in the signal. Most analysis techniques can do this to a degree, but some are optimised for stationary signals with no transients and are hence unsuitable for this application of timbre spaces. Similarly, analysis techniques suited to percussive signals may perform poorly over stationary signals.

4. **Output compatible with dimensionality reduction techniques.** The statistical analyses used for dimensionality reduction in existing automated timbre spaces all expect a rectangular matrix where the columns and rows are independent, and each row represents a data point in the input space. Whilst there are minor variations in how much importance is placed on independence in the input space, for example whether the input points are considered separate or are considered samples during the evolution of a system, none can analyse data that cannot be represented in an orthogonal input space.

Before the separate techniques are discussed, three properties common to many signal processing techniques are discussed. The stationarity assumption applies to most Fourier-based methods and states that the signal under analysis does not change over time. Since most instrument and environmental sounds violate this assumption, additions to the techniques that overcome this assumption must be considered. Most of these additions involve breaking down the signal into smaller segments, which requires the use of window functions to separate the segment under analysis from the rest of the signal. The limitations of this approach are due mainly to a trade-off between the frequency and the time resolution of the analysis data, so this is also discussed.

Stationarity assumption The Fourier analysis and its relatives all assume that the signal under analysis is infinite and constant. This presumption of a constant signal is known as the *Stationarity Assumption.* The practical implications mean that finite-time analysis windows are required to model changes in the sound, and unless each analysis window contains an exact number of wavelengths of a particular frequency, the energy of the signal due to that frequency will be spread across a number of frequency bins. This process is theoretically reversible when the frequency bins are mapped directly back to a signal using the magnitude and phase information for each bin. Practical limits due to sampling and quantisation of the signal to be analysed lead to smearing effects within and between frequency bins that can affect further analysis and limits the accuracy of the reproduction, since modifying the signal can introduce noise. Also, any changes of frequency within an analysis window will not be picked up in the analysis and this loss cannot be reversed, although phase information may aid detection of such changes.

The usual way to minimise the effect of the Stationarity Assumption within the Fourier Transform is to apply a window to the signal to remove the edge effects and then super-sample the signal so that more than one window covers each sample in the signal. The effect of the super-sampling is to match the transients more closely and avoid missing frequencies otherwise masked by the windowing. This process introduces noise into the signal around the small number of frequencies that are otherwise cleanly captured by the Fourier Transform, but vastly improves the response of the Fourier Transform to frequencies that do not fall cleanly on the centre of a frequency bin.

Window functions When a signal is broken down into a number of time windows, the windows are not guaranteed to contain an exact number of cycles of each frequency in the signal. The sine wave in Figure 3.1**A** has a value of zero at the start of the window and a value of -1 at the end. This discontinuity in the signal will produce artifacts in the output as the signal analysis tries to model this difference as a series of sines due to the stationarity assumption.

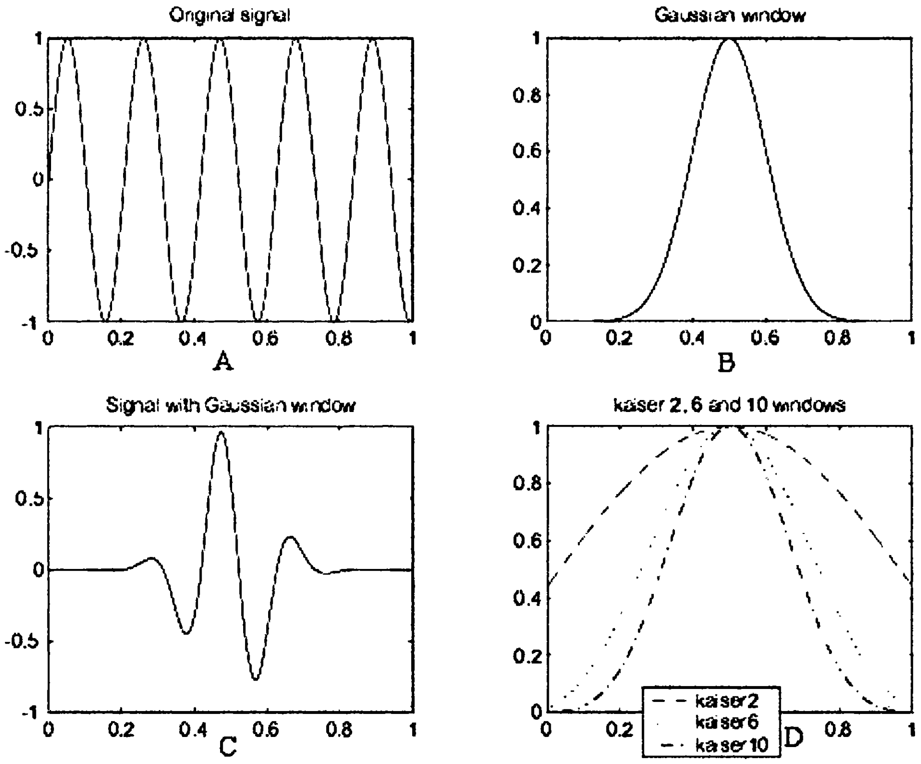


Figure 3.1: A simulated input signal and a selection of windows. **A** shows a portion of an infinite sine wave. **B** shows a Gaussian window of the same length as the signal portion in **A**. **C** shows the result of convolving **A** and **B**. **D** shows three Kaiser windows of the same length as **A** and **B** with varying values of alpha.

To reduce these artifacts, the signal being analysed is shaped by a window function so that the signal falls to zero at the boundary of the time-step, as shown in

Figure 3.1C. There are many types of windows that are used, each with its own tradeoffs between resolving the strong signals cleanly and resolving the minor details in the signal that can be masked by a stronger signal.

Two popular windows are the simple Gaussian window, based on the Gaussian distribution [87], and the more complex Kaiser window [88], which can be parameterised to fine-tune the trade-off. These windows can be seen on the right of Figure 3.1.

The major problem with windowing occurs when the signal is to be re-synthesised. Since the data was cut to zero, some of the original signal is inevitably lost in the analysis. To combat this, the windows super-sample the signal such that 2 or more windows sample the same space, each offset by a fraction of the window size, in order to capture all the detail in the signal. The signal can then be re-synthesised by overlapping the time steps in the output.

The choice of window hence has an important effect on the quality of the synthesis from an inverted TFR, as well as on the ability for a technique to generalise across a wide range of timbres. For this reason, a number of windows should be considered to ensure the lowest reproduction error.

Time-Frequency resolution In order to accurately gauge the frequencies present in a signal, the window must be wide enough to capture a full wavelength or more of each harmonic and the sampling rate must be high enough that each sample represents no more than half a wavelength. If, for example, the frequency of a sound is equal to the sampling frequency, the sampled sound will appear to be constant. The maximum resolvable frequency, at half the sampling frequency, is known as the Nyquist limit [89].

The default sampling frequency of CDs and many audio formats 44.1 kHz, since the human ear can detect frequencies up to 22kHz, i.e. half the sampling frequency. However, audio processing software commonly uses higher frequencies so that there are no artifacts when the signal is recorded at CD quality.

If the harmonics in a signal are known in advance, it is easy to calculate the minimum window size required to discriminate the harmonics and to reliably detect the fundamental frequency. Where these harmonics are not known in advance, a trade-off must be made between discriminating a large number of frequencies and accurately determining the onset and offset of transients in the sound.

The two common ways around this problem are demonstrated by the Wigner-Ville Distribution and Wavelet analysis, both discussed in more detail below. In the Wigner-Ville Distribution, derived independently by Wigner [90], Ville [91] and others, and similar analysis techniques, the time and frequency resolution are both improved, reducing the error due to averaging the signal, but at the cost of introducing noise terms in the analysis. In the Wigner-Ville, these terms appear in between pairs of frequencies in the analysed signal and are known as cross-terms, but other analyses introduce errors in other places.

The wavelet solution [92] to increase resolution relies on the fact that lower frequencies need a longer time window for analysis and so have a better frequency resolution at low frequencies and a better time resolution at high frequencies. This allows the Wavelets to capture low frequencies in the signal whilst still reliably detecting the transitions within a signal.

These two techniques both produce significantly more complex TFRs than the standard Fourier based techniques. The WVD produces a TFR with more noise, although its presence is predictable. The Wavelet produces a TFR that cannot be easily represented by a matrix. In order to use either of these techniques commonly found in Analysis-Synthesis therefore, a complex pre-processing step is required to generate a valid TFR matrix. The unknown complexity of this processing step precludes either technique from inclusion in a timbre space analysis. Due to its importance to the analysis-synthesis field however, the Wavelet technique is discussed further below as it has other advantages that may be useful to apply to timbre spaces.

Heterodyne filtering Heterodyne filtering [93] is the method used by Grey to develop his timbre space. Hourdin *et al.* used it in their analysis [12] in order to compare their results with his. Heterodyne filtering grew out of hardware-based auditory analysis from electronics work done in the 1960's. Its descendents are still used in devices such as electronic tuners for electric guitars and radios today, but has many flaws that limit its use in general frequency analysis.

In Heterodyne filtering, the input is convolved with a sine of a set frequency, and the output is then summed over a short period of time. If the input contains components at the same frequency as the sine wave, the overall output has a large magnitude, otherwise it is close to zero. The heterodyne filter approach is very sensitive to the frequency of the input, so some knowledge of the frequencies in the signal must be obtained in advance.

Since the filter can only have one reference frequency at a time, it is a slow process to generate a complete TFR from this analysis. In Hourdin's paper, the main advantage of heterodyne filtering is exploited, that the filter can track a signal over time. This results in a TFR where the frequencies are not constant over time.

The main disadvantage to this technique, according to Roads [58], is that it is poor at detecting fast changes in frequency or amplitude over time, where this change occurs over a range of 2% and in a time under 50ms. This means that the quickly decaying high frequency components of contact sounds modeled by Gaver, as well as many other percussive sounds, will be poorly represented in the TFR. This makes it unsuitable for this research.

Fourier The Fourier transform (FT) is such a popular spectrum analysis technique the terms "Fourier transform" and "spectrum analysis" are often used interchangeably, as noted in Masri *et al.* [82]. There are many books on the subject, and it is the default technique in most signal processing packages.

The basic concept of the FT is that every periodic signal can be composed from a set of infinitely repeating sine waves of varying frequencies, amplitudes and phases. The goal of the FT is to find a set of frequencies, amplitudes and phases for the sine waves that will recreate the input signal when summed.

The most commonly used implementation is known as the Fast Fourier Transform (FFT) [94], which recursively transforms half the signal then recombines the two results. The FFT is designed for use with discrete input signals since it works in the digital domain.

The FFT assumes the signal, like the waves that compose it, is infinite in time with constant frequencies and amplitude. For most real-world signals, this is a poor assumption. In order to analyse real-world sounds therefore, the input signal is broken up into small time steps such that the instantaneous frequency at a certain point in time can be determined. This process is known as the Short Time Fourier Transform (STFT) [95].

The output of the STFT is a series of magnitudes and phases for each frequency for each time step, and is often shown with time on the x-axis, frequency on the y-axis and magnitude as the intensity of the corresponding cell. This representation is known as the *spectrograph.*

The Fourier Transform is designed to be invertible as the theory on which it is based states that all sounds can be reproduced by an infinite number of sine waves. The fact that no synthesis can produce an infinite number of sine waves means that some noise is introduced into the re-synthesised signal, but there are many ways to reduce this. The Short-Time Fourier Transform has a frequency and time resolution determined solely by its parameters and the sample rate of the input, and so its output will be similar for all timbres, provided the sample rate is normalised before analysis. Adjusting the parameters to balance the time and frequency resolution of the analysis allows the STFT to model either the onset of a transient or its frequency content accurately. The STFT hence fulfills all three criteria listed above, as well as being the most common analysis technique used for analysis-synthesis, so it is considered here for comparison with this wider field.

Fractional FFT The original signal and the FT can be considered as two points on a continuum and the Fractional Fourier Transform (FRT) [96] allows the analysis to produce a result anywhere along that continuum. The FRT comes in two forms. In the linear form, an additional term is added to the exponent within the window analysis function. In this form, an inverse can only be guaranteed when this additional term is relatively prime to the analysis window length. In the quadratic form, the Fourier Transform itself is raised to the power of the additional term. When this term is 0, the result is the original signal, when the term is 1, the standard Fourier Transform is produced, a term of 2 or 3 respectively compute the original signal reflected in time, and the Fourier Transform of the original signal. With a term equal to 4, the result is the original signal. The effect of this term means that the quadratic form of the Fractional Fourier transform represents a rotation in the time-frequency plane by an angle of $\frac{\pi}{2}$ times the supplied term. As the quadratic form is the form most commonly used in signal analysis, it is the form considered here, and all references to FRT in this thesis discuss the quadratic form.

The advantage of the FRT is that it models linear changes in frequency over time very accurately, allowing it to more accurately model sounds such as those produced by string instruments whose frequencies drop over the course of a note due to changes in tension of the string. In the general case, across a wide range of sounds, linear changes in frequency are not very common, as most sounds either have constant frequencies or frequencies that experience an exponential decay. Due to this, the FRT adds complexity and time to the analysis but does not produce a noticeable effect on reproductive quality in the general case, when implemented within the timbre space framework. This was tested both by an informal listening test within the research group and by visual comparison of sounds produced by the Fractional and original FT based timbre spaces.

The Fractional Fourier Transform therefore appears to offer no advantage over the standard Fourier Transform in the listed criteria, but does have a greater computational cost. Consequently, it will not be considered further for the audio design system.

Discrete Cosine Transform The Discrete Cosine Transform (DCT) used by Terasawa *et al.* [44] in their timbre space, is a variation of the Fourier Transform that produces a real-valued result instead of the complex result of the standard Fourier. The Modified DCT (MDCT) optimises this analysis for non-stationary signals, by incorporating window functions into the analysis [97]. This model represents the transients in a signal as a wave in the frequency representation, which is easily modified, but loses the ability to accurately model stationary signals. Verma and Meng's model [97] uses the STFT to model the stationary portion of a sound and the MDCT to model the transients in the sound.

The addition of the MDCT to a timbre space analysis seems tempting, but as it also requires another model to cover the stationary signal, the analysis is complicated. Furthermore, Wang and Vilermo [98] state that the output of the MDCT is not orthogonal, which not only complicates its use for Analysis-Synthesis, but also invalidates one of the assumptions of the dimensionality reduction analyses. Whilst this is addressable, the extra computational cost for the entire analysis on top of the standard STFT prevents its implementation in this prototype.

CQT The Constant Q Transform (CQT) [99] is the method used by Kaminskyj [78] when constructing his automated timbre space for the purposes of automatic instrument identification. Like the STFT, the CQT divides the signal into time steps that are then windowed and analysed. The same issues of windowing apply to CQT and the STFT.

Unlike the STFT, which has constant bandwidth across the frequency spectrum, the CQT adjusts the width of the frequency bin such that the bandwidth increases with frequency, ensuring that the ratio between them, representing the energy in each frequency bin (defined by the letter Q) is constant. This produces a TFR whose frequency scale is logarithmic. Since the scale of musical pitch is also logarithmic, in that one octave difference in pitch corresponds to a doubling of frequency, the CQT seems a more natural model to use for musical listening. This logarithmic property is also useful for a timbre space approach since it is closer to the response of the human ear than the STFT's linear frequency scale.

The other great advantage to the logarithmic scale is that fewer frequency bins are required to cover the entire scale at adequate resolution. In the STFT, the bandwidth must be chosen such that there is adequate frequency resolution at low frequencies. To get quarter-tone resolution at 250Hz requires a bandwidth of 7Hz either side of the centre frequency, whereas quarter-tone resolution at 20kHz, which is close to the limit of human hearing, requires a bandwidth of 586Hz. The STFT will use the same bandwidth across the frequency range resulting in either a loss of resolution at low frequencies or too much resolution, and a lot of data that the ear cannot distinguish, at high frequencies. Since the CQT can adjust the bandwidth it can get a good resolution across the frequency range without generating a lot of data.

The CQT, unlike the STFT, is not guaranteed to be invertible. The CQT from Brown [3, 100] does have code for approximating an inverse, and its ease of implementation makes it a good choice for the research.

Formant Analysis / Phase Vocoders Formant analysis is often implemented as a post-processing step to the STFT, modifying the TFR produced. Although the STFT is the most common source of the TFR for this technique, other analysis techniques such as Auto-Regression (see below) have been used.

Formant analysis uses a process known as phase vocoding, and was first demonstrated by a talking robot at the 1936 World's Fair in New York City [101]. Formant analysis uses the phase vocoder to enhance the accuracy of the frequency bins across the representation. To do this, the phase of the signal across a time window is analysed and compared to the expected phase difference if the frequency of the harmonic under analysis was precisely at the central frequency of the bin. The variation can then be used to determine if the actual frequency is lower or higher than the analysis frequency and by how much.

To improve the results of the phase vocoder, the FT is often tweaked according to the estimated fundamental frequency of the input sound such that the expected harmonics of the sound will fall close to the central frequency of the bins, and the lowest frequency bin will contain the fundamental. This approach does have problems as not all sounds have an easily detectable fundamental frequency, particularly percussive sounds, or sounds where certain partials are missing, as found with some brass instruments.

Formant analysis is a popular technique in analysis-synthesis fields since it reduces a sound to its harmonics, and so reduces the amount of data required to store the sound. Often, the amplitude and frequency envelopes for each partial over time are simplified further to allow real-time processing and synthesis. Instead of recording the values for each time step, the general trend is approximated by a series of straight lines or other approximations, and only the parameters for those lines need to be stored.

Whilst the ability to track small variations in frequency and amplitude does seem appealing for this work, the resultant analysis data is not easily confined to a matrix suitable for multi-dimensional scaling since MDS assumes that all the columns of the input data are independent whereas the TFR produced by formant analysis links an amplitude column to a frequency column. There is no guarantee that MDS will preserve this connection.

This problem, coupled with the difficulty of modelling percussive sounds using this technique, suggest that this technique is not suited to the goals of this project. However, one popular alternative to the Fourier Transform used in this technique is known as Auto-Regression or Linear Predictive Coding analysis and does seem to offer a few advantages over the Fourier Transform.

Auto-Regression and Linear Predictive Coding In Auto-Regression (AR) analysis [70], a signal is compared with itself to find common patterns that repeat across the lifetime of the signal. A single sine-wave, for example, is simply one pattern that repeats with an period of 2π. In detecting the period of these patterns, AR seeks to uncover the wavelengths of any patterns in the signal. Once these are discovered, it is trivial to convert the wavelength in a sound signal to frequency data, provided the sampling rate is known.

The class of techniques under the banner of Autoregression Spectrum Analysis all seek to predict the future values of a signal by looking at past values. This prediction requires the algorithm to estimate filter parameters based on the signal history and the inverse of the resultant filters provide a measure of the frequencies in the sound. The filter parameters are often defined using linear regression, and these AR techniques are hence also known as Linear Predictive Coding (LPC) methods [70].

The filters are constructed with a noise model that allows the model to be split along harmonic and noisy components. The harmonic component closely fits the models produced by the Fourier based techniques, but by modelling the noise separately, the frequencies in the sound can be tracked more accurately during Formant Analysis, for example in Bailey and Cooper [102].

Where the spectra of a sound contains discontinuities or is a broad-band signal that is not very smooth, such as a percussive sound, Auto-Regressive Moving Average analysis (ARMA) [70] may provide a much better model than standard AR by including both the history of the signal and the history of the filters in its parameter estimations. In doing so however, there is a considerable computational cost.

Wavelets Wavelets [92] can be classed as a CQT approach as they also use a logarithmic frequency scale. Unlike the CQT approach described above however, wavelets have varying time resolution at different frequencies, such that when the frequency resolution is good (i.e. at low frequencies), the time resolution is poor and vice versa.

Pioneering work on wavelets for signal analysis was performed by Kronland-Martinet in the late 1980's and early 1990's [92, 103] and much of his work has applied the wavelet transform to Analysis-Synthesis, such as [71].

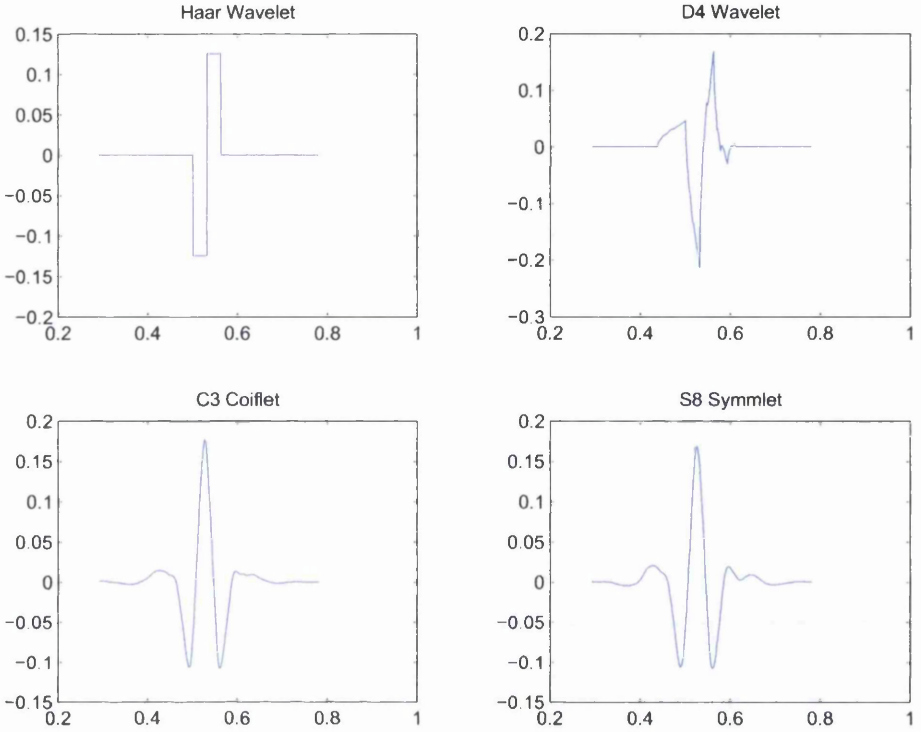


Figure 3.2: A selection of wavelets, taken from the WaveLab [104] package for Matlab. [105]

A wavelet is a function that is non-zero for some finite portion of space, and zero otherwise. The total area under the wavelet always sums to zero. When the length of a wavelet is modified, its frequency is adjusted to compensate such that the signal always contains a constant number of cycles.

Wavelet analysis is performed by taking a wavelet of the largest required time resolution and convolving this with the signal, performing the frequency analysis and the windowing in one step. The wavelet is then moved along the signal to create a time-magnitude graph for the lowest frequency, the wavelet is then shrunk and the process repeated with a smaller time step over the whole signal. In this way the complete TFR is calculated across the frequency range.

The wavelet transform has a lot of advantages over the STFT and CQT approaches due to the flexibility of the analysis, where each wavelet brings out different features of the sound, and new ones can be designed if there is a feature that is particularly important to the analysis.

The biggest problem to using the wavelet in the timbre space analysis is that the time resolution is not constant. Although this could be resolved by taking the smallest time resolution at the required level and repeating frequency values across larger time steps, this does not follow the way wavelets work and this situation is more reliably modelled by the CQT.

Since the wavelet has this problem, it will not currently be considered. If a better technique for handling the variable time resolution of the wavelet can be defined, possibly with an enhanced timbre space representation that can handle different time steps per frequency, and manipulate these intuitively, then the wavelet could prove to be a very useful and adaptable component in the system.

Human perception analysis In Roads, there is a section entitled 'Auditory Models' [58, Part IV, Chapter 13] that discusses two models that attempt to simulate the response of the human ear to sound. These models, the *cochleagram* and the *correlogram* both manipulate the TFR produced by, for example, the STFT, according to existing psychoacoustic models of the human hearing system. Since they both use an extra processing step, they will be slower than spectral analysis on its own, but if the extra processing is acceptable in the application, they could provide a useful extra step towards a complete perceptual model in the timbre space which should make it more intuitive.

The cochleagram adds a model of the neurons in the cochlea inside the ear, developed from work by auditory scientists. It is very sensitive to sound onsets although the frequency response is delayed for low frequency components due to the working of the ear so a strong onset is represented by a diagonal line in the cochleagram. Such a representation complicates the dimensionality reduction and resultant timbre space as the onset of transients in the sound will be spread across a number of points in the space, reducing the correlation between frequencies at the onset of a sound. This complication of the TFR requires a time-based dimensionality reduction technique for accurate modelling.

The correlogram adds a further level of processing beyond the cochleagram and shows the response of each point on the cochlea through time against the history at that point, in a process known as autocorrelation that measures how closely a signal matches its own past, which is the same process used in the auto-regression analysis described above. The correlogram is a 3D representation of the sound, unlike the standard 2D representations of standard Fourier based techniques, which makes it much harder to perform a multidimensional analysis on the data, which expects a matrix to be passed as the TFR into the dimensionality reduction stage, where the rows of the matrix represent the points and the columns represent the dimensions in frequency space. Whereas the TFRs produced by the other techniques discussed here can be said to represent a path through frequency space over time, where the axes of that frequency space are represented by the columns of the TFR matrix, the representation produced by the correlogram generates a history for each intersection in the Time-Frequency Representation, which is in itself a path through frequency space over time, which leads to an exponential growth in data over the TFRs produced by the other techniques. This complexity requires a more advanced dimensionality reduction than even the time-based representation needed for the cochleagram and prevents the correlogram from being a useful timbre space model within this framework.

Choice of analysis techniques for further investigation For the timbre space, the Heterodyne and CQT techniques have already been applied for generating a space, although the CQT space of Kaminskyj [78] was not used for synthesis. For comparison with existing analysis-synthesis, the Fourier Transform is considered. This provides an insight into how effective other, more computationally intensive, transforms might be, such as the Wigner-Ville or cochleagram. However, initial tests with one such technique, the Fractional Fourier Transform, show that this technique does not appear to offer any advantages over the standard Fourier analysis. The other techniques presented here, whilst used throughout the analysis-synthesis literature, do not fit into the stricter framework required of the timbre space. The Formant Analysis technique introduces too much dependence between dimensions in the TFR, whilst the Wavelet, cochleagram and correlogram techniques produce a TFR too complex for the matrix representation required for the second stage of the analysis.

Of all the common techniques used for analysis synthesis, only the STFT, the CQT and auto-regression appear to fit the criteria above of being invertible, generalisable to a variety of timbres including the percussive timbres of the Auditory Icons, and providing a TFR matrix suitable for input to a dimensionality reduction analysis as discussed in the next section.

3.3.2 Dimensionality reduction

The second stage of the automated timbre space is to take a matrix representing the TFR generated by the first stage and convert this into a more compact representation that finds the important characteristics of the data and sorts it to allow the less important data to be ignored.

If the TFR is constructed of columns representing frequencies and rows representing points in time, the TFR matrix can be thought of as a cloud of points in frequency space. The dimensionality reduction algorithms described here will find the directions in the frequency space that correspond with the most variance across the points and then rotate the space so that those directions become the axes of the new space. Since these dimensions are ordered by decreasing variance, most of the deviation between the sounds is captured by the first few dimensions. In Hourdin *et al.*'s space, for example, using 40 sounds, the first 5 dimensions captured 80% of the variation between points, and the first 8 captured 91%.

The resulting timbre space can be constructed with as many dimensions as the original frequency space, but by concentrating on just the first few dimensions, the data size, geometric complexity and associated run-time complexity can all be reduced without a great loss of useful information. Much of the following information is due to the review of these techniques by Roweis and Ghahramani [106] which also presents an algorithm capable of calculating all the techniques listed here, allowing easy comparison of the strengths and weaknesses of each one.

An example of the types of paths generated by the timbre space can be seen in figure 3.3. The flute sound in figure **A** can be seen to have a similar shape to some of the other paths in figure **B**, and these paths are other woodwind instruments. The flute path looks very different from the paths with very different timbres.

Implementation There are two main ways to implement the following techniques. The basic PCA method, which works for small data sets, is to create a matrix from the data and calculate variance statistics across that matrix. The eigenvalues of the covariance matrix can be ordered and the largest n of these eigenvalues and their corresponding m-dimensional eigenvectors are used to generate the $n \times m$ mapping between the data and the resultant space. The coverage of the space can be calculated by dividing the sum of the first n eigenvalues by the sum of all

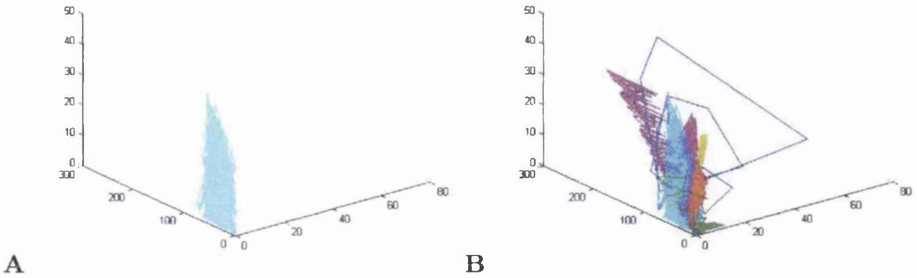


Figure 3.3: Examples of timbre space paths. In figure **A**, a single sound (a flute played flutter-tongued) is shown. In figure **B**, the sound from figure **A** is shown amongst other paths. Only the first 3 dimensions from the 8 used to construct the timbre space are shown. These figures were generated in MATLAB, using paths generated by the software that was developed as part of this thesis.

m eigenvalues and calculating the percentage. Small variations on the statistical matrix produced and the addition of a separate $m \times m$ or $n \times n$ matrix, before or after the statistics are calculated, provide the basis for the rest of the techniques here.

Where not enough memory exists for the entire matrix, or the data set is infinite, an approximation has to be used. This approximation takes the matrix and slowly iterates it toward a solution by calculating the errors between the current matrix and a statistical model calculated from it that calculates some measure of the deviation between the model and the original data. This is known as the Expectation Maximisation (EM) algorithm. These iterations continue until either a maximum number of iterations is reached or the error falls below a preset value, ϵ.

One example of this type of implementation, based on Roweis and Ghahramani [106], is the EM-Kalman implementation. This uses the above technique to implement Kalman filters [107], which are a time-variant statistical model. The EM-Kalman algorithm can be modified according to Roweis and Ghahramani to model PCA by restricting the model to a single time step and not using the Kalman noise model. This modified model will henceforth be known as *EM-PCA*.

The analysis technique used for this prototype for audio design should have the following properties:

1. **Easily invertible.** Whilst this is not strictly necessary for an audio design system that includes parameter maps for various synthesisers, it is useful whilst exploring the prototype as the effectiveness of the parameter map can be tested against the inverted output to provide some measure of the limits of quality that a parameter map can provide. None of the techniques listed here are directly invertible since the goal of dimensionality reduction is to remove data from the representation. What is important is that there is a method to convert the reduced representation back into a usable TFR for synthesis.

2. **Orthogonal bases.** The result of the analysis should be a timbre space whose axes are orthogonal. This prevents unexpected changes to the sound when points are adjusted along a single dimension. Without orthogonal bases, the timbre space is less predictable and hence less useful as a design tool.

3. **Compatibility with signal analysis techniques.** Since the output of the first stage of timbre space construction is a TFR matrix, the techniques must accept such a matrix as input. Other signal analysis techniques and other dimensionality reduction techniques are available but without an obvious way to connect the output of one to the input of the other, they cannot be used for timbre space construction.

In addition to these properties, all the techniques listed have the possibility to over- or under-fit the data. In the case of over-fitting, the timbre space will not generalise to timbres outside those used in its construction. In the case of under-fitting, the timbre space is too general and does not accurately capture the important features of the sounds used in its construction. Only some of the following models provide a way to test for over- and under-fitting during construction. If this cannot be tested during construction, the timbre space may not be optimal. None of the linear models presented here incorporate such a test.

PCA Principal Components Analysis (PCA) is the simplest of the dimensionality reduction techniques and the most susceptible to any noise in the input data. It is the technique used by Kaminskyj [78] in his timbre space paper, where the input sounds used contained very low noise levels due to the recording process.

For the timbre space in this research, it is assumed that noise is a minimal problem as all input sounds are generated from low-noise environments. For timbre spaces and data sets where this may be a problem, PCA will become unsuitable for the task and another more complex technique will be required.

The PCA technique calculates the covariance of all the input data as a matrix C, such that C_{ij} is the covariance between dimension i and dimension j[1]. PCA then calculates the eigenvalues and their corresponding eigenvectors of C, and orders these such that the largest eigenvalue comes first. The mapping then consists of taking the first m eigenvectors, which each have n dimensions, and composing the m × n matrix from this. The m-dimensional timbre space is generated by multiplying each n-dimensional point in the TFR by this new matrix. The PCA matrix can be inverted by the Moore-Penrose pseudo-inverse [108] to provide a n × m inverse mapping to generate a TFR from a path in timbre space.

The eigenvectors calculated in the PCA are orthogonal to each other by definition and the PCA can take any n × c matrix as input, so it can generate a timbre space from any c n-dimensional points. The sensitivity to noise is easily controlled by careful selection and processing of input sounds, and this technique has already

[1] or frequency i and frequency j, where $C_{ij} = \frac{1}{c}\sum_{k=1}^{c}(x_i^k - \bar{x}_i)(x_j^k - \bar{x}_j)$, c is the number of points in the input space, x^k is the k'th point in the input space, x_i is the i'th dimension of point x and $\bar{x}$ is the mean of all the points in the input space.

been applied to the generation of timbre spaces, making it an important technique to explore for the audio design framework.

FAC Factoral Analysis of Correspondences (FAC) is the technique used by Hourdin *et al.* [12], using the ANCORR program from Addad Software [109]. As this package is proprietary software, there is no further information about the technique, but the description in the paper fits the Roweis and Ghahramani [106] discussion of Factor Analysis, which has many similarities in structure and use, but differs in naming conventions.

Like PCA, Factor Analysis generates a translation matrix from n-dimensional to m-dimensional space, but also generates a set of *uniquenesses* in m-dimensional space which model the variation in the output space. These uniquenesses give the model a certain tolerance to noise in the input.

The major problem with factor analysis is that the translation matrix is highly sensitive to the rotation of the input data, so that if the input data is skewed in any way, such as a Doppler effect in the sound of an accelerating source, the translation matrix varies dramatically. If the magnitude in one dimension changes dramatically however (such as two data sets from microphones with different resonant frequencies), then the algorithm can re-scale that component to produce the same output.

Like PCA, this technique is invertible, has orthogonal bases and accepts a TFR as input, and has an advantage over PCA of being able to factor out noise owing to recording using a variety of input devices. Whilst this was important for Hourdin *et al.* who used both Grey and MUMS sounds for their space, it does not affect the current research which uses a single source for all sounds. For this reason, the prototype will not use FAC, although later iterations of the timbre space may find incorporating this technique useful for countering the effects of using multiple sources for timbre space generation.

SPCA Sensible Principal Component Analysis (SPCA) is presented by Roweis and Ghahramani [106] as a less flawed alternative to PCA. Like FAC, it also uses a separate matrix to model noise. Unlike FAC however, it is insensitive to the rotation of data but sensitive to the scaling. This is achieved by ensuring all the uniquenesses are equal.

The scaling of the data could be an important feature if the timbre space is designed to more closely match the acoustic response of the ear as measured by the phon scale. If this is the case, SPCA would preserve the integrity of this response much better than Factor Analysis. Due to this consideration, should a noise model be deemed appropriate in the timbre space mapping, SPCA should be preferred to Factor Analysis.

SPCA fits all three criteria listed above but is untested for the construction of timbre spaces. None of the analysis techniques chosen model effects such as loudness, which this method appears to be most suitable for. Whilst this technique is not used for this prototype, in order to preserve the integrity of the original sounds, it is worth considering if other signal analysis techniques are tested in a timbre space context.

Temporal methods Temporal methods consider the evolution of a system over time, creating two matrices for the transformation, one showing how each n-dimensional state evolves into the next, the other showing how the n-dimensional hidden state of the system maps to the visible m-dimensional output, as used in the static methods already discussed. Most temporal models also include matrices for the variation in the signal, similar to the uniquenesses found in the Factor Analysis approach. As these produce a point in a space which represents a temporal model and a mapping, these could be useful to produce a comparison with the work of

Grey [13]. The main focus of the project however is not absolute perceptual accuracy of the space but rather ease of use in creating sounds in a space that has some perceptual relevance.

Choice of scaling technique The advantages offered by paths, which allow flexible, time-dependent morphing into other sounds appear to outweigh any benefits that could be gained from a more accurate but less flexible approach that uses a point representation, for the same reasons an automated timbre space is used rather than a human space. Time-dependent techniques are most useful for simulating human timbre space behaviour using automated timbre spaces, although it is unclear how closely such a space would match a human timbre space.

The path based representations can be split into the FAC and SPCA techniques which model noise, and the PCA which doesn't. Given that the input sounds have been recorded in a low-noise environment, and that there exist signal analysis methods that separate the modelling of noise from the modelling of the signal, the PCA will be used for simplicity. The usefulness of the other techniques should be considered for noisy inputs and alternative signal analysis techniques but will not be used for this prototype in order to limit the number of independent variables available for the analysis of different timbre spaces.

3.4 Perception in timbre spaces

Using a human perceptual space, Rumelhart and Abrahamson [110] created a parallelogram model to demonstrate that given a set of five points, A,B,C,D',D" paired such that A is paired with B and C is paired alternately with D' and D" (and possibly other points), participants can reliably tell which of the distances $\overline{CD'}$ or $\overline{CD''}$ is closest to the distance $\overline{AB}$. The space they used was a classification of animals, which was constructed from a Multi-Dimensional Scaling analysis based on participants' similarity ratings between animals, and comparisons were of the form "Cat is to lion as dog is to ?" with possible answers of "wolf" or "cow".

McAdams and Cunibile [19] extended this demonstration to confirm that listeners can perceive constant changes in timbre within a point-based human timbre space. Given a vector **AB** in the timbre space, listeners are able to reliably determine whether vector **CD** or **CE** is closest in direction and in magnitude to **AB**, although there is a variation in performance between musicians and non-musicians. Their results suggest that not only is distance between timbres detectable, but also the direction from one timbre to another can also be detected.

This parallelogram model seems to offer a powerful new design concept that is unique to timbre spaces. Whereas existing analysis-synthesis models can apply morphing to change a sound according to a remote target, the parallelogram model can use a vector to adjust the sound in a way that does not affect its overall character. A graphical analogy can be seen by imagining a picture of a printer. To show urgent action is required, a morphing model would fade out that picture into one of an exclamation mark, losing the shape and colour of the original picture. The parallelogram model could adjust merely the colour, reddening the printer to indicate an urgent condition.

McAdams and Cunibile provide no indication of a perceptual description of a vector **AB**, or whether the parallelogram model applies to automated timbre spaces. As part of the investigation presented in this thesis the effectiveness of a perceptual vector in an automated timbre space is evaluated. Such a description allows us to name relationships between sounds and explore sound using a ecological rather than an analytical model since these relationships define the changes in texture between sounds.

To test their effectiveness, this *vector* model of perceptual transforms using parallelograms can be compared against a more traditional *gravity* model of transforms based on morphing between sounds (or "pulling" a sound towards another). In the vector model, we define a vector **AB** between the two extremes of a perceptual parameter (e.g. alarm to ambient, urgent to non-urgent), then apply that vector to a sample sound to adjust the value of that perceptual parameter. This model is exclusive to timbre spaces. In the gravity model, the sound is simply morphed in the direction of the perceptual extreme we wish to capture. This model equates with the standard process of mixing or interpolating two sounds in standard analysis-synthesis. More information on the comparisons can be seen in Chapter 6 where the perception of the two models is compared in a controlled study.

3.5 Synthesis from timbre spaces

In order to use the timbre space in a design platform for sound, it must be capable of interfacing with a synthesiser to generate output. Given a path or a point in the space it must be possible to generate a set of parameters for a given synthesiser. This 'parameter map' should be fast to calculate so that the mapping of a point from timbre space to parameter space and the subsequent generation of a synthesised sound from parameter space can be calculated within the time step of the timbre space path for real-time synthesis to be possible.

This section will first look at synthesis by inverting the dimensionality reduction mapping, which will create a baseline against which other techniques can be tested for reproduction quality and speed. The general rules to map to a variety of generic synthesisers are then discussed as well as a number of synthesisers these can be applied to.

Several synthesisers are discussed in the context of what they can do to extend the timbre space. Although only a limited number can be tested within the prototype, a number of others are discussed since their addition to the timbre space will increase the design capabilities of the system. The synthesisers chosen to be implemented are the additive synthesiser, which is used for the inverse mapping, and the FM and Auditory Icon synthesisers in order to demonstrate how the parameter map can be used to extend the capabilities of the timbre space. The FM synthesiser is chosen as a fast, efficient synthesiser that can produce a wide range of sounds from a small range of inputs, but is usually hard to parameterise. The Auditory Icon synthesiser is chosen to investigate the effectiveness of the timbre space in modelling existing interface designs, and as an example of a set of physical models.

3.5.1 Additive synthesis

For additive synthesis, all that is required is to reverse the mapping from the TFR to the timbre space, which is a matter of finding an inverse to the matrix produced by the MDS analysis. As the original matrix is unlikely to be square, as that would be a mapping that does not reduce dimensionality, the inverse will not be perfect. The information lost in the reduction of dimensionality cannot be recovered, but the MDS technique aims to ensure that the lost data is the least important.

Once an inverse is found, applying it to a timbre space path will result in a TFR. Provided the centre frequency of each bin in the TFR is stored (which must be the same across all sounds for the timbre space to be valid), it is a simple matter to feed the amplitude and frequency for each frequency bin into an additive synthesiser to produce an output.

Alternatively, where an inverse to the analysis exists, as it does for the STFT, the inverse can be applied to the TFR. Depending on the complexity of the TFR, this may be faster than additive synthesis. However, for the purposes of generality only the additive synthesis method is used.

Since the inverse mapping is simply an inverse of a matrix, the mapping in this case is guaranteed to be linear and will have the run-time complexity of a standard matrix multiplication for each point (i.e. $O(m \times n)$ where m is the number of rows and n is the number of columns in the matrix). The computation time of the inverse matrix is negligible as this needs only to be calculated once per timbre space.

The additive synthesis is the simplest method of generating output from an automated timbre space, and was used by Hourdin *et al.* [63] to generate sound from their timbre space. It also demonstrates the power of the automated timbre space since the conversion of a human timbre space point into a sound is a complicated process, requiring extra processing to extend the sound to the desired length. No work has been uncovered that applies this synthesis model to human timbre spaces.

3.5.2 Non-linear mapping

For general synthesisers, the TFR will not be an obvious source of parameters so a parameter map must be defined that will transform any point in the timbre space to a set of synthesiser parameters. This result can also be thought of as a point in the synthesiser's *parameter space.* The synthesiser will be represented by a multi-dimensional mesh inside the timbre space and the *parameter map* between the timbre and parameter spaces is not guaranteed to be linear. For example, the FM synthesiser has extremely non-linear behaviour in terms of the modulating index. The mesh can be as simple as a cloud of points within the space but also includes fitting grids or other functions into the space to enable interpolation between known points and faster searching, although there may be an increased error due to this interpolation. If such a grid is used, the grid is not guaranteed to be linear or regular, so a very general mapping technique would have to be considered that could handle a complex grid.

There are two methods that can be used for this mapping, the simpler of the two methods is a proximity detection method, i.e. find the closest point in the mesh to the target point and return it. There are limited opportunities to add interpolation techniques to this method, but it is fairly simple and there are many algorithms available. If a more complex mesh such as a set of Bezier patches is used then this technique can be used to find a set of patches to test against rather than a set of points.

A more complex mapping can be constructed in a similar way to the MDS scaling techniques used to generate the map between frequency space and timbre space. As the required mapping may be non-linear however, the same techniques cannot be used. Two methods that do allow non-linear mapping are considered, but speed complexity prevents their use in a real-time synthesis in most modern computing environments.

Proximity detection

Proximity Detection, or closest point, methods simply try to find the object in the mesh that is closest to a given input point. Although the mesh is generally constructed of points, it can also be constructed of structures such as Bezier patches, where the input point would be compared against a series of convex hulls[2] of those patches, and the closest Bezier patch could then be tested against the search point to generate a more accurate result point.

The brute force method of proximity detection is to compare the target point against every point in the mesh. Smid [111] claims that there is no more efficient algorithm than brute force in the general case, but this is not proven and the upper bound

[2] A convex hull being a set of points guaranteed to completely contain the object.

to the search time is currently an open problem. The techniques presented here perform significantly better than brute force on average.

The brute force method has a time complexity that grows linearly with both the number of points in the mesh and the dimensionality of the space. By eliminating large sets of points to test, the expected time complexity for defined meshes can be reduced significantly, but the elimination process is not trivial, and elimination is impossible if all test points are an equal distance from the search point within the elimination margin of error. As is shown below, there are methods to reduce the complexity to be constant in terms of the number of dimensions, but this still leaves an algorithm with a worst case that is linear in terms of the number of points, as Smid warns.

The two methods for doing this discussed here are the Loose Octree by Ulrich [112], which is extended here beyond his three-dimensional model, and the k-d tree, developed by Bentley [113] and extended by Freidman, Bentley and Finkel [114] to add proximity detection capabilities.

In order to use proximity detection to map between two spaces, the point cloud for the search space (i.e. the timbre space) must be tagged with data from the output space (i.e. the parameter space) such that once the nearest point is returned, the algorithm will output the point in the output space that maps to the returned point in the input space. To do this, a set of points representing a series of synthesiser parameters are generated. The sounds produced by these parameters are then analysed to create a point in timbre space. This point is then tagged with the synthesiser parameters used to create it, and the collection of these points becomes the point cloud used to conduct the search.

Loose Octree The high-dimensional analogue of the binary tree is known as the Octree. Each node in the tree represents a hypercube. Each node has 2^d children, where d is the number of dimensions of the tree. The edges of each child are exactly half the length of the edges of its parent, and each child's volume is 2^d times smaller than the parent's volume. The children are arranged in order so that together they cover the full hypercube of the current node. In 3D, the situation looks like Figure 3.4. The root node is divided into 8 children numbered 0 to 7. Child 1 in the figure is further subdivided, and child 6 is obscured.

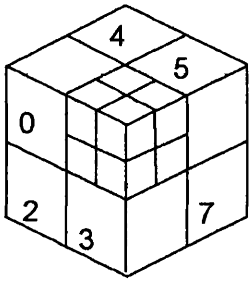


Figure 3.4: Children of an Octree node

Each node in the Octree contains either a set of Octree children or a set of objects contained by the Octree. For the proximity detection case, each object is a single point from the mesh. The Octree will subdivide the space into children and grandchildren and so on until either a maximum depth is reached or the number of objects in a node falls below some minimum value.

To search the Octree, each child node is tested to see which one contains the test point. The process is repeated on each child node until a leaf is reached. The leaf then returns its list of objects, hence providing an constant upper bound on the number of brute force tests required. The algorithm hence has a best-case time complexity logarithmic in the number of points in the cloud, but still linear to the number of dimensions.

In proximity detection, there is a problem when the test point and its closest neighbour lie opposite sides of a node boundary, as the closest point will not be returned. To solve this, a concept can be borrowed from Loose Octrees [112]. In a Loose Octree, each child is bigger than its Octree equivalent, and overlaps the other children. In the original paper, this was to prevent objects half the size of the parent node having to be split to fit into a child node. In the closest point version, the Octree is constructed the same way, but during searching, each node has a catchment area which extends beyond its dimensions, allowing it to cover the same volume as its parent, but with a different centre.

Despite the good search complexity, there can be problems with this approach, particularly when the data has a non-random distribution. The first major problem with this approach is the size of the data structure that holds the tree, and the time taken to construct it, both of which increase exponentially with dimensionality and with the number of points, since to accommodate more points, the tree requires a greater depth, which requires more children.

If the data are distributed in many small clumps across the space, it is likely that many hyper-cubes will contain no points, and others will be greatly subdivided. If the query point is in one of the hypercubes that has no points, the closest point cannot be returned. To deal with this case, we must return points from somewhere, so we must either return a random set of objects or return the entire set of objects and perform a brute force test.

k-d Tree The k-d tree, developed by Bentley [113], was extended by Freidman, Bentley and Finkel [114] to allow it to perform proximity detection.

The k-d tree, like the Octree, subdivides the space between a set of children. Unlike the Octree, each parent only has two children. The division point is not fixed at half the size of the parent as in the Octree. The k-d tree chooses a split point that divides the objects into two equal sections. It does this by finding the dimension of greatest variance and then placing the dividing hyperplane across that dimension such that it passes through the median-valued point along that dimension.

The tree hence sacrifices some simplicity to adapt to the distribution of data allowing it to avoid some of the problems of the Octree. The tree still suffers from the same boundary problem as the Octree however, where a test point and its closest neighbour may lie just on opposite sides of a boundary.

The k-d tree has been extended, as in Egas *et al.* [115] for example, such that not only does the search have a logarithmic complexity in terms of the number of points, but it also has a linear complexity in terms of the number of dimensions. This is a product of the subdivision technique, but also of an efficient way to calculate the divisions that is independent of the number of dimensions they have. Egas *et al.*'s paper also suggests a solution to the boundary problem by calculating the near and far boundaries of each node as the tree is descended. If the closest point in the current node is further than any near point of another node, then that node must also be checked. There are obvious cases in which this will be no better than the brute force approach, such as where the center of a sphere is compared against points on that sphere, but as the Octree has no trapdoor for such a case, the k-d tree can be considered at least as good as the Octree in the worst case.

The space complexity of the tree itself is dependent only on the number of objects since the number of children of a node is independent of the dimensionality. The construction of a k-d tree is not only faster than the Octree because it is independent of the dimensionality, but the k-d tree ends up more balanced than the Octree due to how the nodes are divided. This balancing means that searching too can be faster. For these reasons, and the increased accuracy due to Egas *et al.*'s solution to the boundary problem, the k-d tree has been chosen as the proximity detection method to implement.

The implementation of the k-d tree used for this work can be found in the ANN package by Mount and Arya [4], based on their (and others) work on optimising the k-d tree [116].

Dimensionality reduction

Dimensionality reduction approaches as used in the construction of the timbre space can also be used to create a mapping between the timbre space and the parameter space. As the mappings are non-linear however, the analysis techniques required are more complex.

kPCA In kernel PCA (kPCA) [117], the non-linear data set is mapped into a higher dimensional space where it is linear. This transformed data set is used as the input to the standard PCA algorithm. In practice, the non-linear mapping function can be implicitly embedded in the PCA analysis by implementing it in the form of a *kernel*, which calculates the variance over the higher-dimensional points directly from the non-linear data without having to project them into the higher-dimensional space. The use of a kernel gives the algorithm access to spaces that are otherwise computationally intractable, such as a space with infinite dimensions. Kernels can be arbitrarily combined with each other and scaled to best fit the input data. This flexibility allows the algorithm to adapt to any input, but adds to the complexity, particularly when the format of the input data is not known in advance, as is the case with synthesiser meshes. The kPCA algorithm, like the PCA, can be implemented using Expectation Maximisation (EM) [118] allowing it to avoid memory problems, as described in Section 3.3.2.

SOM The Self-Organising Map (SOM) by Kohonen and Oja [119] was developed as a visualisation technique and is designed to work best when the output space is 2 or 3 dimensions. The initial iteration starts with a regular grid representing the output space. The grid can be formed from any regular shape but squares (or cubes) and hexagons are the most common, with hexagons popular for 2D visualisations.

In each iteration, each data point is assigned its nearest grid point or *cell*. The cell location is then updated to better contain the points assigned to it, but it also constrained to remain close to its neighbouring cells. The next iteration will then assign the data points to the new cell positions. The algorithm completes after a set number of iterations or when the maximum error between a data point and its cell falls below a pre-defined value. This is the same EM algorithm used for calculating the linear dimensionality reduction matrices.

The SOM is not guaranteed to find the best fit for the data since there can be situations where the algorithm gets stuck in a sub-optimal fit but the errors are too small for the cells or points to be reassigned to make a better fit.

Choice

Due to speed constraints, proximity detection will be used as it has a much more predictable upper bound for searching. Between the two methods, the k-d tree has a much lower upper bound both in construction and search due to its dimension-agnostic approach to the mapping.

Since the system is left open to allow new synthesisers to be added at a later date, the simplicity and speed of construction of the k-d tree is preferred over the dimensionality reduction methods. The system will however be kept flexible so that other techniques can be added where a faster mapping is possible. For example, the additive synthesis mapping is simply the inverse of the timbre space construction and so involves a single matrix multiplication per point to generate the parameters for the synthesiser.

3.5.3 Synthesiser choice

For the experiments comparing timbre spaces, fast synthesis is not a top priority and the simplicity of the additive synthesis approach means that the timbre spaces can be compared without the computational complexity of analysing a parameter set and generating a non-linear mapping to a synthesiser. For any given timbre space, the choice is not as clear but where speed is not an issue, the additive synthesis approach is closer to an inverse of the analysis and so allows a more direct comparison between input and output sounds. For cases where the additive synthesis method in unsuitable, the k-d tree non-linear mapping has a complexity which is linear in the number of points, and it can be tuned to give a trade-off between speed of mapping and resolution of parameter space allowing it to adapt to the limitations imposed by its environment, so the mapping can be customised to a range of devices and uses. The k-d tree is hence chosen for non-linear mapping. Demonstrating this technique also shows how new models can be incorporated into the timbre space.

Whilst the experiments all use sounds generated via the additive synthesis technique, the non-linear mapping is used to demonstrate the extensibility of the timbre space and the possibility of real-time synthesis of complex sounds from the space. The FM synthesiser and the Auditory Icon model both require a non-linear mapping.

3.6 Alternatives to timbre spaces

The timbre space is not the only model that can be used to explore the interaction of timbre and perception, and is also not the only method that can be used to combine Auditory Icons and Earcons. This section discusses some of the problems with and objections to timbre spaces to show that amongst these alternatives, the timbre space is a good choice of technique for a proof of concept system that does explore these intersections.

In their paper on granular synthesis, Keller and Berger [72] list a number of reasons why timbre spaces are problematic for modelling environmental sounds, and they suggest granular synthesis is a better approach. The next section examines each of their points in turn to show that the timbre space approach is valid, even though the synthesis of environmental sounds is a prime goal. It should also be noted that the paper reports "To date, no systematic work has been done to establish a perceptual classification of granular sounds." [72, p3]. Cook [120] has demonstrated some work on applying everyday and perceptual models to granular synthesis, but there is still no evidence of work to provide a systematic perceptual classification for granular synthesis. This suggests that timbre spaces which can encode perceptual information are more useful as a design tool than granular synthesis, although a number of the objections listed in the paper should be addressed to show that a timbre space is useful for sound design.

3.6.1 Suitability of timbre spaces

Keller and Berger[72] discuss the assumptions Grey made in his timbre space and seek to show they are invalid. The assumptions, as listed in their paper, are as follows:

> "(1) timbre space dimensions should be orthogonal, i.e. changes on

> one dimension should not affect the other dimensions; (2) the distance between samples A and B has to be equivalent to the distance between B and A; (3) the topography of the timbre space must be independent from the number and type of classes it contains." [72, p2]

In the automated timbre spaces created by Hourdin *et al.* [12], the Multidimensional Scaling Analysis is specifically designed to ensure the output dimensions are orthogonal to each other such that a change along one dimension has a well-defined effect on the sound signal that is separate from any change on any other dimension.

It should be noted, however, that Edworthy *et al.* [30] have shown that perception of sound is not orthogonal and changes in a sound concerning one perceptual construct, such as urgency, can affect our response to another perceptual construct, such as speed. This suggests that the application of a perceptual map to a timbre space will be more complex than a rotation of axes since the perceptual constructs in the perceptual map will not be orthogonal. This problem is not unique to timbre spaces, so any audio design system must be able to take such interactions into account.

The second point is supported by the parallelogram model [19]. The claim states that two sounds in the space must sound the same timbral distance apart no matter which is played first. In this model, listeners can reliably determine both the distance and the direction between two timbres. Moreover, timbres an equal distance apart in space are reliably detected even when the direction of the comparison is reversed. This result is statistically significant for both musicians and non-musicians. In other words, given two points A and B, listeners can reliably detect not only that the distance $\overline{AB}$ is equal to the distance $\overline{BA}$ but that the direction of $\mathbf{AB}$ is the complement of the direction $\mathbf{BA}$.

For points that fall on the line between A and B, Keller and Berger claim that the timbre distance is not constant since experiments show that when played sounds starting from A and ending at B, participants classify the sounds as being more like A for over half the given sounds, but when they are played from B to A, participants classify the sounds as more like B for over half of the given sounds. The argument given by Grey [13] to explain this effect notes that the auditory system tends to prefer to classify sounds as closer to what has already been heard, leading to a hysteresis effect when morphing between two sounds. It is this effect which is cited by Keller and Berger in support of the second point. This is a similar effect to the Persistence of Vision phenomenon in the visual system, an example of the phenomenon of "set" from psychology [121]. This philosophy claims that humans will tend to classify their knowledge into sets or groups and often find it hard to classify anything new into a different set without additional cognitive effort. This is said to be closely connected with the pattern-recognition abilities of humans. The phenomenon also appears to have close similarity with the concepts of audio continuity and timbral grouping discussed in Deutsch [41].

This similar effect in the visual system and the related audio phenomena would tend to suggest that the audio hysteresis problem is a factor of human perception rather than of the timbre space. McAdams *et al.*'s results on the parallelogram model appear to support this assumption since they found no difference in the timbre distance when distance was reversed.

Keller also mentions a previous oral discussion concerning bias in timbre spaces. Specifically, that timbre spaces are better at representing sounds similar to those used to generate the space. Although no evidence is presented for this, it does sound like a reasonable assumption. To test this, the reproductive quality for sounds not in the timbre space will be determined. This evaluation is discussed in Chapter 5. Keller states in particular that timbre spaces exclude impulse and continuous sounds as these are not represented by natural instruments.

Keller backs up the assertion that timbre spaces have a limited timbral palette by citing a paper by Grantham [122] that states the timbre, intensity and phase of a source are affected greatly by the environment and are used to determine the spatial location of a sound. This is a valid point for situated sounds. Chapter 2 makes the distinction between single sources and multiple, situated sources for exactly this reason. A single source will produce the same timbre no matter what type of room it is in or what other sound sources are nearby. In the case of a violin, the size of room and type of nearby instruments do not affect how the string or cavity vibrate. They do affect the passage of that sound to the listener, but it would be inefficient for each instrument to model possible interactions with the environment, so the environment modelling is left to a higher level of control that can adjust the parameters in and the sound wave out of a single source depending on the environment containing the sound source.

Timbre spaces are a poor model for impulse sounds for other reasons due to the time-frequency resolution trade-off at the heart of Fourier analysis. In order for a timbre space to accurately model the harmonics of a sound, a large time window is needed to get a good frequency resolution, and this prevents accurate modelling of transients in the sound.

A solution could be to build a timbre space specifically for impulse sounds and ignore harmonic sounds altogether by focusing on a sharper time resolution. This approach limits the ability of timbre spaces to be used for Earcon design however so is unsuitable for the stated purpose of this research. The design system could implement a range of these spaces and choose between them according to the timbral range required, using a percussive space for Auditory Icons, for example. This would require modifying the perceptual map and the synthesiser map for each individual space and finding some way to choose which space to use and some way of morphing between sounds that exist in different spaces.

Tuning the analysis for each sound would add complexity to the timbre space and prevent easy combination of sounds. A compromise would build a timbre space with a good frequency resolution but allow each point on the timbre space path to represent a varying amount of time. Most sounds would be modelled directly by the analysis used to construct the timbre space but impulse sounds could be modelled at a much higher time resolution, requiring a post-processing step to redistribute the lower number of frequencies among the frequencies in the initial analysis prior to mapping down to the timbre space dimensions. In the absence of *a priori* knowledge about the best analysis resolution for each sound, a compromise resolution should be chosen between the extremes of too low time resolution and too low frequency resolution. The ability to adjust the time length of each point of the timbre path, taken from analysis-synthesis techniques, is a useful addition to the timbre space that can be useful for data compression via run-length encoding[3] resolution tuning scheme described above provided an analysis technique can be found that can tune its parameters to produce the closest match to the original sound. Unfortunately, whilst there exists automated measures for audio quality such as *PEAQ* (Perceptual Evaluation of Audio Quality) [123], the standard is not complete [124] so cannot currently be used to train such an analysis.

3.6.2 Alternative technologies for audio design

The goal of this research is to create a representation of audio that provides more design options than are currently available. Section 3.2 proposed that an intermediate representation that abstracts the sound into a more perceptually relevant form provides such an abstraction. The timbre space is not the only such abstraction that can be used. Here a variety of alternatives will be discussed.

[3] i.e. encoding a long list of identical values by recording just the value and the number of repeats.

The abstraction of FM synthesis has been described by Rolland and Pachet [79] to describe instrument rather than synthesiser properties and by Delprat *et al.* [125] towards reproduction of input sounds by processing the TFR of the sound to be recreated. The generation of physical modelling parameters has been demonstrated by Guillemain *et al.* [126]. As there is a crossover in authors between this and Delprat *et al.*, similar motivation and techniques are used in both papers, suggesting a more general model of synthesisers is possible.

Whilst these papers demonstrate that parameter mapping from a TFR to a synthesiser is possible, none have an intermediate model with an abstraction at the level of Edworthy *et al.* [30] which uses adjectives to describe sound. Unlike the timbre space approach, none of these uses a generalised model for sound manipulation and generation, preferring instead to build the model up from the synthesiser instead. This choice fixes these techniques to a single synthesis technique and a limited set of transformations.

The Timbre Engine [64] is based on the Timbre Model [127]. The Timbre Model uses the Fourier transform in a similar way to the timbre space to break the signal into its component frequencies. Post-processing is then performed on this representation. Parallel streams of processing extract different pieces of information in order to build a perceptual model of the input sound based on the parameters defined by McAdams *et al.* [80].

This representation is cleaned, removing noise from the components. This representation is used in the Timbre Model for sound classification and sound morphing between timbres. In the Timbre Engine, this representation is presented to the user as an interactive interface that allows manipulation of the sound.

The model produced is similar to the human timbre spaces of McAdams *et al.* and Grey, and can be thought of as a way to generate such spaces algorithmically with pre-defined axes. This similarity means that each sound is represented by a single point, as the parameters affect the entire length of the sound. The perceptual space developed by Terasawa *et al.* [44] also exhibits this type of parameterisation but their axes are carefully chosen to have some perceptual significance.

For sound reproduction, the clean signal is augmented by a noisy signal that models the noise that was removed during the analysis. This process is common to other analysis-synthesis models such as Stochastic Modelling Synthesis [128] or Adaptive Analysis [129]. The clean signal is modelled by parameters such as the spectral envelope, frequency deviations, and irregularity control as well as speed of attack and decay. The noisy signal is defined in terms of shimmer (noise affecting the amplitude of the partial) and jitter (noise affecting the frequency of the partial).

The parameters are presented to the user as a series of graphs, where each graph shows the value of a parameter across all the partials in the sound. The interface allows adjustment of these parameters and real-time synthesis of monophonic, mono-timbral sounds using an additive synthesis model.

The abstraction into the timbral domain provided by this system allows greater control over the sound properties than other analysis/synthesis techniques. Unlike the timbre space however, the interface and parameterisation are focussed on a single synthesis model. This situates the synthesis in a specific point on the trade-off between signal complexity and speed. In addition, the interface is not suited to adding further models of perception, such as those defined for timbre spaces by McAdams and Cunibile [19] or defined for signals by Edworthy *et al.* [30].

3.6.3 Discussion of alternatives

Keller and Berger compared granular synthesis to timbre spaces and raised a number of interesting points about the suitability of the timbre space for audio design. Since their model has also been used for Auditory Icons, it is an important alternative. In order to demonstrate the effectiveness of the space, their concerns should be addressed, and the advantage that the timbre space offers over their solution, namely the perceptual map, should be considered. The experiments performed to validate the timbre space, described later in this thesis, reflect these concerns.

The other alternatives listed offer other intermediate models for an audio design system that can be used to represent and manipulate sounds. Whilst most of these have a point-based representation of a sound, limiting the expressibility of the representation, they do have a number of ideas which may be useful if they can be applied to the automated timbre space.

The Timbre Model and a number of analysis-synthesis techniques all simplify the signal by separating the clean and noisy parts of the input and storing and manipulating these separately. Applying such techniques to the timbre space would generate a significantly different signal analysis matrix, containing a signal portion and a set of noise parameters for each sound. Noise in this sense doesn't just apply to unwanted variations in the signal, but is also used to model micro-variations and inharmonic components of the sound. In the case of many percussive sounds, the noise component of such techniques is more important for accurate reconstruction than the signal component.

These techniques tend to use analysis such phase vocoders or auto-regression to generate a clean signal, then calculate the residue between that signal and the original in order to generate the parameters for the noisy part of the model. In order to use such techniques in the timbre space therefore, a good auto-regression analysis or phase-vocoder post-processing step is required.

Delprat *et al.* and Rolland and Pachet use the TFR of a signal to directly generate synthesiser parameters. Whilst the timbre space offers another level of abstraction, their techniques could be used to run a variety of synthesisers from the timbre space by using the inverse of the PCA map to generate a TFR for the required sound. The two disadvantages of such a process from a design point of view are that the process may be slower than a direct output from the timbre space via a parameter map, and without a parameter map for a variety of synthesisers, the design space does not have access to the parameterisation offered by, for example, Auditory Icons. Their work may however be useful where a particular synthesiser is already implemented as part of an interface. The timbre space could be used to generate a set of important transformations, which their mapping work could then translate, via the TFR, into a set of efficient parameterisations within the given synthesiser, giving an interactive audio environment without requiring the timbre space itself on the target platform.

3.7 Conclusions

The human timbre space is a well-understood model of human perception and has been used for many years for sound discrimination and to break timbre down into further parameters that aid our understanding of how we hear. It thus provides an important source of information that is useful in constructing new sounds, and has been used in this way by many musicians.

In contrast, the automated timbre space is a much newer and consequently less understood model of perception. Hourdin *et al.* have demonstrated similarities between the two timbre space models, but there are still many questions that need answered, such as which analytical techniques are the most useful for creating such a space and do the transformations discovered by McAdams *et al.* in human timbre spaces apply to automated timbre spaces?

The timbre space concept is a promising platform for audio design if these questions can be answered. Despite the objections encountered, it appears that a timbre space representation can offer a generic and flexible framework for exploring sound, although the cost of that flexibility may compromise the speed or size complexity of the space itself or limit the reproductive accuracy of the sounds it produces. The speed issues of mapping and synthesis are alleviated if a single, optimised synthesiser model is chosen, but this hinders the adoption of new models such as Auditory Icons which can be used not only as a synthesis model but also as a design model.

The reproductive accuracy of the timbre space and synthesisers may not be perfect, but it should be recognised that accurate reproductions of environmental objects are not always essential in the desktop context that is being explored. Examples are common in the graphical icons present in modern WIMP systems, whose image may represent a real-world object but is often simplified and abstracted in order to make the intent clearer than a photograph of the object could. It should also be noted that the machine Auditory Icons are not designed to sound like a realistic machine, only to give the impression of a system that can get noisier, smoother, faster or slower. In such light, so long as the fundamental characteristic of the sound is relevant to the intended effect, the similarity of the sound to any real world object is of lower importance. Whilst Gaver's sounds were designed to simulate real-world objects, later work, such as the Sounding Object project [46], has shown that so long as the fundamentally important changes in the sound are kept, such as pitch changes or changes in decay rate, the sound can be dramatically simplified or *cartoonified* without losing its ability to convey information. The timbre space therefore should seek to capture these important changes in sound rather than aiming to fully capture the timbre since those changes are the most important in audio design.

In addition, if we are to assume that most listeners will not receive intensive training prior to exposure to the sounds, Edworthy *et al.* [49] has shown that humans perform equally well on remembering the meaning of environmental and abstract sounds. The following chapters will describe the construction and evaluation of a timbre space as a basis for perceptual audio design.

Chapter 4. Overall system design

4.1 Introduction

In the previous chapters, the current state of audio design was presented and a possible new method for sound design was introduced: the timbre space. This chapter will discuss the design requirements and decisions taken in developing a proof of concept system to test the suitability of the timbre space for audio interface design. The focus of the requirements is towards the aim of creating a generalised, powerful audio interface design system. The prototype built does not have to fit all the requirements of a full audio design system, but should demonstrate how a full system would be constructed from the ideas presented. All the details required to re-create the system developed can be found here and in the Appendices. All the code produced is available on the CD included with this thesis, with the exception of the code for the CQT [3], EM-Kalman (modified to EM-PCA) [2] and k-d tree [4] techniques, which were developed by other researchers.

In order to test the suitability of the timbre space for audio design, it is necessary to build a system which is flexible enough to allow experiments to be designed, that allows new models and experiments to be added quickly and easily and be powerful enough to allow the user to perform a number of transformations within the space. As this system is designed as a proof-of-concept for use of timbre spaces in audio design, the interface to the system is designed for flexibility and power rather than ease of use, although the means to create alternative user interfaces is presented. For a full production system, a number of models and experiments would need to be encoded into the timbre space. These models, whilst important to designers, are generally formalisations of existing research and so only a small number will be incorporated for this research to provide a basis for evaluation of the capabilities of the system and to demonstrate the ease with which new models can be provided as the result of work by practitioners in fields outside traditional computer science.

As well as providing a platform for the experiments discussed in the next two chapters, this framework also provides novel contributions through creating a parameter map that can be used for the Auditory Icon and FM synthesisers within the timbre space, providing novel outputs from the timbre space and providing the timbre space, and hence designers, with access to Auditory Icons, and other physical models, within the same design space.

In addition, the creation of a novel mathematical language within the file format for the timbre space provides a number of novel geometric manipulations of the sound. These are discussed further in Chapter 6, but can also be used to accept interactive input, as discussed in this chapter.

4.1.1 Overview

Section 4.2 discusses the requirements for an audio design system, both in order to demonstrate its compatibility with the stated goals of the project, and to allow the resultant system to be extended through further research to provide a tool that can be used by designers.

Section 4.3 shows how the audio data will be represented and transformed within the system and Section 4.4 shows how this system fits into a design context, with reference to the concepts introduced in previous chapters.

Section 4.6 describes the design goals and decisions made in the implementation of the synthesis framework for outputting sound from the timbre space. Section 4.7 discusses how data are represented at each stage of the system and discusses the tools for manipulating that data and how they are transferred between components.

Section 4.8 evaluates the progress made in developing the prototype.

Section 4.9 briefly discusses how new synthesisers and analysis techniques are added to extend the system for further evaluations and Section 4.10 discusses how a user interface can be implemented on top of the system.

4.2 Requirements

For a complete audio design and synthesis system, the following requirements are needed to ensure the system is interactive and flexible enough to explore different timbre spaces and different interface paradigms.

- **Real-time sound production:** If the system cannot produce a sound in real-time there will be a lag between user action and feedback. This will be a distraction to the designer who may find that there is confusion between the adjustment of a parameter and the change in the sound.

- **Easy to apply reverberation, surround sound, atmospheric, Doppler and other effects to the output sound:** There are many effects that we can apply to change the character of a sound that can only be effectively applied from an environmental model, in a stage after the synthesis.

  There are many existing tools and algorithms for these, and the system should make it as easy as possible to apply these effects to a generated sound.

- **Implementable on a wide range of devices:** Where two devices are communicating in a common task, a common audio interface can be very useful to allow a user to switch readily between devices and also to present a consistent state between all devices.

- **Strong specification allowing others to write their own components:** If the system is built to a well-defined specification using popular standards

> then it is easier for others to add components to the system.
>
> Useful extra components could include the implementation of new technology at the network or sound production level, replacement of existing code with more efficient algorithms, optimised for certain situations, new methods for studying and manipulating the data within the timbre space and parameter space or at the signal level.

For this prototype however, whilst these will all be considered in the design in order to ensure the prototype can be extended to experiment with new timbre spaces and audio interface designs, it will only be implemented on one device (a desktop PC), and the speed consideration will be relaxed in order to explore a variety of techniques for synthesis and manipulation to see which are fastest and which have the highest quality reproduction.

4.3 Data flow overview

Figure 4.1 shows how data is processed and represented throughout the Analysis-Synthesis process.

There are 4 data stages and 3 translation stages that map one data type to another. The data can be stored on disk or it could be data that is simply passed between stages within a machine or between machines. The 7 stages are outlined below, with further explanation as required below that.

The 5 centre stages are defined in the system specification, which is currently a single library defined in an Interface Definition Language (IDL) file[1] used to define a component architecture. The first and last stages are standard waves and are encoded as PCM data.

[1] Many dialects of IDL exist as there are many competing component architectures. The Microsoft IDL [130] is used here since that fits with the chosen architecture discussed later in this chapter.

The three translation stages (the analysis, transformation and synthesis stages) are represented by interfaces in the IDL file, and by darker boxes in the figure. The data stages are represented by data structures in the IDL file and by the lighter boxes in the figure.

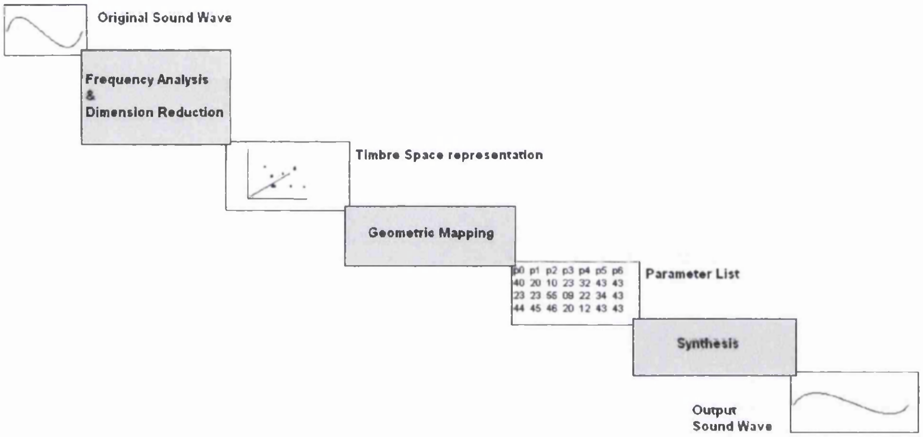


Figure 4.1: Data flow model of the system.

The seven stages of the process are:

1. *Input sounds - from MUMS CD, other recordings, or system process*

   The original timbre space is to be defined from a series of samples from the McGill University Master Samples (MUMS) CDs [131] following the work of Hourdin *et al.* [12] and of Kaminskyj [78]. The sounds are taken from the set used by Hourdin *et al.* and are listed in Table 5.1. Users will be able to input their own sounds as a basis for generating novel sounds within the interface. Some sounds will come from other processes, such as synthesiser output for analysis, or from Gaver style auditory icons.

2. **Via Spectrum analysis and dimensionality reduction**

   This is the two stage process for converting a sound into a path in timbre space. It is encapsulated in a single process to emphasise the fact that the output space is very sensitive to both the spectrum analysis technique and the dimensionality reduction technique. If either technique is substituted for another, or the parameters of the technique change, then the space is no longer valid and all paths in the space have to be recalculated using the original set of MUMS samples.

By using a unique identifier for each timbre space that defines the techniques and parameters used in its construction, later stages can verify that any data to be processed conforms to the current timbre space and perform any necessary conversion where possible. This prevents sounds being corrupted when, for example, a path generated for a CQT timbre space is manipulated according to a STFT timbre space.

3. *Path in timbre space - which can be manipulated in many ways, including via object-based perceptual cues, and can be sent between processes and devices*

   This path is represented by a list of points or a formula and is intended to be a lightweight but robust format that is suitable for inter-process and inter-device communication. It is described in more detail below once general rules for file formats have been discussed.

4. **Via transformations, matrices for linear transforms, but some configurations may require non-linear transforms**

   The projection from timbre space to the synthesiser's parameter space must be performed with full knowledge of the synthesiser's limitations. Usually this requires the transformation and synthesis to be performed on the same device.

   The transformation will ideally be linear as this will be a faster operation, but this cannot be guaranteed so the projection component must be general enough to allow this.

5. *List of parameters through time*

This is the low level commands that the synthesiser needs to create the required sound. The parameter file format specification later in this section describes the issues concerning this.

6. **Via synthesiser, possibly hardware optimised. Environmental effects can be added to the synthesiser output after this stage if required.**

   The synthesiser, although currently DirectX based (see Section 4.6.2), may be generalisable to a multi-architecture design incorporating other standards, such as VST (see Section 4.6.1) by separating the synthesiser logic from the DirectX API. This allows the synthesiser to be used in many hosts beyond those that only support the DirectX component interface.

7. *Output sound - sent to disk or output device in real time*

4.4 Conceptual model

In Figure 4.2 below, the overall design of the system can be seen. On one side there are the synthesisers to choose from and on the other are the object models that will be manipulated by the user. Connecting these two sets is the perceptual mapper (see Section 3.2) that will convert an object level description into a timbre space representation and then map this directly into a synthesiser to produce the required sound.

4.4.1 Object parameters

At the object level, properties of an object can be described and manipulated. Each of these properties will relate to a path or a set of transformations in timbre space.

The objects can be defined either by building up a model of each property via the

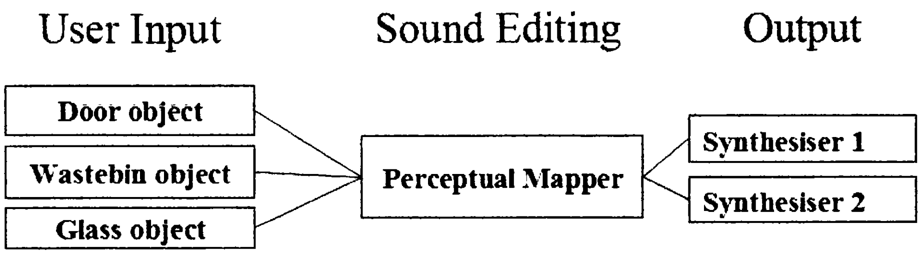


Figure 4.2: Conceptual model of the design system, showing how Auditory Icons can be manipulated using the system. Parameters of the Auditory Icons are presented to the user and map onto a representation in the perceptual mapper. Synthesiser models allow the sound to be output to a specified device. Perceptual models such as those in Chapter 6 can be used in place of the Auditory Icon models shown here.

perceptual parameters that are relevant, similar to the construction of Auditory Icons, which is hard, or by taking sample sounds representing the range of values the property will take and analysing them together to define the point, paths and transformations automatically.

To apply the second approach to Gaver's Auditory Icons (Section 2.3.3), a set of sounds across the parameter range is produced, and this allows Auditory Icons to be implemented as a synthesiser in the system. Using the Auditory Icon synthesiser is useful for sounds that fall within its timbral range, but the addition of the parameters into the timbre space allows other synthesisers to be modified by the Auditory Icon parameters.

This approach provides a direct way to compare the Auditory Icon and the general timbre space based synthesis. With the Auditory Icons situated within the timbre space, it is easy to compare the output of the timbre space model of the Auditory Icons with the output of the standard Auditory Icons and to test the effectiveness of the new design techniques exposed by the timbre space that are unavailable to Auditory Icons otherwise.

4.4.2 Timbre space

The timbre space can encode the perceptual information in the sound. The methods for capturing this information are discussed in Section 3.4 and tested in Chapter 6. The perceptual map in the timbre space is the mediator between the object properties and the synthesis engine. It also exposes perceptual properties via a sound control panel or a general programmatic interface depending on the implementation.

The timbre space is parameterised according to properties of the sound spectrum. The actual properties are highly dependent on the frequency analysis techniques chosen to create the space.

4.4.3 Synthesis engines

The synthesis engine has been designed as a self-contained module that can be replaced by any other synthesiser for reasons of speed, accuracy or coverage of perceptual space.

Each synthesis engine only covers a portion of the entire perceptual space. A Gaver style synthesiser optimised for contact sounds produces poor-quality representations of sustained sounds such as the flute whereas an analogue synthesiser whose sounds are made of summed waves has difficulty constructing the complex spectrum of a scraping sound which has many inharmonic frequencies across the spectrum.

The first synthesiser implemented was the FM synthesiser since it is simple to implement and Kronland-Martinet *et al.* [71] have already studied the parameter estimation problem for the FM synthesiser. This can be used as a basis for comparison with the parameter estimation via timbre space.

Using similar techniques, an Auditory Icon synthesis model (Section 2.3.3) has been constructed to demonstrate that the system can cover at least as much space as existing techniques and to demonstrate the ease with which a new model can be added to the space, since the Auditory Icon provides a set of perceptually relevant parameterisations as well as a synthesis engine.

When outputting to disk, a much higher quality output can be produced by using Additive Synthesis rather than FM synthesis to reproduce exactly the input spectrum to whatever accuracy is deemed necessary. This is the method used by Hourdin *et al.* [63] to evaluate their timbre space, and has been used for evaluating the timbre space analysis (see Chapter 5) although it has not been used in a real-time interface.

4.5 Mapping of space

The timbre space representation presents a new view of audio that has been shown to have some perceptual significance (see Chapter 6). However, in order to use this for design using perceptual constructs, it is necessary to add a model of these constructs to the space as they are unlikely to align with the mathematically generated axes. Interface design has not been previously considered within a timbre space, so a number of ways of modeling the status of objects within the interface must be considered.

Any status variables that we wish to track can be mapped to a dimension, a perceptual construct, or a path in timbre space, such that a variation in status produces a variation in the output. Where the status variable is categorical, the variable will be represented by a series of paths, one per category. In the case of a binary variable, there will be a path for one state and not for another (e.g. a 'click' sound when a button is pressed, and no sound otherwise).

If all the variable paths are straight lines, and they are all perpendicular to each other, there will be a linear transformation between the status and the timbre space which can easily be encoded via a matrix.

If the paths are non perpendicular or are curved, a more complex mapping will be required, such as a Bezier surface [132] or Bezier hyperplane (which is curved) or an arbitrary function from the n-dimensional status space (where one variable is represented by one or more dimensions) to the m-dimensional timbre space.

In a similar fashion, the parameters for the synthesiser can be mapped into the timbre space. Providing this mapping can be inverted, the inverse mapping will translate any point or path in timbre space into a point or path in the parameter space of the synthesiser.

4.6 Studio technology

Studio technology [133] is a set of technologies developed for music software to allow developers to create plug-in effects for the music software. Such effects include reverberation, pitch-shifting and reversing the signal in time. Recent extensions have allowed these plug-ins to also be used to create instruments that accept MIDI parameters and input from a GUI and create an audio stream.

The original technology in the area was developed by Steinberg [134] under the name VST (Virtual Studio Technology) [133]. This was later followed by the DirectX plug-in (and DXi - DirectX instrument) [135] from Cakewalk [136]. The LADSPA format [137] is a more recent open-source alternative to these technologies that offers better real-time control and more powerful parameter handling but was not widely available when the technology decisions for this research were made.

Thie studio technology approach to developing the components of the system allows for multiple inputs and outputs, including the possibility of controlling some components from one input device and some from another. This philosophy also extends to the synthesiser which can have multiple outputs. At the very least, it should output to a system device such as a sound card. This output could have one or more output channels (e.g. mono, stereo, 5.1 surround sound). It could also produce output to disk if certain devices require pre-rendered sounds. Disk output also allows analysis of the sound, which is useful both for development and for generating a mapping into the timbre space from a synthesiser.

Alternatives to studio technology: Using pipes, which are simple streams stored in shared memory to facilitate data transfer between programs, it is hard to quickly bypass a single stage or change parameters on the fly, particularly from an interactive process. The component approach offers more flexibility and allows more control over the data flow such as re-sending data and synchronising systems for a real-time response.

4.6.1 VSTi from Steinberg

Steinberg's VST was the original studio technology specification and is very popular amongst developers, although developers must agree to Steinberg's licence in order to use the developer's kit. The format is supported by many plug-in components and by many audio host applications across a few platforms, with VST applications available on Mac, Windows and BeOS. Steinberg have not provided a Linux port of this format. The VST format is a less open format than DXi, described in the next section, as DXi's can be written without their developer's kit using DirectShow technologies described later. For maximum compatibility with existing Earcon design software, the synthesiser should support VST, but with suitable separation between the implementation and the API, support for VST can easily be added later.

4.6.2 DXi from Cakewalk

Cakewalk's DirectX plug-in SDK is a less mature specification than VST that was popular at the time the system was designed, but is only supported on the Microsoft Windows platform. Importantly for this application, Cakewalk has released sample code for a DXi host application, which provides a lot of useful information on top of the documentation on running a DXi outside a monolithic host application, such as Cakewalk's SONAR [138], which is unsuited and unavailable for portable devices. This specification is built upon technologies from Microsoft, utilising the DirectShow component of Microsoft's DirectX suite of APIs.

DirectX API from Microsoft

DirectX is a marketing term from Microsoft that encapsulates a lot of useful multimedia technology to allow the programmer to interface with sound and graphics cards and be guaranteed the best performance on systems that support the latest features, as well as graceful degradation where the hardware does not support the features. The DirectX SDK is available from the Microsoft web-site [139].

The core of DirectX is built around a set of interchangeable components where data filters, hardware drivers, file readers and various other systems are all implemented as components that can be loaded and unloaded at runtime to interface with whatever files and hardware are available. The component technology at the heart of all these interactions is known as the Component Object Model (COM).

COM

COM is a very complex technology, and there are many books available describing how it works. This will only be a very short overview of how the COM technology works in order to give a justification for using this technology in the project.

COM technology is built around standard object concepts of inheritance, polymorphism and encapsulation. Unlike standard object programming however, where the object is inserted as code into the current project and is compiled with it, the COM object is a pre-compiled binary object that resides in an external file.

The binary object can be written and called from any language that supports the specification (such as C and C++). The object can reside on the machine that requested it, or on any other machine for which the process can gain execute access to the object. In this remote case, the COM framework handles the complexities of executing methods and passing code across the network.

The specification of a COM object is defined in an Interface Description Language (IDL) file [130]. It is similar in structure to a C++ header file and describes an interface (which is a class declaration), the methods of that interface and the data types used by those methods. The IDL file also defines which arguments of the methods are inputs and which are outputs and provides fields to describe the function and to specify certain properties it is expected to have, such as whether variables are read-only or whether the method is expected to execute if invoked remotely.

All COM objects inherit ultimately from the IUnknown interface which provides a standard way of finding out which interfaces a COM object supports and a standard method for reference counting an object to make sure it only deletes itself once all processes accessing the object are complete. This makes it easy to replace one COM object with another that supports the same interface.

To identify each COM interface, the IDL file assigns a unique number known as the Globally Unique ID (GUID) to each interface. All COM interfaces have a GUID and it is guaranteed to be different from the GUID of all other COM interfaces. This makes it easy to identify separate components and to tag data so that will only work with the correct COM object.

To implement a COM interface, a component simply inherits the interface and defines all the methods of that interface. Many COM objects can implement the same interface, such as a hardware driver and a software renderer both implementing an interface for displaying graphics on the screen. Each COM object also has a unique GUID in addition to its interface GUID, and this number is not shared by other COM objects or interfaces.

To use a COM object, a program simply invokes the COM framework and requests an object via its object and/or interface GUID. The object is then guaranteed to have implemented all the methods of that interface and the caller can use them as if they were part of its own code.

4.6.3 Cross-platform support

For this project to achieve its overall goals, it must be possible to run it on a number of different platforms. Using COM and DirectShow is a step towards this since it is available on Windows and Mac platforms on the desktop (as well as WinCE on portables), and is available for a cost on IBM and SUN operating systems too. The WINE project [140] is seeking to port all the Microsoft Windows 9x and NT APIs to Linux, allowing Windows software to run under Linux. The DirectX APIs are also being ported under the name Cedega by the TransGaming team [141].

Alternatively, there is an XPCOM API from the Mozilla organisation [142] that accepts IDL defined components and can run plug-ins on Windows and Unix, as well as allowing these objects to be used in JavaScript, Perl and Python scripts. The format is similar, but not identical, to the Microsoft COM specifications. Unlike Microsoft's COM, this format does not have a defined multimedia library.

In contrast to these, there is a cross-platform component architecture known as CORBA that could be used for this project. There is a good overview of this technology at `http://linas.org/linux/corba.html` but there is no multimedia plug-in format designed for this similar to DirectX, so it has not been investigated further, although there are a number of related open-source technologies that could be incorporated into its framework.

There are many cross-platfrom multimedia projects such as OpenAL and the Simple DirectMedia Layer (SDL) [143] API that allow quick and easy access to hardware in a similar thin layer concept to DirectX, allowing real-time components to be written for this system. There is also a European project to develop a Linux distribution optimised for multimedia applications. This new project is hosted at [144] and completed in Summer 2004, so was not complete in time to be useful for this research, but it has produced multimedia components that can work with CORBA, and also works with the LADSPA [137] plug-in format.

Sun also supports COM technology in its Java language, opening methods to communicate with any device that runs Java. As Java includes its own audio APIs, this should provide all the tools needed to implement the system on a portable device.

Although none of these options has been tested, it is important to note their existence, as they open up the system to the widest possible audience. It should also be noted that most of the design and analysis systems are only viable in a high-powered and extensible desktop system.

If the system is ported to a portable device, it will be optimised for streaming and producing sound with perhaps a minimal command set for generating pre-compiled sounds or for recording new ones. This will be a limited capability however due to the speed and storage limitations on most of these devices. Most of these devices will be designed to stream the timbre space data from a server and render it, submitting few sounds back to the server. These devices can also use pre-recorded samples generated by the timbre space.

4.7 Storage and manipulation of audio data

In order to achieve a strong design, the interchange of data between components must be standardised. This consists of two parts. On one side is the direct transfer of data between components based on the component specifications. On the other side is a well defined file format that allows the data to be stored, transmitted, or manipulated by external tools.

Ideally, the two formats should be as close as possible, but there are extra considerations that a file format must take into account to recover from malformed or missing data. The file format has the advantage of flexibility, in that it can describe sounds via meta-properties such as the start and end points of a path in the timbre space and the number of time steps, rather than just listing the points themselves. The parsing component is then responsible for converting this meta-property into a list of points that the lower level processing tools can understand.

An annotated example of a parameter file is available in Appendix B. The format is designed to be extensible so that new techniques can be added without affecting older ones, so there may be features useful to audio interface design that have not yet been considered and hence are not included in this example.

The use of a parameter file allows novel interactions with the paths in the timbre space via geometric manipulations such as interpolation between and averaging over paths in the space. To facilitate this, the existing APIs must be extended to cope with the timbre space data requirements.

The general principles that the file format must conform to are as follows:

- **Robust grammar:** The file format must be strongly specified so that malformed or missing data is easy to detect. Header information and format specifiers for all data are essential to achieve this.

In addition, the format should include directives on what a parser should do if it encounters missing values, whether the current input should be abandoned or whether values should be interpolated around the malformed data.

Since the data is used to drive a temporal system, any malformed data will define only a short period of time. Since the system should degrade gracefully and still play a sound rather than declare an error, the specification suggests that malformed or missing data is simply ignored. The implementation can decide whether to repeat old values or interpolate over the malformed data using the values before and after it in time.

- **Easy to encode/decode:** A file format that is easy to understand is much easier for others to incorporate into their code, and hence makes it much easier to share data.

  For this reason all files will be based on a standard format. ASCII is an obvious choice for readability, and XML is an ASCII format that is currently popular so that is a reasonable choice for exporting data. Binary formats are often more compact however, so if size is a problem, a binary-based format such as SDIF [83, 85] could be a better choice.

  In the current development stage, the intention is to make things as simple and as flexible as possible so an ASCII format has been chosen. XML produces a large overhead, particularly where the data is primarily numerical, so a simple ASCII based format is preferred.

- **Small file size:**

  Provided all the above constraints are met, the file format should not waste unnecessary space as larger files take up more space and are slower to load into memory for parsing.

However, a file format which is easily and efficiently compressed will take up less space and can be transferred between devices faster.

The data formats used are defined using an IDL (Interface Definition Language) class hierarchy, based on the structures used by the DXi synthesiser format. The complete definition, alongside the APIs for each component are included in Appendix A. The most used parts of the system are the format used to define transformations in the timbre space and data transfer between the timbre space and the synthesiser. As most of this data transfer and manipulation is done via file formats, in order to store transformations, this file format is discussed in most detail. A quick look at the motivation for the data formats is provided, but the specification for those can be found in Appendix A.

4.7.1 Parameter file format

The parameter space format contains a time-ordered list of parameters to be sent to the synthesiser, or a set of commands to generate parameters. The actual parameters are specific to the synthesiser, and may not include all the parameters of the synthesiser. Those not listed are left at their defaults, where the default value has to be defined as the parameter 'off' setting. An FM synthesiser, for example, may only use some of its oscillators for particular sounds, so the parameters for the other oscillators are not included and so default to zero.

The format takes its inspiration from existing music formats such as Csound [15], where each line of the file represents a note. Csound, unlike this system, defines the instruments themselves in a text file, as well as defining the notes they play.

If the synthesiser has its own Csound representation then it should be a trivial matter to convert from this parameter space format into Csound, but it would be hard to perform the inverse operation since Csound is designed for off-line synthesis and can have arbitrarily complex instruments.

The file format is designed to work as a text stream. No line in the file depends on the lines that come after it. This allows a synthesiser to play a file while it is still being read. The file format reflects the fact that the parameters are synthesiser-specific and uses a unique ID to reference the correct synthesiser in the file: the GUID (globally unique ID) of the COM object that implements the synthesiser. The file format assumes that the data is highly dynamic, and assumes that all data will change between time steps, although there are shortcuts to repeat data for 1 or more steps.

This design allows the system to use multiple models, by providing access to a number of synthesisers. The streaming option provides interactive control of the sound which aids the construction of audio, although it is not essential. The opportunity to convert to a CSound or SDIF representation allows comparison of the timbre space with existing analysis-synthesis techniques provided the timbral range of the space is sufficient for such comparisons. The following sections explain how this design can incorporate geometric transformations within the timbre and parameter space and also encode the mapping between the spaces required to output to a variety of synthesisers. In order to understand these aspects of the design, an overview of the main features of the file format is provided below.

Design

For full details of the parameter file format, see Appendix B. A brief overview is provided here since the terms and formats introduced here are used to describe current and future research work described later in this thesis. The format is defined in reference to existing formats such as Csound in order to simplify the creation of new tools for analysis of data files and to build on their experience over many years and users.

A parameter (*.tsp) file is divided into sections denoted by text delimited by square brackets, such as [Synth]. The [Synth] and [Parameter] sections define global settings and allow the correct synthesiser for the file to be identified.

The [MonoOut] section generates a single output that will be sent to the current file or the current output device. This is the simplest type of output. If no section information is found in the file, this is the default. It consists of a list of values, where each line represents a fixed time step, and each column represents a synthesiser parameter.

Each value in a row must evaluate to a number, but simple mathematical equations can also be used. In addition, variables can be defined using lines starting with the # character, allowing easy adjustment of several parameters by changing one line. Inspired by the Csound format, these variables can change according to other criteria, such as the time since the start of the sound, or the number of times another variable has changed value, allowing sounds to naturally change over time by simply assigning some values to dynamic variables. The code fragment in Figure 4.7.1 shows how to write a sound whose pitch rises every quarter of a second for 5 seconds.

```
#variable = 1                       % initial value
#variable = variable + 1 | 250 ms   % Add one to variable every 250 ms

[MonoOut]
% pitch                parameter1   parameter2
  (440+(variable*110)) 15           20
{5 seconds}                         % Repeat above line for 5 seconds

% pitch will be automatically raised after 250ms when the value
% of 'variable' changes.
```

Figure 4.3: A demonstration of the use of variables in a file. The % character indicates comments, the # character indicates a variable definition and the | character defines the interval of evaluation. Whitespace is used solely for separation and has no special meaning.

The [Multifile] section will generate a series of output files according to a set pattern. For example, an Earcon sound in a Multifile section can output a set of files, differing only in the instrument used, where each file is named after the instrument it uses.

Each sound produced by a `[Multifile]` section has its own name. If the output is to disk, an appropriate extension will be added to the name to make the output filename. If the output is to a device, it is expected that the host will create a stream with that name and allow the user to manipulate that stream further, for example, placing within a 3D space or applying an echo filter to it.

Any other section names are currently ignored allowing further sections to be added at a later date without affecting existing software.

4.7.2 Timbre space file format

In addition to the parameter information for the synthesiser, it is necessary to store information about the timbre space. The paths representing the sounds and the points representing regions of space need to be encoded and easily manipulated. There also needs to be an easy way to define a mapping between the timbre space and the parameter space, as discussed in Section 3.5.2. The paths in timbre space will be defined either as a list of points or as a function of time (and possibly other parameters).

This format can be accompanied by converters to convert between this and a series of other popular multi-dimensional formats in order to use tools devised in other fields to be used to manipulate audio paths to generate novel sounds and transformation. Possible formats include those used by engineering and mathematical software, visualisation software and possibly software that interfaces with advanced I/O devices such as haptic systems or location devices that could be used to generate a path in space that could be interpreted as a path in timbre space. Another important format to consider is the SDIF format used in Analysis/Synthesis tools (see Section 3.3.1). If a converter between SDIF and timbre space was available, a wide range of A/S tools become available to the sound designer.

Design

By a slight modification to the parameter file format, it is possible to encapsulate timbre space points and paths within the same framework. Not only does this simplify the creation of the format, it also provides access to the mathematical constructs already defined. These constructs allow the spaces to be manipulated mathematically within the parsing environment, allowing access to complicated transformations. Some of these transformations can be used to construct the models of urgency evaluated in Chapter 6, but can also include transformations based on traditional uses of analysis-synthesis such as morphing between two or more timbres, using linear or non-linear interpolation. Other transformations possible include applying the amplitude envelope, or other property of one sound to another timbre, by substituting dimensions of one timbre for another, or stretching a sound in time without changing pitch, by increasing the time step between points, interpolating between points, or creating loops within the sound.

These geometric modifications can therefore provide a platform for generating new transformations outside of the final graphical interface. These new transformations allow audio designers to manipulate sounds in novel ways that are not available in other systems, particularly where the designer is working at the synthesiser level. The abstractions of these mathematical transformations free the designer from knowledge of individual output devices. Designers themselves may however require other researchers to develop complex transformations for specific tasks, but these transformations are easily encapsulated within the file format and made available to the community.

The addition of timbre space data into the file format is accommodated by two minor and two major changes. The minor changes are:

1. Add a new `[Analysis]` section, analogous to the `[Synth]` section, to describe the technique and parameters used to create the space to ensure manipulations are performed on sounds from the correct timbre space.

2. Add a new section type [=*Instrument*] to describe a path in the timbre space. This looks just like a `[MonoOut]` section but all the columns are floating-point numbers and represent an axis in the timbre space, and the matrix defined by the section can be recalled using the instrument name.

The major changes required are:

1. When an [=*Instrument*] section is played, it is converted into a `[MonoOut]` section via the mapping already defined for the timbre space before being heard.

2. To perform geometric manipulations, variables must be able to hold a 2D matrix of values as opposed to a single value. The [=*Instrument*] section described above is defined using such a variable. These matrix variables must have a range of possible geometric manipulations in order to perform the required manipulations.

Geometric manipulations which can be performed given the above structure include:

1. Morphing between paths on a parameter map, for example between a metal tap and a wooden tap on the Auditory Icon map.

2. Applying a perceptual construct such as 'urgency' to the path.

3. Adjusting the volume of a sound by multiplying its matrix by a constant. Values between 0 and 1 will make the sound quieter. Values greater than 1 will make the sound louder but may cause the resultant sound to overflow.

For ease of parsing, a matrix section takes three parameters, the last of which is optional. A full matrix section header hence looks like [=*Variable-Name*@rows=r,columns=c, **static**] where r is the number of rows defined, c is the number of columns and the optional **static** argument is a hint to the parser that the data in the matrix does not change (i.e. does not depend on global or dynamic variables) and the matrix can be saved in its computed state (i.e. without variables) to save space and calculation time. Leaving this parameter off allows the matrix to change depending on the state of the global variables in the file, but can increase the processing time since the matrix will need to be re-calculated every time it is used.

Given a set of matrix variables, a grammar is required to define transformations using those variables. Examples of how such transformations are written are discussed in the following section.

4.7.3 Transforms

The addition of arithmetic operation and matrix variables to the file formats allows generic transformations to be performed in both timbre and parameter space. In the following discussion, assume that the instruments and transformations exist in the file as a result of prior analysis or experiments to verify their validity.

With these additions, the merging of three sounds looks similar to:

$$0.3 * \texttt{piano} + 0.4 * \texttt{violin} + 0.1 * \texttt{guitar}$$

where we assume the three instruments are defined as matrix blocks with the same number of columns as the timbre space. Where one sound is shorter than another, it will be padded with blank rows (i.e. silence) at the end.

Further, assume the urgent variable is defined as a single-row matrix (i.e. a vector) with the same number of columns as an instrument, and that it defines a sound accepted to be urgent (for example, as defined in Edworthy *et al.* [31]). When such variables are combined with a multi-row matrix, the combination will apply the single-row vector to every row of the multi-row matrix.

Two models for applying the urgent variable to an existing sound are easily implemented using the file format, and many others may be available. In the *'gravity'* model, inspired by the morphing techniques common in analysis-synthesis,the sound is pulled towards the urgent sound, and is represented in the file as:

$$0.7 * \texttt{piano} + 0.3 * \texttt{urgent}$$

In the *'vector'* model, inspired by the parallelogram model [19], a vector of increasing urgency is defined between a non-urgent and an urgent sound, and is applied to the `piano` sound as follows:

$$\texttt{piano} + 0.3 * (\texttt{urgent} - \texttt{nonurgent})$$

where `(urgent - nonurgent)` could be precomputed and stored in a variable, and `nonurgent` would define a sound accepted to be non-urgent, from the same source as the urgent sound.

In addition, with a minor change to the variable format to allow an [=*Instrument*] section to accept optional arguments after the matrix definitions, that are implemented as local variables, it is possible to encode instruments and regions of space as parameterised matrices, so that, for example, a flute sound could be used as follows:

$$\texttt{flute(\_TIME)}$$

where `_TIME` is a variable indicating the current playing time. The flute could also be combined with an alarm of strength 3:

$$0.7 * \texttt{flute(\_TIME)} + 0.3 * \texttt{alarm}(3)$$

This feature is not currently implemented as it is not required for the proof of concept, but would be important for creating space-optimised representations of sounds required for a complete tool. This feature would require the addition of branching statements in the format to allow accurate modelling of the attack, sustain and decay portions of the sound by branching according to time.

4.7.4 Mapping

To implement a mapping from timbre space to parameter space, it is necessary to define a set of points in timbre space and their corresponding points in parameter space. Whilst there are many ways to represent this mapping, for compatibility with the chosen k-d tree method, the mapping is defined by a special section in the file. This section takes three arguments, the set of points in timbre space, an equal number of points in parameter space and a mapping method, currently restricted to **kdtree**.

The mapping section relies on both sets of points to be previously defined by matrices and no mapping will be performed otherwise. The section header takes the pattern:

`[TimbreMap@TM=`*TimbrePoints*`,PM=`*ParameterPoints*`,Type=`*MappingMethod*`]` where *TimbrePoints* and *ParameterPoints* are the pre-existing matrices and *MappingMethod* is currently **kdtree**.

To use the mapping for generating output, the argument **timbre** must be added to the appropriate section as follows: `[MonoOut@timbre]`. Only one mapping can currently be defined at any one time. Further mapping types may be able to make use of parameterised matrices.

Whilst this technique has been tested within the parser, the complications of adding the k-d tree implementation to the C++ framework has prevented the testing of the parser's ability to use the k-d tree according to the format defined. The rest of the grammar presented here has been implemented successfully in C++ and the kd-tree has been implemented successfully in Matlab.

4.7.5 DXi parameter data format

In order to pass data to the synthesiser, a format must be declared that allows the host to configure the synthesiser and pass parameters to that synthesiser to change the timbre it produces. The format should not break existing data exchange formats used by the DXi API. Specifically, the synthesiser should respond as defined by the MIDI standard when messages defined in that standard are passed. Although it is free to ignore any message it wishes, the synthesiser should not map a defined MIDI parameter onto an internal parameter not recognised by the standard.

The data format should be general enough to cover a wide range of synthesisers whose parameters may be real or integer types and may cover a wide range of possible values. If the timbre is to be adjusted whilst notes are playing, the data format should be able to tag the parameter data with a time stamp to indicate when the timbre change should occur. The format should also allow the synthesiser to specify what type of data it expects and to degrade gracefully if the data supplied is of the wrong type. This prevents the synthesiser from generating garbage or crashing when the input data is corrupt.

The DXi specification already allows two methods for passing parameters to a synthesiser, the MIDI NRPN[2] signal and the DXi parameter interface. There is a third

[2] Non-Registered Parameter Number, i.e. reserved for vendor, used to allow vendors to extend

option: to create a unique parameter specification for the system. This option is compared to the existing options in this section.

The passing of and encoding of parameters is important as a good design not only allows for efficient passing of data between the timbre space and the synthesiser, but also allows easy creation of the suite of sounds required to create the mapping between the synthesiser and the timbre space. Although the timbre space will treat all parameters as floating point values, it is necessary in many cases to use fixed integer parameters for speed and accuracy reasons within the synthesiser. A good parameter format will encode the type of each variable so that the timbre space to parameter space mapping can produce sensible parameter values.

MIDI NRPN signal

The NRPN (Non-Registered Parameter Number) signal of the MIDI specification allows the synthesiser to accept up to 65535 parameters each taking a value in the range 0-65535.

A non-registered parameter is synthesiser specific as opposed to the registered parameters which are defined by the MIDI standard. NRPNs hence allow the synthesiser designers to extend the functionality of their device so long as it is communicating with compatible devices, such as others from the same manufacturer.

Using NRPNs for passing parameters leverages the existing MIDI technology and allows parameter change messages to be sent from external programmable MIDI devices. It also allows the synthesiser host to use MIDI commands alone to control the synthesiser, giving the option of controlling pitch, loudness, duration and timbre completely via MIDI messages, rather than requiring a separate set of parameters simply to control timbre.

the MIDI format for their own devices without abusing the MIDI standard that other devices expect.

The advantages of the DXi or a customised approach are that the number of controllers and their ranges are limited only by the language the components are written in, allowing much larger parameters than the MIDI spec without the need to manipulate data at the bit level when a larger range is required. The MIDI solution is also slower since it sends data in packets with headers as large as the data.

The primary advantage for the non-MIDI approach is that the pitch in a non-MIDI message can be sent using a floating-point frequency as a parameter, allowing much finer control of the output sound. This advantage allows for an easier synthesis of sounds such as the pouring sound Gaver used to indicate progress where the sound is never guaranteed to be on a musical pitch.

DXi parameter interface

As part of the DXi specification, a synthesiser must expose a set of parameters to the host. These parameters are usually controlled by the instrument's GUI displayed by the host, but can also be recorded and played back alongside the MIDI data.

The synthesiser exposes an interface to the parameters it will accept and also exposes a set of methods to set and get those parameters. The synthesiser also allows all parameters to be set at once by passing an IDL structure to a method in the synthesiser.

Using the DXi parameters provides a consistent method for updating parameters to the host and to the GUI of the instrument. The specification for these is already tested and included in the DXi SDK[135], providing a lot less work for development.

The problems with DXi parameters become apparent when considered in an environment that may produce an invalid external parameter file. The methods for setting all the parameters can only check the size of the incoming data. They cannot check if the data is well formed or if the data is in a valid range. An example when applying a range to data is useful is to prevent the volume of the output signal exceeding 100%.

In order to check the data, the parameter structure needs to know what data types are expected and the range for each type. This can be easily added with a complex data type in a custom parameter specification. Although the COM API defines a method for doing this through the `IDispatch` interface and the `ITypeInfo` interface, it can only return the types of the parameters via their get/set methods, and through several levels of indirection. The interface also does not provide methods to check the range of data.

More fundamentally, unlike MIDI signals, which all have an associated start time, DXi parameters are set immediately. The host must therefore inject the signals directly into the DXi at the appropriate time, removing the advantages MIDI offers of queuing up events in advance, allowing batch processing of messages.

These considerations have led to the decision to implement a custom parameter data format, which is defined in the IDL file alongside the interfaces.

Custom data format

The custom data format used has a time stamp, which allows for queueing of parameter changes alongside MIDI events, but also has a type system, allowing parameters to take floating-point or integer values without having to perform bit-manipulation as in the MIDI NRPN case.

The custom format also allows a full set of parameters to be set at once, which improves performance as the overhead for data transfer is reduced. In order to achieve this, a set of fixed parameters is defined for each synthesiser (namely frequency, start time, volume and duration), and a separate array is defined for the synthesiser-specific parameters. This array is defined as a pointer and an integer which defines the number of parameters defined in the array. To help mitigate possible memory errors, each synthesiser must define a function which returns a full set of parameters that have been properly allocated allowing the host to adjust only those parameters it needs to, and also confirming the type of each of those parameters.

In addition to the type and value, each parameter is also given a name. This is useful when a synthesiser has a large number of parameters and a host only wants to set a few. The host can determine from the names which parameters it is interested in and only adjust the values of those parameters.

4.8 Current implementation status

The DXi synthesiser and a host have been implemented in C++ in order to test the file and data formats proposed. The host implementation includes a parser for the timbre space and parameter space files and will output to the FM synthesiser. Unfortunately, due to technical difficulties, the k-d tree has not been incorporated into this framework so the discussion here can only evaluate the synthesiser and parameter space link and the tested parsing stage of the timbre space.

4.8.1 Current status of the DXi

The DXi has been implemented as a simple FM synthesiser with two oscillators and no envelope. The enveloping is provided via the supplied parameters, allowing a wide range of external envelopes to be used. An Auditory Icon synthesiser has also been developed that understands MIDI signals, but the problems with the k-d tree have prevented its implementation against the parameter space framework.

The DXi currently understands 4 MIDI signals: NoteOn, NoteOff, ChannelVolume and ChannelAftertouch, which allow full control of pitch, loudness and duration of each note, providing a minimal set of commands to support Earcon design. It responds to both pre-recorded and live input (which are represented differently by the DXi format), and it also responds to parameter updates sent via the GUI and via the parameter space interface.

The DXi has been tested against large input queues with complex data via Cakewalk SONAR on the complete set of MIDI files provided with Windows 2000, and is stable and produces output for all the files with the correct pitch, onset and offset for all notes with up to 16 notes playing at once. Where more than 16 notes are requested, only the 16 with the highest pitch are played.

This synthesiser is the first to accept MIDI commands for Earcon design and timbral commands, via the parameter space, required for Auditory Icon design. The FM synthesiser is simple and so does not produce the complete timbral range required for a full audio design system, but has shown that the combination of the two fields is possible within the same system. However, the synthesiser currently lacks a GUI for controlling the timbre space and can only accept commands for timbral changes. A discussion of how such a GUI can be implemented can be found later in this chapter.

The synthesiser logic in the DXi is incorporated in a number of worker functions, separate from the DXi API. This design allows the synthesiser to support a number of other APIs with minimal changes. Currently only a small portion of the IDL that defines the synthesiser requires the DXi API to function but that can easily be removed by allowing the synthesiser multiple inheritance from the DXi IDL and the IDL defined for the framework. This completes the implementation required at step 6 of the Data Flow Overview.

4.8.2 Current status of the synthesiser host

The parameter space is currently implemented within a Windows application that provides an editor for the synthesiser parameters. This application will parse an input file in a specified format and allows modifications to be made to the file. The application acts as a host for the DXi and currently sends output to disk only, to a file specified via the GUI.

The application is designed in a modular fashion with each of the GUI components, the perceptual map itself and the DXi host encapsulated in reusable classes. See Appendix A for full details.

The parser is robust enough to parse incomplete files and can identify many errors. The parsing component is designed to be portable enough to be reused on a large number of platforms with minimal modification.

The parser currently handles the `[MonoOut]` and `[Variables]` sections completely, and most of the [=*MatrixName*] section apart from the parameterisation of matrices. The host however only works with the FM synthesiser. It does not yet handle output to multiple files from one input file, which would be required for adding new synthesiser meshes to the timbre space.

The incorporation of the mapping from timbre space to parameter space is complete at the parsing level but the current implementation of the tokenise and parsing cycle alongside the current implementation of the matrices required to store the input and output points for the map lead to a solution that is currently far too slow for a production environment, and noticeably slower than the Matlab implementation of the same mapping. Whilst the synthesiser and host can be demonstrated, the evaluation of the mapping has been performed using the Matlab implementation. The Matlab implementation currently supports all three data translation stages from the Data Flow Overview, but the parser and synthesiser present only a partial implementation of the manipulation and sound production stages (i.e. stages 4 and 6) due to problems encountered in the timbre space to parameter space mapping.

4.9 Extending the system

This section discusses how the system is constructed and how it can be extended. It explains both how new analysis and synthesis techniques can be added into the Matlab framework and how the data can be stored and manipulated to create new sounds.

4.9.1 How to add a new analysis technique

The analysis code is currently written in Matlab [105]. It is intended that until C++ offers native matrix support via the Standard Template Library (STL), all new techniques should be added as Matlab code.

To add a new function, follow these steps:

1. Write a new Matlab function that takes in sound data and a set of parameters and produces a Time-Frequency Representation matrix. This has been done for the CQT, STFT, Fractional Fourier Transform and AR/LPC techniques.

2. Choose a unique name for the technique to identify it in a test suite such as "CQT", "STFT" or "FRT".

3. Add the technique into `generateAnyTFR()` using the unique name to pass the data onto the new function

4. Add the technique into `parseTSsettings()` to allow it to be easily incorporated into a test suite. See Section 5.3 for more on how the test suites have been constructed and used to evaluate a range of spaces.

5. (*Optional*) Add the technique into `invertAnyTFR()` if it has a special resynthesis technique you want to test other than sending it to a synthesiser. This is useful for testing by generating the highest quality output if the TFR is not immediately suited for input to an additive synthesiser.

4.9.2 TFR matrix

All the analysis techniques tested in the first set of experiments (see Section 5.3) produced a TFR matrix where each row was a period of time and each column a frequency bin. When phase information is added to the STFT analysis, the TFR is expanded to hold phase columns as well as frequency columns. So long as every sound produces the same number of columns when analysed, this matrix can still be used to construct a timbre space, although the additive synthesis algorithms used to reproduce the sound data expect to find only frequency columns. Consequently, when a matrix with mixed columns is used, a new technique must be added to `invertAnyTFR()`.

4.9.3 How to add a new synthesiser

Although there is no code to automate the procedure, new synthesisers can be added to a timbre space within Matlab by the following procedure:

1. Create a synthesis function in Matlab that takes at least a sample rate and a time length parameter and outputs an array representing the sound. It should be possible to send the parameters to the function as an array. A wrapper function can be used to accept the array format.

2. For a selection of points, produce a sound from this function no shorter than the analysis window of the timbre space.

3. Use the timbre space code to analyse each point into the timbre space. Use this to create two arrays: one with the synthesis parameters and one with the points in timbre space corresponding to those parameters.

4. Use these two arrays as input to a modified k-d tree [116] algorithm that will tag the parameter information onto the timbre space points.

The updated k-d tree algorithm, with nearest neighbour search, has been implemented for the FM synthesiser, using the tagging method discussed in Section 3.5.2. This method should work effectively for other synthesisers without modification but this has not been tested.

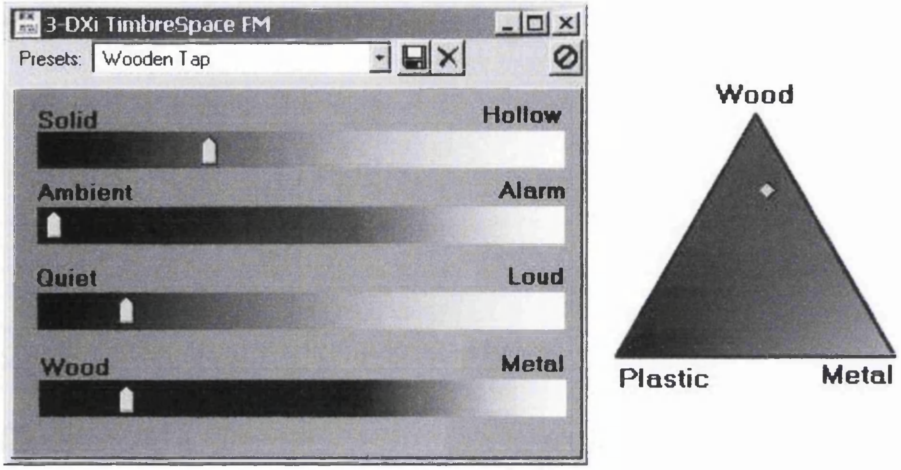
With a complete parameter file parser, the `[Synth]` section can be used to generate the set of parameters and sounds for step 2 above, discarding each sound after analysis so that only the parameter and timbre space points have to be stored. The code for handling this allows the resolution of the parameter space to be tuned so that the space is small enough to ensure a reasonable runtime.

The inclusion of an Auditory Icon or other synthesiser into the timbre space via this technique situates its parameters in the timbre space allowing Auditory Icon transformations to be mapped via the timbre space onto any other synthesiser.

4.10 A simple GUI

4.10.1 How to add GUI components

Since the goal of the research is to examine ways to aid audio design, it is instructive to see how well the file format can be applied to a GUI interface in order to expose the audio design possibilities to the user. This section demonstrates the ability of the file format to respond to input from a slider and a more exotic input widget. The actual implementation of the message passing between the widget and the file parser will depend on the capabilities of the windowing toolkit that is used.



3-DXi TimbreSpace FM

Presets: Wooden Tap

Figure 4.4: Possible GUIs for audio design.

Since the timbre space file format allows the use of variables, it is easy to provide a GUI to access the commands. A slider represents a range from 0 to 1, indicating how strongly the effect is to be applied. For example, an 'urgency' slider using the gravity model will have the same effect as the following code in a file:

$$(\mathtt{1}-\mathtt{i}) * \mathtt{piano} + \mathtt{i} * \mathtt{urgent}$$

where i is the value of the slider.

For the triangular slider in Figure 4.4 the xy position in the triangle is translated into three values, i, j, k, such that $i + j + k = 1$ and the ratios between them match the ratios between the distances between each corner and the selected point. The transformation shown can then be represented by the following code:

$$\mathtt{i} * \mathtt{wood} + \mathtt{j} * \mathtt{plastic} + \mathtt{k} * \mathtt{metal}$$

4.10.2 Extending the GUI

A timbre space synthesiser could implement a range of selectable sliders whose function would depend on which model was chosen and a number of selection boxes to set the configuration of the model. For example, in the Auditory Icon model, the sliders would represent the hardness and the dampening of the material used, and the selection box would choose between the type of object (such as 'string' or 'bar'). The parameters set within this model could then be modified by perceptual constructs such as 'urgency' or 'hollowness' by changing the active model from Auditory Icons to a perceptual model.

To create more complex transformations such as pitch-shifting with a change in envelope, it should be possible to script transformations in a similar way to scripting in modern graphics packages or MIDI scripting in modern sequencers. These scripts would use the same structure as the file format discussed previously, but would be accessible to designers through a standard interface in a similar way to scripts in existing packages.

Whilst these transformations and interfaces have not been implemented, the examples here show that the barriers to implementation are primarily software engineering problems. This thesis has provided the foundations necessary to implement these ideas and the experiments that follow demonstrate their fidelity. As the actual implementation of these ideas is highly dependent on the system environment and available input and output hardware, only the scientific aspects are presented here and it is left as an engineering exercise for others to implement them on specific environments.

4.11 Conclusions

This chapter has covered how the system used in this thesis has been constructed and why it was built the way it was. The system has been developed in order to test the goals defined for this research and has been left open so that it can be extended from a research project to a wider tool via further engineering using the techniques presented here.

The three data translation components of the Data Flow Overview provided at the start of this chapter are the capture stage which converts sounds into a timbre space representation via analysis, the manipulation stage which adjusts that representation, and the output stage which maps the timbre space representation onto a set of synthesiser parameters and outputs the result.

The generation of a timbre space is discussed in more detail in the next chapter, as is the conversion of timbre space data to parameter data. The manipulation stage has been demonstrated by the creation of a novel file format and parser to allow geometric manipulation of the paths. This ability is demonstrated in the discussion on adding a GUI, and is explored further in Chapter 6 where two manipulation techniques are compared for their ability to generate a perceptual map in the timbre space.

The novel contributions of geometric manipulation of paths in the timbre space and of synthesis of a sound from a timbre space via a parameter representation has been shown only within the Matlab framework due to difficulties with the C++ implementation of the k-d tree. Parameters from Matlab can be used to drive the DXi synthesiser alongside MIDI data, showing that the synthesiser itself is capable of both modes of operation. The implementation in C++ would produce a faster and more integrated solution, but this is not necessary to demonstrate the effectiveness of the ideas presented in this thesis, so priority was given to extending the research instead of completing the C++ implementation.

The setbacks in the implementation have prevented the construction of a complete timbre space design prototype in C++. The experiments described in the next two chapters concentrate on the Matlab analysis and transformations of the data, but can be applied directly to a completed C++ implementation. The completion of this implementation, and the addition of other synthesisers, analysis methods and transformations, may enhance the novel results uncovered across the next two chapters, but add little scientific benefit towards the goal of demonstrating whether audio design is possible using a timbre space, since they merely provide generalisations of the existing results.

For further information on the technical details of the system, including all the information required to re-create the system described, see Appendix A for the interface definitions used to construct the objects, which is based in part on the DXi specification. Appendix B provides a sample parameter file demonstrating all the concepts listed in this chapter.

The system itself is not a complete design tool but the concept has been developed and a structure for such a tool has been presented. The research presented in the rest of this thesis presents an argument for the validity of the structure and discusses the technical and research challenges that must still be overcome to turn this proof of concept into a tool.

Chapter 5. Evaluation of timbre spaces

5.1 Introduction

The previous chapters demonstrated the need for a new audio design technique and have shown that the timbre space has features that may allow it to fit that need. In order to test the proposed concept, it is necessary to build and evaluate timbre spaces to see how well they fit the stated goals.

Neither Hourdin *et al.* [12] nor Kaminskyj [78] provide a methodology for choosing techniques for an automated timbre space and both use focussed justifications for their choice of timbre space without consideration of alternative techniques. Since neither paper discusses timbre spaces with respect to audio interface design, it is necessary here to compare a number of different timbre spaces to determine which of these is best for this purpose. This analysis provides a novel contribution to the field and will allow timbre spaces to be used more widely.

In addition to choosing a space, a number of possible novel enhancements to the space are tested, including testing whether the addition of phase information can improve the output quality from a timbre space.

Once a timbre space is chosen, it must be tested against the arguments raised against timbre spaces by Keller and Berger [72] that were considered in Section 3.6. Those criticisms are addressed by expanding the timbre space in various ways. The research adds new sounds differentiated from the original sounds by pitch and by timbre in order to test the hypothesis proposed by Keller and Berger that timbre spaces are poor at encoding such sounds. This procedure, and the addition of

Auditory Icon sounds to the timbre space as a design option, are novel contributions to the field.

To structure the tests, a number of hypotheses and evaluations are proposed in Section 5.2, looking at both subjective and objective tests. The subjective tests are examined in more detail in the following chapter. The choice of techniques to use for the timbre space is discussed in Section 5.3 with regard to a wide range of metrics, and the integrity of those metrics is considered.

To address the criticisms from Keller and Berger, Section 5.4 seeks to evaluate their assertions that the timbre space is poor at representing sounds across the frequency range of an instrument. An experiment is described where new sounds are added to the space to test the extremes of the frequency range for a number of instruments. New timbres are also added to the space to evaluate how well the timbre space generalises to new timbres.

The output from the timbre spaces is evaluated in Section 5.5 which compares the additive synthesis to the FM synthesis approach and considers the possibilities of adding new synthesisers to the space. Section 5.6 summarises the important results and concludes the chapter.

5.2 Hypotheses

The general hypotheses to be explored during the next two chapters are as follows. The methodology for testing these is reviewed in the next section. Each of the hypotheses below represents a novel extension to the existing body of work on timbre spaces and audio interfaces, and are designed to reflect the overall aims of the research. These hypotheses are as follows:

- CQT will produce better quality output than the STFT for the same size of timbre space since the logarithmic nature of the CQT fits closer with how the human ear works and so will capture more perceptually relevant data [58].

- It is possible to encode perceptual information as regions or transformations in the timbre space in such a way that they can be used by designers to define sounds with required properties. Properties which may be important in sound design may be, for example:
  - Does not interfere with speech;
  - Classifying a sound as 'hollow';
  - Classifying a sound as 'urgent'.

Speech tends to fall in the range 200-2,000 Hz [10]. If a non-speech interface is to be used where there is a lot of speech (e.g. alongside a screen reader or on a mobile phone whilst on a call), it may be desirable to avoid those frequencies when designing sounds. The timbre space could facilitate this by creating a region where sounds fall completely within the range of speech and another region where sounds have no frequency components in the range of speech. Sounds could then be compared with the two regions and modified appropriately.

The regions defined for the other classifications above should have properties matching those described by, for example, Howard [145] following work by Helmholtz [18]. One example cited by Howard of Helmholtz being:

> "If only the unevenly numbered partials are present... the quality of the tone is *hollow*, and, when a large number of such partials are present, *nasal*." [145, p71]

A hollow sound could be useful in a desktop interface for indicating an empty file or folder, or to indicate a broken link on a web page. Helmholtz describes a large number of such properties that may be useful for indicating state is similar ways.

Edworthy *et al.* [31] provide a definition of a signal rated by human participants as most urgent amongst a set of sounds. This definition is used in the next chapter to determine how such a concept may be encoded in a timbre space. In the desktop context, heightened urgency could be used to indicate a system file or folder that should be avoided. In an environment where each application has its own distinctive sound, increasing urgency could be used to indicate information, warning and error messages produced by the application.

- The timbre space is able to model existing interfaces, including:
  - An Auditory Icon enhanced interface;
  - An Earcon enhanced interface.

  The timbre space can interface with the MIDI specification which can be used to generate Earcons, and the timbre space is a superset of Gaver's Auditory Icons and other physical models. The former is covered by incorporating the timbre space into a synthesiser that receives MIDI commands, and by modelling how timbre changes across the frequency range of the instrument. Physical models are incorporated by parameter maps within the space that can be used to drive the physical models, or allow adjustment of the parameters to drive an alternative synthesiser.

- A timbre space representation for audio design can address the issues raised by Keller and Berger in their discussion comparing Granular Synthesis to timbre spaces.

  According to Keller and Berger [72], timbre space based synthesis, as used previously, makes a poor model for sound analysis and morphing for the following reasons:

1. Timbre space studies tend to take no account of the variation in timbre across the range of the instrument.
2. The sounds produced by a timbre space exist in a homogeneous virtual environment, i.e. one without environmental characteristics such as reverberation that have a large effect on the perceived timbre.
3. A timbre space is only good at recreating sounds similar to those used to create it.

This research will address all of these problems by:

1. Populating the space with a range of notes from each instrument to capture its full timbral range, and to use interpolation to generate in-between timbres. This interpolation can be linear or non-linear.
2. Creating the output synthesiser as a filter (in this case a DXi filter) allowing its output to be processed through environmental effects to provide, for example, reverberation and spatialisation.
3. Noting that the timbre space can be populated with existing Auditory Icons to allow it to replicate them.

Furthermore, the timbre space solution provides an easier to use interface to the underlying sounds, matching models of human perception more closely than Granular Synthesis, as suggested by the previous hypotheses.

5.2.1 Testing of hypotheses

In order to test the hypotheses listed, a number of general tests are useful. At the most basic level, a timbre space can be measured on the speed it takes to generate it from a fixed set of inputs and on the final size of the space consisting of the paths of those inputs. The storage requirements for the space can be used to determine whether a device can support a timbre space based audio interface, or requires a sample based one. The speed is mainly limited by the processing of the dimensionality reduction technique, so provides a measure of the complexity and variance of the timbres in the space. Where the variances in timbre are concentrated into a small number of dimensions, the space should converge much quicker.

The accuracy of any timbre space representation can be measured subjectively in terms of how closely an output matches a defined target. It is a useful measure for classifying the suitability of the timbre space representation and the mapping of the timbre space into parameter space. Although automated quality measures such as PEAQ (Perceptual Evaluation of Audio Quality) [123] exist, there are problems with the specification that have hindered implementation of the standard [124]. The objective accuracy measures used here are therefore cruder than the PEAQ and may over-estimate the perceptual error.

Representation accuracy

Objective tests of the analysis-synthesis approach can be performed by taking a recording and passing it though the system without alteration. The input and output files are then compared on a sample-by-sample basis to find the minimum, maximum and average error, as well as the variance.

If this error is calculated on the PCM data it gives a measure of the signal error, the variation the system introduces in the raw signal. The difference between the two signals is often called the *residual*, particularly within the analysis-synthesis techniques that separate the signal into smooth and noisy components. The residual taken between the original and re-constructed smooth signal is used to generate parameters for the noisy component [128, 64]. When calculating aggregates over this residual, the term 'residual error' will be used. The aggregates calculated are the mean, standard deviation and range of the residual.

This approach is very sensitive to noise and to differences in phase between the input and output sounds, and often over-estimates the perceived difference between two sounds. As an example, see Figure 5.1 where a Fourier approximation of a square wave is shown with and without a phase difference between the component sine waves. Although the two waves look very different and would have a large residual error, they sound very similar to the human ear. The use of statistical measures to determine the error means that the results may over-estimate the error compared to subjective tests, so relative comparisons between errors will be used so that results from all techniques will over-estimate the error in the same way. The results will be checked via informal listening tests, but the data set is too large for a formal subjective test across all sounds, so the subjective validity of the results cannot be formally verified.

A similar procedure can be used to determine the difference between the TFR representation of the input and the output. The TFR residue between the two can be analysed and summarised in a similar way to the residue across the two signals, and the aggregates produced on this will be known as the 'TFR error'. Whilst this error is less sensitive to phase differences, timbre spaces whose analysis method more closely matches the TFR used will have less error. This is caused by the way deviations from the assumptions of the analysis method will be modelled differently by each TFR. To combat this, each timbre space will be analysed by its own TFR. Although this produces the least error per timbre space, there is no way to generate a TFR which is not biased towards one particular timbre space.

If the error is calculated as the Euclidean difference between points in the timbre space, it will give a measure of the resynthesis error, which captures how closely the

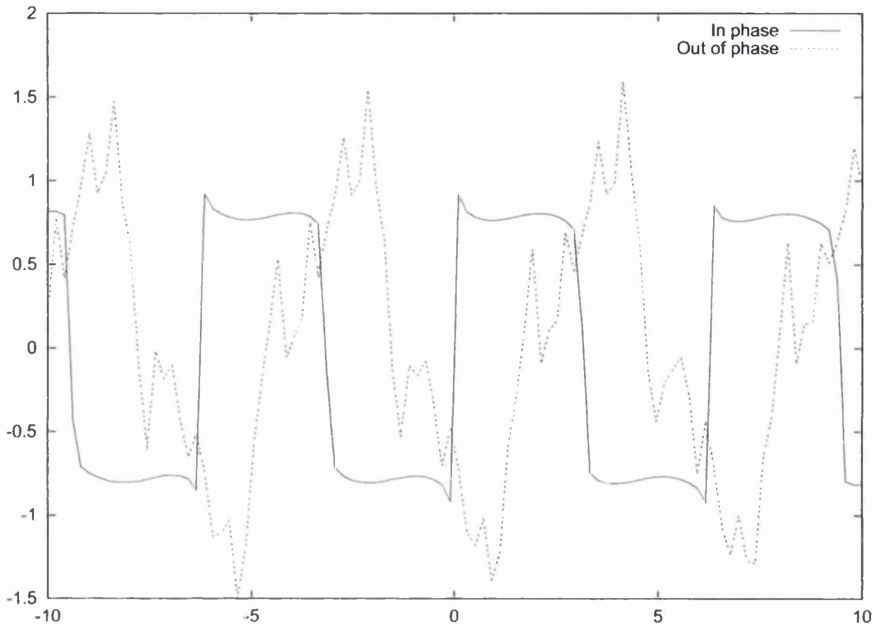


Figure 5.1: Two Fourier approximations to a square wave. In the "In phase" line, the sine waves are all in phase. In the "Out of phase" line, each sine wave has a different phase. Although both sound the same (taking the start and end clicks into account) and have similar Fourier Transforms, the signal error between the two is large due to the phase differences.

mapping and synthesis stages fit to the original analysis of the data. This error is sensitive to the timbre space analysis used and is likely to give much smaller errors if the error and the data use similar analysis. This may be useful for testing alternative synthesis models but is a fairly meaningless measure for comparing timbre spaces.

All these errors can be calculated as absolute values, or as relative values, taking the percentage variation of the output from the input, with the largest absolute input value as the origin for calculating the percentage. Storing the absolute values allows generation of relative values using the ratio between the max of the input and the max of the output. Only the absolute values will be stored and processed since the relative values are a trivial extension.

Subjective perceptual tests

Once perceptual regions are defined in the timbre space, their effectiveness must be tested by asking participants to rate sounds. These sounds should be defined in the space according to a perceptual construct such as 'urgency'. The participants rate the sound according to their perception of 'urgency' in order to determine if the transformations in the timbre space relating to the perceptual construct produce the required perceptual change in the sound. Full details of such an experiment can be found in Chapter 6.

5.3 Choosing a timbre space

There are a wide range of techniques and parameters that can be used to generate a timbre space. This section seeks to test a subset of them to determine a good timbre space to choose as a basis for audio design. Analysis has been performed on a range of musical and synthetic signals including some output from the DXi FM synthesiser itself. The spaces themselves have been constructed from a set of 27 sounds taken from the MUMS CDs [131] chosen to match as closely as possible the sounds used by Grey [13] and Hourdin *et al.* [12]. Many of Hourdin *et al.*'s sounds were taken from this source. The sounds chosen and their internal names (taken from Hourdin *et al.*) are shown in Table 5.1. Unlike the previous work, which used $\frac{1}{4}$ second to 1 second samples from the steady portion of each sound, the whole sound is used here in order to test the ability of the timbre space to capture the transient portion of the sound. For many sounds, such as percussive sounds, there is no steady state portion. A timbre space that cannot model transients is no good for modelling Auditory Icons such as taps that are characterised by harmonics that decay at different rates so the timbre changes dramatically over the length of the sound.

Number	Short Name	CD num	Track num	Sample name	Note
Sounds from Grey					
1	BN	2	14	bassoon	D#4
2	C1	2	11	e flat clarinet	D#4
3	C2	2	12	bass clarinet	D#3
4	EH	2	9	English horn	D#4
5	FH	2	19	French horn	D#4
6	FL	2	1	flute, vibrato	D#4
7	O1	2	8	oboe	D#4
8	O2	*	*	*	*
9	S1	*	*	*	*
10	S2	1	11	cello, bowed, with vibrato	D#4
11	S3	*	*	*	*
12	TM	2	17	c tpt., Harmon,stem out	D#4
13	TP	2	16	c trumpet	D#4
14	X1	3	16	alto saxophone	D#4
15	X2	*	*	*	*
16	X3	3	17	soprano saxophone	D#5
Sounds from Hourdin *et al.*					
17	Ar	*	*	*	*
18	Bp	1	18	double bass, pizzicato	D#4
19	Ff	2	2	flute, flutter	D#4
20	Fy	2	6	bass flute, vibrato	D#4
21	Fz	2	5	alto flute, vibrato	D#4
22	Gh	*	*	*	*
23	Gp	*	*	*	*
24	Gt	*	*	*	*
25	Ha	*	*	*	*
26	Hc	*	*	*	*
27	Lw	1	6	viola, bowed, with vibrato	D#4
28	Ma	*	*	*	*
29	Om	*	*	*	*
30	Or	*	*	*	*
31	Pf	3	2	9' Hamburg Steinway, loud	D#4
32	Pm	3	1	9' Steinway, soft	D#4
33	Sm	1	15	cello, martelé	D#4
34	Sp	1	13	cello, pizzicato	D#4
35	TC	2	16	c trumpet	D#4
36	T1	3	10	tubular bells	D#4
37	Tt	2	22	tenor trombone	D#4
38	Tu	2	25	tuba	D#4
39	Vh	3	7	vibraphone, bowed	D#4
40	Xy	3	5	xylophone	D#5

Table 5.1: Short names, source CD and track, pitch and full name for each sound used in the timbre space experiments. All sounds used by Grey [13] and by Hourdin *et al.* [12] are listed, but only some are available on the MUMS CDs [131]. Those not available are noted by asterisks. All sounds are chosen to match the pitch used in those papers. The short names for the sounds are taken from Hourdin *et al.* and the full names are taken from the track listings for the MUMS CDs.

Versions of the STFT and CQT algorithms have been tested with various parameter settings using Gaussian windows, two Kaiser windows, and no window, all with varying overlaps between time steps. Both methods take several minutes to complete analysis once the input signal is longer than 400,000 samples (or 9 seconds at a 44.1kHz sampling rate). All tests have been done with the PCA algorithm.

5.3.1 Test suite of timbre spaces

A set of functions has been developed to automatically generate a set of timbre spaces so that their speed, size and accuracy of reproduction can be compared. These experiments are slow however as each timbre space takes as much as 4 days to compile. Once the space is compiled, it takes under 5 minutes to process each sound through to its path representation, and some sounds can be synthesised from the space using additive synthesis in a time frame shorter than the length of the sound, suggesting that real-time manipulation and synthesis is possible, even when other processes are running.

The options presented in Chapter 3 have been reviewed in order to provide a number of plausible timbre spaces to explore. The test suite covers CQT and STFT based timbre spaces with 4 window conditions: none, Gaussian window, Kaiser 6 and Kaiser 8 (see Chapter 3 for full details of these). It covers 2ms, 4ms (as used by Hourdin *et al.*) and 400ms window sizes with window overlap of 1 (i.e. no overlap), 2 or 4 times, and covers 3 frequency resolutions for the CQT, one equivalent to 2 bins per octave, one for 12 bins per octave, (i.e. standard western musical scale) and one for 24 bins per octave, this provides 36 different Fourier-based timbre spaces and 108 different CQT-based timbre spaces.

Only experiments using PCA scaling have been implemented, and it is this stage of the analysis that takes up most of the processing time. The EM-PCA procedure

Window Length (ms)	Window Type	Overlap	Analysis Type			
			STFT	CQT Resolution (Notes per octave)		
				2	12	24
2 (short)	boxcar	1	fsb_1_8	clsb_1_8	cnsb_1_8	chsb_1_8
		2	fsb_2_8	clsb_2_8	cnsb_2_8	chsb_2_8
	gaussian	1	fsg_1_8	clsg_1_8	cnsg_1_8	chsg_1_8
		2	fsg_2_8	clsg_2_8	cnsg_2_8	chsg_2_8
	kaiser 6	1	fsk6_1_8	clsk6_1_8	cnsk6_1_8	chsk6_1_8
		2	fsk6_2_8	clsk6_2_8	cnsk6_2_8	chsk6_2_8
	kaiser 8	1	fsk8_1_8	clsk8_1_8	cnsk8_1_8	chsk8_1_8
		2	fsk8_2_8	clsk8_2_8	cnsk8_2_8	chsk8_2_8
4 (medium)	boxcar	1	fmb_1_8	clmb_1_8	cnmb_1_8	chmb_1_8
		2	fmb_2_8	clmb_2_8	cnmb_2_8	chmb_2_8
	gaussian	1	fmg_1_8	clmg_1_8	cnmg_1_8	chmg_1_8
		2	fmg_2_8	clmg_2_8	cnmg_2_8	chmg_2_8
	kaiser 6	1	fmk6_1_8	clmk6_1_8	cnmk6_1_8	chmk6_1_8
		2	fmk6_2_8	clmk6_2_8	cnmk6_2_8	chmk6_2_8
	kaiser 8	1	fmk8_1_8	clmk8_1_8	cnmk8_1_8	chmk8_1_8
		2	fmk8_2_8	clmk8_2_8	cnmk8_2_8	chmk8_2_8
400 (long)	boxcar	1	flb_1_8	cllb_1_8	cnlb_1_8	chlb_1_8
		2	flb_2_8	cllb_2_8	cnlb_2_8	chlb_2_8
	gaussian	1	flg_1_8	cllg_1_8	cnlg_1_8	chlg_1_8
		2	flg_2_8	cllg_2_8	cnlg_2_8	chlg_2_8
	kaiser 6	1	flk6_1_8	cllk6_1_8	cnlk6_1_8	chlk6_1_8
		2	flk6_2_8	cllk6_2_8	cnlk6_2_8	chlk6_2_8
	kaiser 8	1	flk8_1_8	cllk8_1_8	cnlk8_1_8	chlk8_1_8
		2	flk8_2_8	cllk8_2_8	cnlk8_2_8	chlk8_2_8

Table 5.2: Names used for the 108 timbre spaces tested. Spaces with an offset of 4 are omitted for clarity. Each sound generated from the space has this name appended to it for reference purposes. The first letter defines the analysis method, with the second letter of the CQT defining the resolution. The next two letters define the window length and window type respectively, with a numeric argument appended for the Kaiser window type. An underscore separates the window type from a number representing the overlap, in terms of number of windows that start within the given window length. The final number defines the dimensionality of the timbre space. Each timbre space was only tested in 8 dimensions since no reliable data on coverage could be collected.

is a slow iterative process, but due to its lower memory requirements it is able to complete on the test machines whereas the standard PCA algorithm fails when it runs out of memory whilst generating a matrix of covariances between the points in the frequency space. For the 4ms windows, this matrix consists of 32000 rows and 32000 columns, representing the 32000 time steps required to cover the 27 input sounds with the 4ms window. The nature of the EM-PCA algorithm means that the eigenvalues for all 80 possible timbre space dimensions are unavailable, so it is impossible to compute the coverage of the space as done by Hourdin *et al.* [12] for the FAC algorithm. Due to this, their estimate of 8 timbre-space dimensions is used.

5.3.2 Speed and size of spaces

All the STFT spaces with a 400ms window size failed since the Fourier analysis required too much memory. These spaces were not considered further. In order to reduce the amount of data produced, only 2 overlap conditions were considered for the CQT spaces. In the boxcar condition, an overlap of 1 and 2 was considered. In the other window conditions, an overlap of 2 and 4 was considered. These conditions reflect the commonly used overlaps for the boxcar and non-boxcar windows in the analysis-synthesis literature [16]. The results presented later show that the offset has little effect on reproductive quality. The number of spaces compared was therefore reduced to 96. The spaces take between 2 minutes and 3 days to compile, as shown by Figure 5.2, and require between 0.1Mb and 500Mb of storage to contain the TFRs for all 27 input sounds. The original 27 stereo WAV files take up 22.3 MB of space. The data for the graph can be found in Appendix C.

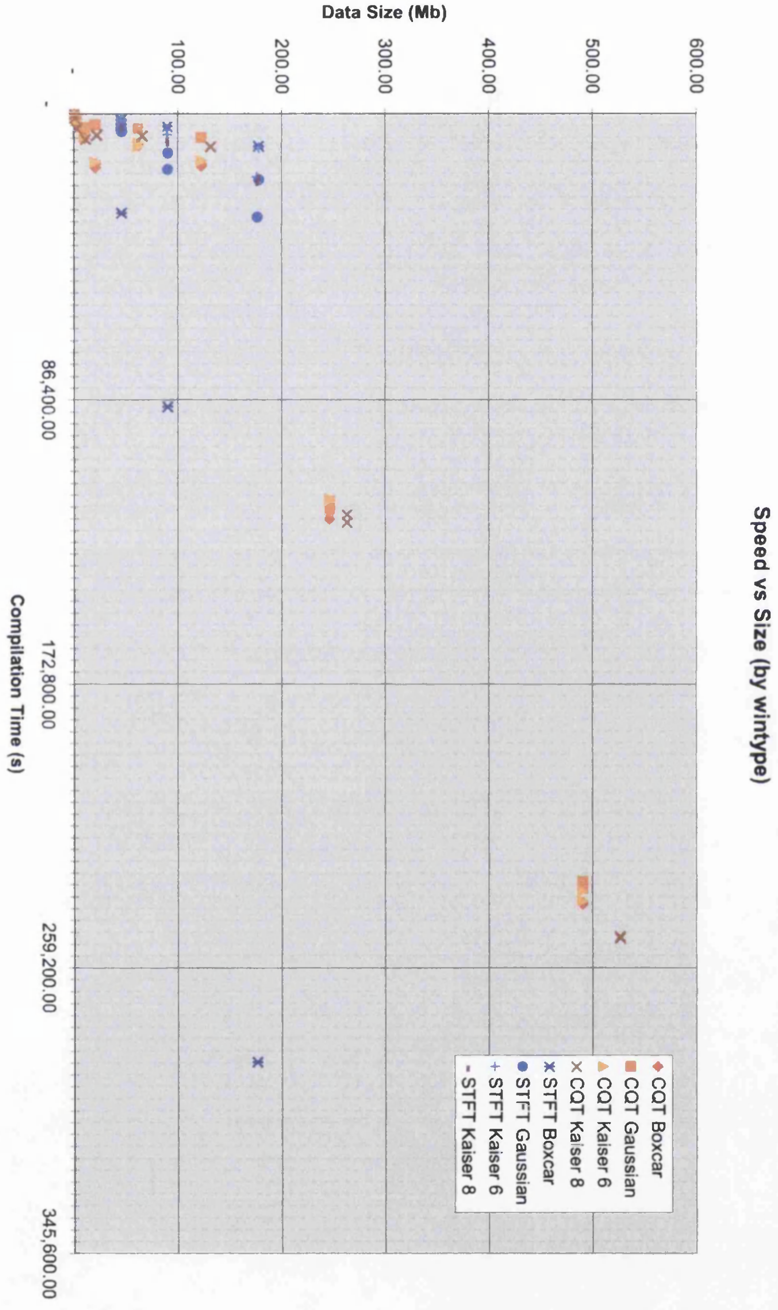


Figure 5.2: Graph of speed of construction against construction size of the timbre spaces. The speed is measured as clock time. The major lines indicate days and the minor lines indicate hours. The size is the amount of data required to generate the space, i.e. the size of the TFR. The final size of the space is 7 to 15 times smaller depending on the frequency resolution of the TFR.

5.3.3 Accuracy of sound reproduction

All the timbre spaces have been evaluated with respect to their accuracy in recreating the original 27 MUMS sounds listed in Table 5.1. These reproductions have been made available on the web-site and in the CD accompanying this thesis, described in Appendix E.

The following discussion of the evaluation of these sounds is in the context of the reproduction quality of the sounds evaluated by informal listening tests by the author and presented to conference attendees. Formal evaluation of the sound quality using human participants comparing the 27 input sounds with all 2,592 output sounds across all the timbre spaces was impractical. No timbre space produced perfect re-synthesis of the original input signal and all introduced some noise, making an objective measure of closest sounds very difficult. On this basis, the spaces were evaluated on sounds which were 'good enough' in that the output sound was audible above any noise and the output sounds were easily distinguishable. Any experiment seeking to ask participants to rate the pairs of sounds which best fit those two criteria are likely to suffer a subjective bias and the best trade-off between speed, size and quality may depend on the individual listener. To support the subjective results, a number of objective tests were performed to systematically compare all the sounds by comparing the error introduced by each timbre space. The timbre spaces with the smallest error have the best reproductive quality.

5.3.4 General method for timbre space comparisons

The two standard tests for determining distinct distributions are the t-test and the Mann-Whitney, both described in Dancey and Reidy [146]. The t-test is used where the two sets to be compared each have a normal distribution and the Mann-Whitney test is used otherwise. Initial results were used to determine the distribution of the errors. None of the error measures discussed below exhibit a normal distribution, so a two-tailed Mann-Whitney test has been used for all comparisons.

Every sound is filtered through every timbre space and the error between the output and input are taken as follows:

1. The residue is calculated. This is the difference between the input and output wave. It is represented as a 1-column matrix.

2. The TFR error is calculated. This is the difference between the TFR matrix for the input and the output waves. generated using the same parameters as the current timbre space.

3. The following aggregates over the residue and TFR error are taken:

   (a) **Min, Max** The minimum and maximum values present in the error matrix.

   (b) **Mean** The mean value across all the raw values in the error matrix.

   (c) **Standard Deviation (Stddev)** The standard deviation across all the raw values in the error matrix.

   (d) **Root-mean Squared (RMS) error** A measure of the absolute mean across all the values in the error matrix. If half the errors are negative and half are positive, the mean error will be close to zero whereas the RMS will be close to the overall magnitude of the error.

The errors for each condition are compared to determine which has the best reproduction overall, and the results have been checked informally by subjective listening using the sounds that are available on CD included with this thesis (see Appendix E). All of the following analysis was performed using MINITAB. Since the error between the input and output sounds is measured, values with a lower magnitude show better reproduction. Minimum, maximum and mean errors can be negative.

In the raw results, the output signal of the STFT has 14 times the amplitude of the output signal from the CQT. All signals are scaled to an amplitude of 1 so that a fair comparison can be made between the signal quality of each technique. The results in the following tables show the errors calculated after this scaling has been applied.

5.3.5 Effect of technique

A number of analysis techniques were discussed in Section 3.3.1. From these, the Short-Time Fourier Transform (STFT), the Constant Q Transform (CQT) and an auto-regression (AR) analysis were selected for further investigation due to their compatibility with the aims of the project. The techniques are compared here in the context of timbre spaces. This section concentrates on the STFT and CQT techniques due to problems with the AR analysis discussed at the end of this section.

Hypothesis: The CQT analysis will produce spaces with better reproduction than the STFT analysis. The CQT and the human ear both have a logarithmic frequency scale, so the CQT should capture perceptual data more efficiently than the STFT.

Null hypothesis: The CQT will not have significantly better sound reproduction than the STFT.

Method

The spaces were divided into two groups, representing the spaces constructed from a CQT analysis and the spaces constructed from an STFT analysis. A Mann-Whitney test was performed on the errors across the two groups.

Results

Measure	STFT	CQT Resolution (bins per octave)		
		Low (2)	Medium (12)	High (24)
Residue Errors				
Min ($\times 10^{-3}$)	-1.0	**-3.2 (< 0.001)**	**-2.7 (< 0.001)**	**-2.6 (< 0.001)**
Max ($\times 10^{-3}$)	1.0	1.0 (0.97)	1.0 (0.98)	1.0 (0.98)
Mean ($\times 10^{-3}$)	0.0	**0.04 (< 0.001)**	**0.05 (< 0.001)**	**0.05 (< 0.001)**
Stddev ($\times 10^{-3}$)	0.11	**0.15 (< 0.001)**	**0.15 (< 0.001)**	**0.15 (< 0.001)**
RMS ($\times 10^{-3}$)	0.11	**0.16 (< 0.001)**	**0.16 (< 0.001)**	**0.17 (< 0.001)**
TFR Errors				
Min ($\times 10^{-3}$)	-0.00	**-1.1 (< 0.001)**	**-1.5 (< 0.001)**	**-3.3 (< 0.001)**
Max ($\times 10^{-3}$)	76	**37 (< 0.001)**	**39 (< 0.001)**	42 (0.81)
Mean ($\times 10^{-3}$)	0.12	**1.3 (< 0.001)**	**1.2 (< 0.001)**	**1.4 (< 0.001)**
Stddev ($\times 10^{-3}$)	1.6	**4.9 (< 0.001)**	**4.8 (< 0.001)**	**5.7 (< 0.001)**
RMS ($\times 10^{-3}$)	1.6	**5.1 (< 0.001)**	**5.0 (< 0.001)**	**6.0 (< 0.001)**

Table 5.3: Mann-Whitney comparison of errors between the STFT and CQT timbre spaces. Numbers show the median calculated for each group, rounded to 2 significant figures. Numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between each CQT resolution and the STFT space. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics*.

	CQT Resolutions		
Measure	High vs Medium	Medium vs Low	High vs Low
Min	**h > m (< 0.001)**	**m > l (< 0.001)**	**h > l (< 0.001)**
Max	**h > m (< 0.001)**	*m > l (0.023)*	**h > l (< 0.001)**
Mean	*h > m (0.011)*	m < l (0.23)	h > l (0.18)
Stddev	**h > m (< 0.001)**	m < l (0.60)	**h > l (< 0.001)**
RMS	**h > m (< 0.001)**	m < l (0.56)	**h > l (< 0.001)**

Table 5.4: Mann-Whitney comparison of the TFR errors between the CQT timbre spaces by CQT resolution. There were no significant results between CQT resolutions for the residue error so those p-values are not shown. Numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between each CQT resolution. The letters in each cell refer to the two resolutions being compared, and represent the median value of the measure across all spaces with the given resolution. Actual values are given in Table 5.3. The < and > symbols indicate whether the magnitude of the error for the first listed resolution is greater or less than the magnitude of the error for the second listed resolution. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics*.

The median values calculated for the Mann-Whitney test are provided in Table 5.3 and p-values for the CQT comparisons only are provided in Table 5.4. The residue errors show significant differences between the STFT and CQT medians at the 99.9% level for all measures except the maximum and for all cases the CQT spaces show a greater magnitude of error. The TFR errors follow the same pattern except the maximum TFR error is greater for the STFT than the CQT spaces and this difference is significant at the 99.9% level for 2 of the 3 CQT resolutions.

Amongst the CQT resolutions, the highest resolution has a bigger TFR error than the other resolutions and the difference is significant at the 99.9% level for all measures except the mean. The significant differences at the 95% level between the medium and low resolutions suggest that the medium resolution has a greater error than the low resolution but only the minimum error shows a significant difference between the resolutions at the 99.9% level. There were no significant differences between the CQT resolutions on any of the residual error measures.

Discussion

The timbre spaces formed from CQT analysis have poorer reproduction than those formed from STFT analysis. This is a surprising result as the hypothesis suggested CQT would capture better information in terms of human perception as it has a logarithmic frequency scale, similar to the human ear. However, the CQT results are worse than STFT timbre spaces across the board. It could be that the logarithmic nature of the CQT disrupts the representation of the harmonics of the sound, which are direct multiples of (i.e. linearly related to) the fundamental frequency. This poor representation of harmonics may lead to lower co-variance across the TFR, causing the PCA to spread the variance across more dimensions than in the STFT case. As the EM calculation of the PCA does not produce eigenvalues, it is not possible to test this hypothesis within the existing framework.

When the spaces are broken down by CQT resolution, shown in Table 5.4, there are no significant results for the residue error. The TFR error results show that for all measures except the mean error, the higher resolution has errors of a higher magnitude than the other resolutions, and this difference is significant at the 99.9%

level. This is supported by informal listening tests where the produced sound bears little resemblance to the input. Since the only significant difference between the lower two resolutions at the 99.9% level is for the minimum TFR error, this result does not appear to be a general trend in the resolution of the CQT. This suggests that the highest resolution of the CQT may be too high for the time resolution, and so violates the trade-off between time and frequency resolutions. Subjective listening to the output sounds suggests that this violation leads to the introduction of a large amount of noise that swamps the original signal.

Fractional Fourier Transforms offer no advantage over the STFT in the general case as it was designed for linearly decaying frequency signals. Few environmental sounds have those characteristics. Timbre spaces constructed using the Fractional Fourier Transform are slower to construct that the STFT and appear to introduce more noise into the output signal so have not been investigated further.

Auto-regression analysis has been studied but the implementation originally used, from the Voicebox speech processing toolbox [147], fails when the analysis window contains silence. Attempts to resolve this by checking for division by zero allowed the matrix to be analysed by the dimensionality stage of the algorithm, but produced warnings that the determinant of the matrix was close to zero. This low determinant suggests that the eigenvalues of such a matrix would show little variance and hence would require a high dimensional timbre space, close to the dimensionality of the TFR, to represent enough variation in the sound to get a reasonable quality signal. The analysis using this representation is hence invalid for a timbre space with few dimensions.

A new implementation has been found which appears to handle the case where silence is analysed more gracefully. This implementation is part of the Auditory Toolbox for Matlab [148]. Experiments on this new space also create determinant close to zero errors, suggesting the TFR generated by the technique may be unsuitable as input for the PCA algorithm. Auto-Regression analysis adds frequency dimensions to its TFR in addition to the amplitude dimensions found in the TFRs of the STFT and CQT analyses. The dependence between these frequency dimensions and the amplitude dimensions violates the assumption of the PCA that all the input dimensions are independent. It would appear that the raw structure of the TFR produced by the AR technique is unsuited for timbre spaces constructed using PCA. Since there are no other papers which seek to use AR to construct a timbre space, there is no independent verification of this result.

The problems with the auto-regression analysis suggest that the implementation of a signal and noise model such as Stochastic Modelling Synthesis [128] is not possible using PCA, since their output is similar to that of the AR technique, requiring storage of both a frequency and an amplitude dimension for each frequency component in the sound. For this reason, no signal and noise model was explored.

5.3.6 Effect of phase

Most of the experiments in this chapter have ignored the effect of phase information on the basis that phase information cannot generally be heard, as noted by Risset and Wessel [23] (see Section 2.2.3). A 440Hz sine wave sounds identical to a 440Hz cosine wave once the effects of clicks in the sound are addressed. Whilst the other spaces were being generated, phase information was added to the set of Kaiser 8 timbre spaces, which were the first to be generated, to see if it was worth exploring phase further.

Hypothesis: A timbre space that includes phase information will have better reproductive quality, as measured by these tests, than a space that does not include phase information, since the errors are sensitive to differences in phase. Furthermore, Risset and Wessel [23] suggest that phase is important in the timbre of complex sounds.

Null hypothesis: There will be no significant difference in sound quality between spaces with phase and spaces without.

Method

The parameters for the Kaiser 8 STFT timbre spaces were used to generate timbre spaces that included phase information. All spaces had 8 dimensions and all other parameters were chosen to match those for the spaces without phase information. A Mann-Whitney test was performed on the errors across the two groups.

Results

Measure	phase	no phase	p-value
Residue Errors			
Min ($\times 10^{-3}$)	-1.0	-0.99	0.99
Max ($\times 10^{-3}$)	1.0	1.0	1.00
Mean ($\times 10^{-3}$)	0.0	0.0	< **0.001**
Stddev ($\times 10^{-3}$)	0.16	0.11	< **0.001**
RMS ($\times 10^{-3}$)	0.16	0.11	< **0.001**
TFR Errors			
Min ($\times 10^{-3}$)	-0.34	-0.00	< **0.001**
Max	1.2	0.067	< **0.001**
Mean ($\times 10^{-3}$)	2.8	0.13	< **0.001**
Stddev ($\times 10^{-3}$)	1.8	0.16	< **0.001**
RMS ($\times 10^{-3}$)	1.9	0.16	< **0.001**

Table 5.5: Mann-Whitney comparison of errors between the timbre spaces that include phase and the spaces that do not include phase. Numbers show the median calculated for each group, rounded to 2 significant figures. Numbers in the last column indicate the p-value, rounded to 2 s.f., for the comparisons between the two conditions. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics.*

The results for the experiment are shown in Table 5.5. All comparisons in Table 5.5, except for the minimum and maximum residue error, show a significant difference at the 99.9% level between the spaces with and without phase information. For all these comparisons, the spaces that do not use phase show a lower error than the spaces that do include phase.

Discussion

In contrast to the hypothesis, the addition of phase information appears to increase the error of the reproduced signal. Subjective listening tests confirm that the results from spaces with phase information had more noise in general than the STFT without phase information since the PCA algorithm has twice the number of inputs and the same number of outputs. Despite the higher level of noise overall, the subjective results from the spaces with phase information suggest that the transient signals that correspond to the window edges were reduced. Overall, the results do not support the addition of phase information to the timbre space.

Another method for reducing transients is smoothing, discussed in Section 5.5.1. This method can remove transient and non-transient noise and does not require the extra input data required to model phase. Smoothing is also a more general technique that can work on any analysis model. These advantages, and the results presented here, mean that smoothing should be used to reduce output noise in preference to modelling phase.

5.3.7 Effect of windows

A number of windows were discussed in Section 3.3.1 as a method for reducing noise in the output for various analysis techniques. The boxcar, Gaussian, Kaiser 6 and Kaiser 8 windows were applied to the STFT and CQT techniques to determine which was most suited for use in a timbre space for audio design.

Hypothesis: The boxcar technique will have the highest reproduction error since it introduces the most noise. The Kaiser 8 should have the lowest error. This is generally true for Analysis-Synthesis techniques [81].

Null hypothesis: There will be no difference in reproductive quality between window types.

Method

The spaces were divided into 4 groups according to the window types used. A pairwise comparison was performed between each of the groups, using the Mann-Whitney test to determine if each window type produced significantly different errors from the others.

Results

Measure	Kaiser 8	Kaiser 6	Gaussian	Boxcar
Residue Errors				
Min ($\times 10^{-3}$)	-0.41	-0.48 (0.78)	-0.39 (0.16)	**0.64 (< 0.001)**
Max ($\times 10^{-3}$)	1.0	1.0 (0.98)	1.0 (0.98)	1.0 (0.98)
Mean ($\times 10^{-3}$)	0.00	0.01 (0.99)	0.01 (0.79)	0.01 (0.32)
Stddev ($\times 10^{-3}$)	0.15	0.15 (0.82)	*0.13 (0.0094)*	*0.16 (0.0054)*
RMS ($\times 10^{-3}$)	0.15	0.15 (0.81)	*0.13 (0.0099)*	*0.17 (0.0016)*
TFR Errors				
Min ($\times 10^{-3}$)	-0.93	-0.91 (0.94)	-0.87 (0.17)	-1.2 (0.93)
Max ($\times 10^{-3}$)	37	40 (0.084)	**28 (< 0.001)**	**86 (< 0.001)**
Mean ($\times 10^{-3}$)	0.78	0.87 (0.11)	**0.56 (< 0.001)**	**2.6 (< 0.001)**
Stddev ($\times 10^{-3}$)	3.5	*4.1 (0.048)*	**2.5 (< 0.001)**	**8.1 (< 0.001)**
RMS ($\times 10^{-3}$)	3.7	4.2 (0.054)	**2.5 (< 0.001)**	**8.7 (< 0.001)**

Table 5.6: Mann-Whitney comparison of errors between the Kaiser 8, Kaiser 6, Gaussian and Boxcar timbre spaces. Numbers show the median calculated for each group, rounded to 2 significant figures. Numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between the Kaiser 8 timbre spaces and the other spaces. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics.*

The median values calculated for the Mann-Whitney test are provided in Table 5.6 and p-values for the Kaiser 6, Gaussian and Boxcar comparisons are provided in Table 5.7. In 7 out of the 10 measures the boxcar exhibits a greater error than either Kaiser window with a significant difference at the 95% level. Five of these

Measure	Kaiser 6 vs Gaussian	Kaiser 6 vs Boxcar	Gaussian vs Boxcar
Residue Errors			
Min	k > g (0.090)	**k < b (< 0.001)**	**g < b (< 0.001)**
Max	k ≈ g (0.99)	k ≈ b (0.98)	g ≈ b (1.00)
Mean	k ≈ g (0.82)	k ≈ b (0.32)	g ≈ b (0.29)
Stddev	*k > g (0.0047)*	*k < b (0.0091)*	**g < b (< 0.001)**
RMS	*k > g (0.0047)*	*k < b (0.0033)*	**g < b (< 0.001)**
TFR Errors			
Min	k > g (0.24)	k < b (0.92)	g < b (0.30)
Max	**k > g (< 0.001)**	**k < b (< 0.001)**	**g < b (< 0.001)**
Mean	**k > g (< 0.001)**	**k < b (< 0.001)**	**g < b (< 0.001)**
Stddev	**k > g (< 0.001)**	**k < b (< 0.001)**	**g < b (< 0.001)**
RMS	**k > g (< 0.001)**	**k < b (< 0.001)**	**g < b (< 0.001)**

Table 5.7: Mann-Whitney comparison of the TFR errors between the Kaiser 6, Gaussian and Boxcar windows. Numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between each window type. The letters in each cell refer to the windows being compared, and represent the median value of the measure across all spaces with the given resolution. Actual values are given in Table 5.6. The < and > symbols indicate whether the magnitude of the error for the first listed window type is greater or less than the magnitude of the error for the second listed window type, ≈ indicates that the two values are approximately equal (within the bounds of significance returned by MINITAB). Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics*.

are significant at the 99.9% level. The Gaussian window shows lower errors than the boxcar that are significant at the 99.9% level for 7 out of 10 measures. The Gaussian window shows lower errors than both Kaiser windows and shows a significant difference at the 95% level for 6 of the 10 measures. Four of these differences are significant at the 99.9% level. There is only one significant difference at the 95% level for the Kaiser windows, with the standard deviation of the TFR error being lower for the Kaiser 8 than the Kaiser 6 window.

Discussion

As expected, the boxcar windows produce the worst quality output, as shown by Table 5.6 and Table 5.7. The spaces using the boxcar windows have a higher error than the spaces using the other windows, significant at the 99.9% level, for a large number of the measures used in both the residue and TFR error results. This performance is due to the fact that they introduce more transient data into the signal at the edge of the windows. These transients often appear in the final signal at the same frequency as the window offsets, i.e. if each window onset is separated by 2ms, the TFR will contain a strong signal at 500Hz.

There is little difference between the two variations of the Kaiser window, with the only significant result at the 95% level showing that standard deviation of the TFR error is lower for the Kaiser 8 window than the Kaiser 6. The errors for the Gaussian window are less than for the other windows, and the difference is significant at the 95% level for the standard deviation and RMS error against the Kaiser windows and significant at the 99.9% level for the minimum, standard deviation and RMS of the residue error against the boxcar window, and for all measures for all other windows for the TFR error except for the minimum TFR error. These results suggest that the best window to use, amongst those tested, is the Gaussian window.

5.3.8 Effect of window length

Three window lengths were compared to see which offered the best reproduction after a sound was filtered through the timbre space.

Hypothesis: Shorter window lengths will produce better quality results since they will capture transients in the sound more effectively.

Null hypothesis: There will be no difference in reproductive quality between window length.

Method

The spaces were divided into 5 groups according to the window lengths used, 2 for the STFT spaces, with the longest window length discarded, and 3 for the CQT spaces. A pairwise comparison was performed between each of the groups, using the Mann-Whitney test to determine if each window length produced significantly different errors from the others.

Results

Measure	Window length			p-value
	2ms	4ms	400ms	4ms vs 400ms
		Residue Errors		
Min ($\times 10^{-3}$)	-0.12	**-0.21 (< 0.001)**	**-0.75 (< 0.001)**	**< 0.001**
Max ($\times 10^{-3}$)	1.0	1.0 (0.91)	1.0 (0.95)	0.94
Mean ($\times 10^{-3}$)	0.07	*0.07 (0.0010)*	**0.00 (< 0.001)**	**< 0.001**
Stddev ($\times 10^{-3}$)	0.17	0.16 (0.56)	**0.13 (< 0.001)**	**< 0.001**
RMS ($\times 10^{-3}$)	0.19	0.18 (0.29)	**0.13 (< 0.001)**	**< 0.001**
		TFR Errors		
Min ($\times 10^{-3}$)	-1.3	**-2.0 (< 0.001)**	**-2.1 (< 0.001)**	0.051
Max ($\times 10^{-3}$)	23	**33 (< 0.001)**	**30 (< 0.001)**	**< 0.001**
Mean ($\times 10^{-3}$)	0.91	**1.3 (< 0.001)**	**2.5 (< 0.001)**	**< 0.001**
Stddev ($\times 10^{-3}$)	3.1	**4.6 (< 0.001)**	**18 (< 0.001)**	**< 0.001**
RMS ($\times 10^{-3}$)	3.2	**4.8 (< 0.001)**	**19 (< 0.001)**	**< 0.001**

Table 5.8: Mann-Whitney comparison of errors between window lengths for the CQT timbre spaces. Numbers show the median calculated for each group rounded to 2 significant figures. The numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between the errors of the 2ms spaces and the spaces for the other two lengths. The last column indicates the p-value, rounded to 2 s.f., for the comparisons between the spaces with a 4ms window and the spaces with a 400ms window. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics.*

The STFT results are only shown over 2 different window lengths as the 400ms windows generated memory errors. Since the CQT and STFT results both show a trend that smaller windows have smaller errors, further tests on windows lengths longer than 4ms for the STFT were not performed. For the CQT spaces in Table 5.8, the 2ms window length shows a smaller error than the 4ms window for all 6 significant results at the 99.9% level, and shows a smaller error than the 400ms window for 6 of the 9 significant results at the 99.9% level, with the 400ms window exhibiting smaller errors for the other 3 measures. The 4ms window length shows lower errors than the 400ms window in 4 of the 8 measures showing significant differences at the 99.9% level, with the 400ms window showing lower errors in the other 4 measures. For the STFT spaces in Table 5.9, all the TFR measures except the minimum error

Measure	Window length		p-value
	2ms	4ms	
Residue Errors			
Min ($\times 10^{-3}$)	-1.0	-0.99	0.24
Max ($\times 10^{-3}$)	1.0	1.0	0.17
Mean ($\times 10^{-3}$)	0.0	0.0	**< 0.001**
Stddev ($\times 10^{-3}$)	0.10	0.11	0.35
RMS ($\times 10^{-3}$)	0.10	0.11	0.35
TFR Errors			
Min ($\times 10^{-3}$)	-0.01	-0.00	**< 0.001**
Max ($\times 10^{-3}$)	52	100	**< 0.001**
Mean ($\times 10^{-3}$)	0.09	0.16	**< 0.001**
Stddev ($\times 10^{-3}$)	1.1	2.1	**< 0.001**
RMS ($\times 10^{-3}$)	1.1	2.1	**< 0.001**

Table 5.9: Mann-Whitney comparison of errors between window lengths for the STFT timbre spaces. Numbers show the median calculated for each group rounded to 2 significant figures. The last column indicates the p-value, rounded to 2 s.f., for the comparisons between the two values of window length. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics*.

show the 2ms window with a smaller error than the 4ms window. All the TFR errors are significant at the 99.9% level.

Discussion

The results in Table 5.8 and Table 5.9 demonstrate that shorter window lengths produce lower errors. The errors are significantly different at the 99.9% level for all comparisons of TFR errors across STFT and CQT spaces except for the minimum TFR error between the 4ms and 400ms CQT spaces. The residue errors show only 2 significant differences between the 2ms and 4ms window lengths, but all except the maximum TFR error show a significant difference at the 99.9% level between the 400ms window and the other two lengths. For the minimum error, the shorter window lengths show a smaller magnitude of error, yet for the other measures, the longer window length shows a smaller error. This could be related to the problem with CQT resolutions mentioned above (see Table 5.4) as the longest window length will be able to accurately reproduce sounds at the highest frequency resolution whereas the other two window lengths will not.

The residue and TFR errors show different results for which window length is best for reproduction quality. Overall, there are more significant results that demonstrate that shorter windows have a better reproductive quality, but these results demonstrate a weakness in the measurement of audio quality, since none of these measures can provide a definitive measure of reproduction quality.

5.3.9 Effect of window offset

Three window offsets were compared to see which offered the best reproduction after a sound was filtered through the timbre space.

Hypothesis: Shorter window offsets will produce better quality results since they will capture more data in the sound.

Null hypothesis: There will be no difference in reproductive quality between window length.

Method

Three window offsets were tested across the timbre spaces. These offsets were equivalent to 1, 2 and 4 windows per window length. For a window length of 4ms, these offsets evaluate respectively to each window starting 4ms, 2ms and 1ms after the previous window. Since not all offsets were represented equally within the CQT data, only the STFT spaces were considered. A pairwise comparison was performed between each of the groups, using the Mann-Whitney test to determine if each window length produced significantly different errors from the others.

Measure	Offset			p-value
	1	2	4	2 vs 4
		Residue Errors		
Min ($\times10^{-3}$)	-1.0	-0.99 (0.92)	-1.06 (0.071)	0.075
Max ($\times10^{-3}$)	1.0	1.0 (0.96)	1.1 (0.092)	0.086
Mean ($\times10^{-9}$)	105	96 (0.18)	**63 (< 0.001)**	*0.011*
Stddev ($\times10^{-3}$)	0.08	*0.10 (0.0022)*	**0.15 (< 0.001)**	**< 0.001**
RMS ($\times10^{-3}$)	0.08	*0.10 (0.0022)*	**0.15 (< 0.001)**	**< 0.001**
		TFR Errors		
Min ($\times10^{-6}$)	-1.21	-1.23 (0.29)	**-1.41 < 0.001**	**< 0.001**
Max ($\times10^{-3}$)	7.6	7.5 (0.91)	7.5 (0.75)	0.84
Mean ($\times10^{-3}$)	0.11	0.11 (0.90)	0.14 (0.18)	0.24
Stddev ($\times10^{-3}$)	1.6	1.6 (0.96)	1.8 (0.48)	0.46
RMS ($\times10^{-3}$)	1.6	1.6 (0.97)	1.8 (0.47)	0.45

Table 5.10: Mann-Whitney comparison of errors between window offsets for the timbre spaces. Numbers show the median calculated for each group rounded to 2 significant figures. The numbers in brackets indicate the p-value, rounded to 2 s.f., for the comparisons between the errors of the timbres spaces with an offset of 1 window length and the spaces for the other two lengths. The last column indicates the p-value, rounded to 2 s.f., for the comparisons between the spaces with an offset of 2 and the spaces with an offset of 4. Differences that are significant at the 99.9% level are shown in **bold** and differences that are significant at the 95% level are shown in *italics.*

Results

The results for the minimum TFR error and mean residue error in Table 5.10 only show significant results for the minimum error, and the median for the error is too small to be shown by Minitab, so the results have been calculated by hand. Four of the 10 measures show a significant difference between the offset of 4 and the offset of one. Three of these show a smaller error for the offset of one (no overlap) condition and only the mean residue error shows a karger error for the no overlap condition. All 3 of the measures that show a difference at the 99.9% level between the offset of 2 and the offset of 4 condition show a larger error for the offset of 4. Only 2 measures show significance at the 95% level between the offset of 1 and offset of 2 conditions, and both measures show a lower error for the offset of 1 (no overlap) condition.

Discussion

The overall results in Table 5.10 suggest that timbre spaces with less overlap are better, but there is little difference between the offset of the full and half window lengths, with only 2 differences significant at the 95% level. For the 4 measures that are significant at the 99.9% level, the standard deviation and RMS residue error and the minimum TFR error suggest that the no offset condition and the offset of 2 condition are better than the offset of 4 condition, but the mean residue error suggests that the offset of 4 condition is better than the other conditions. The results hence suggest no clear choice for offset. For this reason, the central value of an offset of 2 is chosen, since the literature on analysis-synthesis suggests some overlap is needed to restore portions of the signal lost during the windowing process [16]. The lack of significant results for this condition suggests that the addition of the extra offset conditions to the CQT would produce little benefit. Window offsets have not been discussed previously in the context of automated timbre spaces and these results suggest less overlap produces a better timbre space. Lower overlap also reduces the amount of data required to store a path in the timbre space.

5.3.10 Chosen timbre space

With the above results taken into account, the timbre space used for the following experiments is created from an STFT and PCA analysis where the STFT windows are 2ms apart with 1ms overlap (i.e. each window starts 1ms after the last) and each window is shaped by the Gaussian function. All phase information was discarded before the PCA analysis. The resultant timbre space was formed with 8 dimensions, using error limits on the EM-PCA algorithm of 1000 iterations and $\epsilon = 0.001$, causing the algorithm to complete if either the number of iterations exceeds the maximum or the difference in error between two iterations falls below ϵ.

5.4 Extending the timbre space with new sounds

Keller and Berger [72] suggest that a timbre space is poor at modelling sounds that were not used in its creation. However, no proof of this hypothesis is presented so the timbre spaces created in the previous stage were extended with new sounds in order to validate this claim.

5.4.1 Design

Three sets of sounds were generated. The first covers all the sounds present in the construction of the timbre spaces. The second covers the upper and lower extremes of ten of the instruments used in the construction of space to test the error introduced for sounds at different pitches. The third set covers the range of the Auditory Icons, with examples from Gaver's paper and at the timbral extremes of each parameter, at the same D#4 pitch of the first set. This set tests the tolerance of the timbre space to new timbres at the same pitch. The error introduced by filtering these sounds through the timbre space is compared. In the following discussion, the sounds used to generate the timbre space are referred to as the *generating sounds*, the second set are referred to as the *pitched sounds*, and the third set are referred to as the *Gaver sounds* as they are based on his algorithm for Auditory Icons. The pitched sounds and Gaver sounds are collectively referred to as the *new sounds.*

Hypothesis

According to Keller and Berger [72], sounds not used to generate the space will have a much lower quality representation than those used to generate the space. Accordingly, the errors of the new sounds should be distinct and greater from the errors of the generating sounds.

Null hypothesis

There will be no difference between the error distribution of the two sets of sounds.

5.4.2 Method

The Mann-Whitney test used for the timbre space comparisons is used to compare the original sounds with the new sounds. To generate the data for the comparisons, 10 new pairs of sounds are generated to complement the 27 generating sounds. The 10 pairs relate to 10 of the generating sounds and are formed by taking the lowest and the highest pitch provided by the MUMS CDs for each of the 10 pairs of sounds. These new sounds are shown in Table 5.11.

Short Name	CD num	Track num	Sample name	Note
PfL	3	2	9' Hamburg Steinway, loud	A0
PfH	3	2	9' Hamburg Steinway, loud	C8
X1L	3	16	alto saxophone	C#4
X1H	3	16	alto saxophone	D5
X3L	3	17	soprano saxophone	C#5
X3H	3	17	soprano saxophone	D#6
FfL	2	2	flute, flutter	C4
FfH	2	2	flute, flutter	E6
XyL	3	5	xylophone	F4
XyH	3	5	xylophone	C8
BNL	2	14	bassoon	A#1
BNH	2	14	bassoon	F4
FLL	2	1	flute, vibrato	C4
FLH	2	1	flute, vibrato	C7
SmL	1	15	cello, martelé	C2
SmH	1	15	cello, martelé	G5
SpL	1	13	cello, pizzicato	C2
SpH	1	13	cello, pizzicato	D5
BpL	1	18	double bass, pizzicato	C1
BpH	1	18	double bass, pizzicato	F4

Table 5.11: Short names, source and full name for each sound added to the timbre space. The first two letters of the short name correspond to the names in Table 5.1, with an 'H' appended for the highest pitch of the instrument from the CD and an 'L' appended for the lowest pitch.

In addition, to demonstrate the addition of new timbres at the same pitch, 21 sounds were generated using a synthesiser based on Gaver's Auditory Icons. These sounds are listed in Table 5.12

Material	Hardness	Damping	Quality
Metal	0.005	0.001	4
Plastic	0	0.5	10
Wood	-0.001	0.1	5
LHLD	-1	0	6
HHLD	1	0	6
LHHD	-1	1	6
HHHD	1	1	6

Table 5.12: Material names and parameters for the Gaver Auditory Icon synthesiser for the new timbres added to the space. All sounds are pitched at D#5. Each of the seven materials above is recorded in a "Bar", "String" and "Sine" configuration, described in Gaver [14], giving 21 new timbres. Hardness and damping for "Metal", "Plastic" and "Wood" are as described in Gaver. Quality is a measure of the number of partials in the sound.

5.4.3 Results

Aggregate	Original	Pitched	Gaver
Residue Error			
Min	-18	-12 (0.34)	**-3900 (< 0.001)**
Max	18	13 (0.38)	**3900 (< 0.001)**
Mean ($\times 10^{-3}$)	1.2	1.1 (0.23)	0.01 (0.056)
Stddev	3.2	*1.5 (0.047)*	**92 (< 0.001)**
RMS	3.2	*1.5 (0.047)*	**92 (< 0.001)**
TFR Error			
Min	-2.6	**-0.39 (< 0.001)**	-3.2 (0.60)
Max ($\times 10^{3}$)	86	43 (0.46)	**3400000 (< 0.001)**
Mean	340	*66 (0.044)*	**280000 (< 0.001)**
Stddev ($\times 10^{3}$)	3.2	1.1 (0.12)	**47000 (< 0.001)**
RMS ($\times 10^{3}$)	3.2	1.1 (0.11)	**47000 (< 0.001)**

Table 5.13: Mann-Whitney results of the comparisons between the original sounds and the new sounds. Table shows the median values for all comparisons, rounded to 2 significant figures. The numbers in brackets are the p-values for the comparisons against the original sounds. Significant differences in errors between the groups at the 99.9% level are shown in **bold**, significant differences at the 95% level are shown in *italics.*

The results are shown in Table 5.13 and compare the error of the original sounds against the errors exhibited in the two new conditions. Four of the 10 measures show a significant difference at the 95% level between the pitched and the original sounds but only the minimum TFR error is significant at the 99.9% level. For these four measures, the pitched sounds show a lower error than the original sounds. Eight of the ten measures show a significant difference at the 99.9% level between the Gaver sounds and the original sounds and all of these measures show the Gaver sounds have a much larger error than the original sounds.

5.4.4 Discussion

The results show some support for the idea that the timbres of the new sounds may have worse reproduction than the original sounds, since there are significant differences in errors for both the residual and TFR error. For the pitched sounds, the error appears to be lower than for the set of original sounds, and this difference is significant at the 95% level for the standard deviation and RMS of the residue error and the mean of the TFR error. The difference is significant at the 99.9% level for the minimum TFR error.

The timbres can be normalised in pitch once the TFR is produced, echoing the human timbre space of Terasawa *et al.* [44], by translating the TFR so that the fundamental frequency falls in the lowest frequency bin, provided a robust tracking algorithm can be found for tracking fundamental frequency, since the lowest frequency detected by the analysis may not be the lowest frequency heard by the human ear [17]. The original pitch of the sound would need to be stored with the resultant path in the timbre space, for example, as an extra dimension.

The Gaver sounds differ in timbre from the original sounds but do not differ in pitch. The difference between these sounds and the original sounds is highly significant at the 99.9% level in all tests except the mean of the residue and the minimum TFR error. The error of the Gaver sounds has a greater magnitude than the error of the original sounds in all these cases, suggesting that the current timbre space is poor at reproducing such timbres.

The results appear to support Keller and Berger's hypothesis on novel timbres in the timbre space. The timbre space does appear to need to be populated with a good range of timbres to be able to represent them. This result suggests that the

PCA algorithm has overfitted the timbre space to the data. Without creating a corpus of all possible timbres for analysis, there are a number of other ways to improve the generalisation of the timbre space. Machine Learning algorithms other than the PCA allow the separation of data into a training and a validation set, where the training set is used to generate the transformation, and a validation set is used to ensure the transformation can be generalised to new data. The two sets should ideally be randomly chosen from a large population. Alternative analysis techniques may also improve the generalisation of the timbre space by separating harmonics from noise as in, for example, Stochastic Modelling [128] or the Timbre Model [127], or by modelling noise in the transformation itself, via the SPCA or FAC techniques (see Section 3.3.2).

5.5 Comparison of FM and additive synthesis

The two methods for generating output from the timbre spaces that have been tested are the additive synthesis and the FM synthesis techniques (see Section 2.4). All the analysis has been done in Matlab as engineering challenges have prevented implementation of the mapping techniques in the parser, which would be required to allow environmental effects to be added to the DXi output.

The raw output of both techniques is fairly noisy, due both to transient effects from the analysis and phase differences between harmonics and between sounds. In order to correct this, the sounds are smoothed. This leads to the output sounds varying from the input sounds slightly more than otherwise but at the benefit of producing sounds which are more pleasant. The full data for the comparison is discussed in the next section.

5.5.1 Smoothing

There are three points in the synthesis where smoothing can be applied to remove noise from the signal. The timbre space path can be smoothed, the parameter space path can be smoothed and the final signal can be smoothed. The latter is equivalent to applying a high-pass filter to the signal, with the cut-off frequency related to the size of the smoothing window. This has the effect of removing high frequencies and hence dramatically altering the timbre of the sound. In practice, smoothing the paths is more effective, since they can smooth over a wider time window (as each point represents several samples in the output sound) without losing resolution in the high frequencies of the sound. The only smoothing done on the output is to shape the output for each time step by the original analysis window to remove edge effects from the re-synthesis. This windowed re-synthesis is common throughout the analysis-synthesis literature, particularly when overlap-add resynthesis is used [58]. No formal comparison of smoothing techniques has been performed, since there are too many independent variables that apply to smoothing.

The general effect of the smoothing is to remove the harshest transients and minor frequency and amplitude fluctuations from the input sounds to produce a more pleasant output. Whilst this approach does diverge from the general goals of Analysis-Synthesis techniques where accuracy is sought, it fits more closely with graphical interfaces where icons can be simplified representations real-world items designed to be clear and easy to distinguish. Photo-realistic representations of items often are too detailed to be easily used as icons. In this respect, this work follows the lead of those such as the Sounding Object project [46], where caricatures of Auditory Icons are constructed by systematically removing features of the sound until only the essential portion remains, for example a rising tone instead of a filling vase. In their work, the process is known as *cartoonification.*

5.5.2 Additive synthesis

The additive synthesis approach has been used for all the experiments detailed above as the synthesis time is a small fraction of the overall analysis time required to generate the spaces and sounds required above. In the context of reconstructing a single sound, the additive synthesis model, including the time for the inverse mapping, takes up to twice as long to produce a sound as the length of the sound (i.e. a 2 second long sound takes about 4 seconds to synthesise), depending on the complexity of the sound, and the map required to construct the additive synthesis parameters takes up about 1kb per timbre space.

5.5.3 FM synthesis

The FM synthesiser has been implemented both in Matlab and as a DXi host, but due to problems with the parsing system for the timbre space, the following discussion is concerned with the Matlab implementation. The major distinction between the two synthesisers is that the DXi version accepts MIDI commands and can synthesise from a timbre space path whose points do not represent constant time steps. In the Matlab version, each point covers the same amount of time as the analysis window used to generate the space.

A function has been defined that takes multiple values for each parameter of the FM synthesiser and an boolean argument to say whether the origin should be added to the parameter range, in order to correct the multiple-origin problem of the k-d tree as described in the next section. The function generates the timbre space point representing each of the possible combinations of the given parameters by generating a 1 second sound using those parameters, generating the timbre space path for that sound then taking the mean of all the points in the path. This point in the timbre space is then tagged with the original point in parameter space to form one element of the input set to the k-d tree.

The number of points generated for the tree grows exponentially with the number of parameters and linearly with the number of values given for each parameter, so it is very inefficient at storing parameters at a high resolution. The k-d tree alternatives discussed in Section 3.5.2 are more efficient but at the cost of accuracy.

The space used to generate the sounds used to evaluate the FM synthesiser consists of 16666 points. These points are made up of the values $0, 0.1, ..., 10$ for "amplitude", $0, 1, ...10$ for "modulation index", $2, 6, 10$ for "modulation frequency multiplier" and the values $2000, 6000, ..., 18000$ for the parameter "frequency", with an added origin where all parameters are zero. These values were chosen to reflect the important range of each parameter, and reflect the fact that the timbres of the chosen sounds vary most in amplitude.

The addition of Auditory Icon timbres demonstrated that the timbre space does not handle new timbres very well, and this is reflected in the poor initial results from the FM synthesiser. Since the Auditory Icon timbres showed large errors, no formal evaluation of the FM synthesiser was performed, aside from validating that the k-d tree and FM synthesis could produce output from the space. The addition of new timbres needs to be addressed before the FM synthesis can be used as an output device from the timbre space.

k-d tree mapping

The exponential growth exhibited in the parameter map has led to memory problems at higher resolutions. This has led to the need to minimise the resolution of the space whilst still achieving reasonable reproduction quality. Smaller resolution spaces are less accurate than higher resolution ones, and this loss of accuracy can make it hard to distinguish between certain sounds.

Higher resolution spaces are slower to search than lower resolution spaces, resulting in a mapping and synthesis process that takes longer than the synthesis window to compute, preventing higher resolution spaces from being used in real-time situations. The quality of the output therefore depends on the available hardware. Machines with more storage for the map and that are faster in searching the map will allow for higher resolution spaces.

All generic mapping techniques suffer when multiple points in one space map to a single point in the other. In this case, multiple points in parameter space can map to a single point in timbre space, particularly if that point is the origin, representing silence. For the FM synthesiser, this problem is easily addressed by removing the origin from the range of inputs and computing it separately. Other synthesisers may have to deal with this problem in other ways.

5.5.4 Gaver's Auditory Icon synthesiser

The Auditory Icon synthesiser (see Section 2.3.3 and Section 3.5.2) can be modelled in the timbre space in the same way as the FM synthesiser, and uses the existing k-d tree implementation for the mapping. Unlike the FM synthesiser, the Auditory Icon synthesiser does not have a Matlab implementation so cannot be added through the Matlab interface.

The Auditory Icon synthesiser has been implemented as a DXi in the same way as the FM synthesiser. To automate the generation of the mapping requires either an implementation of the DXi host inside Matlab or an implementation of the Matlab analysis within the DXi host. The latter is an easier route since Matlab code can be compiled into C using tools from Mathworks [105]. The DXi host needs to be given the parameter ranges for the instrument and the required resolution for each parameter as well as the signal analysis parameters and dimensionality reduction map for generating the timbre space. The mapping is then constructed in a similar way to the FM synthesiser by creating a sound for each desired parameter set and associating it with a target point in the timbre space.

The DXi host has been used to generate a number of timbres to test how well the timbre space can model timbres that were not used in its construction. The results of the analysis suggest that the chosen timbre space is much worse at modelling the Auditory Icon timbres than it is at modelling the orchestral timbres used to construct it, and so any parameter map that could be built for the Auditory Icon synthesiser will correspondingly perform less well than a synthesiser that produces orchestral timbres. The construction of a parameter map for Auditory Icons is further complicated by the nature of the timbres themselves. Unlike the FM synthesiser, whose timbres are restricted for these experiments to be stationary with respect to the parameters, a set of Auditory Icon parameters produces a timbre whose frequency components change over time, generating a path rather than a point in the timbre space.

Generating new sounds using the Auditory Icon parameters requires parameterising over paths in the timbre space. The map is constructed in the reverse way to the synthesiser map so each point in the parameter space is tagged with a path in the timbre space and searches are conducted in the parameter space. Where sounds are required that fall between parameters that have been generated, interpolation between such paths is necessary. This can be done by weighting the nearest neighbours obtained from the k-d tree search. Since the paths may be complex and of differing lengths, this interpolation is non-trivial and is discussed in Section 5.5.6 below.

5.5.5 Earcons

The sound components required to generate Earcons are loudness, pitch, timbre, duration and onset time. Multiple notes are easily defined using these basic parameters. Duration and onset time are trivially handled by the synthesiser and file format by starting and ending the sound as necessary. Loudness can be controlled by adjusting the amplitude of the synthesiser and timbre is defined by the paths available in the timbre space. Adjusting the pitch of a sound requires extending the timbre space with timbres recorded at the desired pitch and then incorporating a method to adjust the timbre according to the desired pitch.

Therefore, to incorporate Earcons into the existing timbre space framework, the timbre space needs to contain a range of timbres across the frequency range of each instrument. All the other musical cues are handled simply by defining times for the sound to start and end and defining a set of timbres (or paths) for each Earcon in the timbre space and parameter space.

The addition of pitched timbres to the timbre space shows some variation between the error of the original timbres and the pitched timbres but the error was not significant over all tests, suggesting that pitch can be encoded effectively in the timbre space, but only for those instruments used to construct the space.

The frequency range can be encoded by saving every pitch across the range of the instrument, but this is inefficient. Instead, if a few key points along the timbre range are saved, the frequencies in between can be generated by interpolating the paths representing those key frequencies. The interpolation of mixed length paths is however non-trivial.

5.5.6 Path morphing

Earcons and Auditory Icons require the ability to morph not just points in the timbre space, but also the paths representing the timbre of a sound at a certain pitch as it changes over time. These paths are not necessarily the same length and will rarely be the same shape.

Morphing between two sounds that differ only in shape is a simple matter of interpolating their respective points. Where the sounds differ in length, it is necessary to append or delete points from one or both paths before the interpolation is performed. The three ways to normalise the number of points in each path are as follows:

1. **Add silent points to shorter path.** Simplest method, but generates noise in the sound whilst morphing, as noted by participants in the Urgency Experiment (see Chapter 6). Generated sounds closer to the shortest path can change dramatically depending on the length of the longer path and how much energy is present in the longer sound over the silent portion of the shorter sound.

2. **Remove points from the longer path.** Loses data from the longer path. The longer path will sound dramatically different depending on how much shorter the other path is.

3. **Create a new path containing the same number of points as one input path but retaining the shape of the other.** To make this work, the length of each point on the path must be able to change during the interpolation and the overall length of the generated path must be the same as the path that shares its shape.

The first two options are fairly straightforward but the third option is more interesting. If we denote the two paths to be interpolated as X and Y and state that

Y has more points than X, we want to define a path X' that shares its shape with X but contains the same number of points as Y. Since each path is sampled, there is no way to ensure X and X' represent exactly the same sound, so the goal is to minimise the difference between the two paths. Given enough time, this could be achieved by adjusting X' until the sound it produces is as close to X as possible, which turns the generation of X' into an error minimisation problem. This requires a lengthy synthesise-analyse-compare cycle for each new X'. To generate X' in bounded time however, some form of interpolation over the points of X is a faster alternative, although the results will be less accurate.

The simplest interpolation is to take each point of X' to be the moving average of X over some number of points. This has the advantage of simplicity at the cost of losing detail. A weighted average allows X' to more closely match the shape of X at certain times, for example, during the initial attack of the sound, and to approximate X during the steady portion of the sound, where the error would be lower. For this to work, the window would need to be tuned according to the time along the path, taking a narrower window at the start and a wider one towards the end, and each point along the path would have to store its own length as the points near the start of X' should match those of X as closely as possible, and last a similar amount of time, whilst the majority of the new points would be generated in the steady portion of the sound and so would represent a much shorter length of time.

The ability to morph between paths allows Auditory Icons and pitched timbres to be used as parameters within the timbre space, since the paths themselves can be used as the basis for interpolation, rather than just interpolating the points. This allows the timbre space to model the entire range of existing Auditory Icon and Earcon work. The timbre space can hence be considered a superset of both these paradigms, provided the timbres involved can be modelled accurately. The addition of new sounds discussed in Section 5.4 earlier in this Chapter suggests more work needs to be done on adding timbres before the path morphing can be used to model Auditory Icons and Earcons.

5.6 Conclusions

A Matlab test suite and a C++/DXi synthesiser and parser have been written to explore the timbre space. The comparison of the various timbre spaces has been a novel contribution of this work, since previous work has only justified their choice of analyses in reference to human timbre spaces and analysis-synthesis techniques rather than considering the requirement of an automated timbre space. The results suggest that STFT is the best match for PCA of the analysis techniques tested in terms of quality, compilation time and resultant timbre space size. Gaussian windows were found to be better than Kaiser and Boxcar windows and smaller window sizes that are better at capturing the transients in the sound are best suited to the wide range of instruments tested. The windows with least overlap produce the highest quality output. With all these factors taken into account, the timbre space recommended has the following parameters:

- **Signal Analysis**: Short Time Fourier Transform
- **Window Size**: 2 milliseconds
- **Window Separation**: 1 millisecond
- **Window Type**: Gaussian
- **Dimensionality Reduction**: Principal Components Analysis
- **Timbre Space Dimensions**: 8
- **Maximum EM-PCA Iterations**: 1000

- **Minimum EM-PCA Error Difference** (ϵ): 0.001

- **Number of Source Sounds**: 27

The first four of these parameters were taken from the results of the experiments presented here and the other parameters are listed for reference so that others can repeat the experiments performed.

From all the results provided here, the timbre space appears to be worth exploring as a basis for audio design. In order to demonstrate its power in this field, it is necessary to explore things which may be useful to designers that are difficult if not impossible in existing methods of audio design. The inclusion of new timbres in the space does not produce good results, but there are a number of ways to incorporate new timbres into the space which have been discussed but not explored. Despite this, the modification of sounds according to new timbres, as discussed in the next chapter, is feasible and there are a variety of transformations within the existing timbres that are still novel. The systematic comparison of spaces and the study of generating transformations and a variety of outputs from the automated timbre space are all novel contributions that increase the options available to the audio interface designer.

The variety of options available for timbre spaces shows that they provide a rich set of resources for audio designers. The exploration and evaluation of a number of these options provides a novel contribution and supports the first aim of this research. The results shown here demonstrate that the addition of a range of pitches, as required to model Earcons, is feasible. The Earcon model is hence valid within the timbre space, supporting the second aim of this research. The results can be used to compare the timbre space with other alternatives for audio design, and this will be covered in Chapter 7.

To be truly useful as a design tool, it is instructive to explore the addition of perceptual constructs into the space to see if they can define parameters as easily as Auditory Icons or Earcons can. To this end, the next chapter examines the addition of one such construct into the space using two different methods to model the parameter and invites human participants to evaluate the results.

Chapter 6. Perception of urgency in timbre space

6.1 Introduction

Flexible design requires presentation of design parameters at all listening levels, i.e. the signal level, the perceptual level and the object or everyday level. The techniques used here were designed at the signal level, which has been used extensively in prior interfaces. The previous chapter, by incorporating the Auditory Icon model, investigated how well the timbre space can support design at the object level. Design at the perceptual level requires an understanding of how the perceptual descriptions we apply to sounds relate to the timbre of a sound.

In this Chapter, we seek to discover if the timbre space can successfully model one such perceptual description, namely 'urgency'. The timbral description of 'urgency' is taken from Edworthy *et al.* [31] who used human perception experiments to elicit perceptual comparisons between artificial timbres. This perceptual parameter was chosen as it is well defined by the paper, and it shows good consistency between subjects.

As there is no previous work on perceptual parameters within automated timbre spaces, two novel ways of modelling urgency within an automated timbre space are compared and each is assessed on how reliably it fits with the human perception of urgency. Each technique is based on existing work within timbre spaces with adaptations to allow modelling of perception.

The automated timbre space chosen for the modelling uses a 2ms Kaiser 8 window with a 1ms offset for the STFT analysis. This was the best of the models tested during the pilot experiments which led to the complete analysis in the previous chapter. These pilot experiments compared only the 2ms and 4ms sizes, with offsets of full and half window sizes with the Kaiser 8 and boxcar windows. Since the two experiments were run in parallel, this was the best timbre space available at the time.

6.2 Two methods for modelling urgency

In order to test perception of urgency in a timbre space, it is necessary to define sounds which are urgent and sounds which are less urgent. As there is no current work on how to model perception within a timbre space, we propose two models, both of which are demonstrated in Figure 6.1.

Edworthy *et al.* defined a series of sounds and asked participants to rate them on perceived urgency. The ranking achieved was consistent across their subjects. The urgent sound used here is based on the most urgent sound from their set. It is a 530Hz tone with harmonics up to the 4kHz level. Edworthy *et al.* shifted every second harmonic 10% or 50% up. Their trend showed that harmonics closer to, but not exactly, the integer multiples of frequency were perceived as more urgent, and this was confirmed informally by the author to hold at least as far as 1%. This value is hence used to strengthen the perceived urgency of the chosen sound. Therefore, the first harmonic of the 530Hz tone is 1070.6Hz instead of 1060Hz. In the following discussion, and in Figure 6.1, the point in timbre space representing this sound is defined as U.

In our novel 'gravity' model, inspired by the Analysis-Synthesis practice of morphing sounds, each point along a path in the timbre space (denoted by S) is moved towards this target, U, shown as the path **SU** in the figure. In our novel 'vector' model, inspired by the parallelogram model [19], a vector between a characteristic 'non-

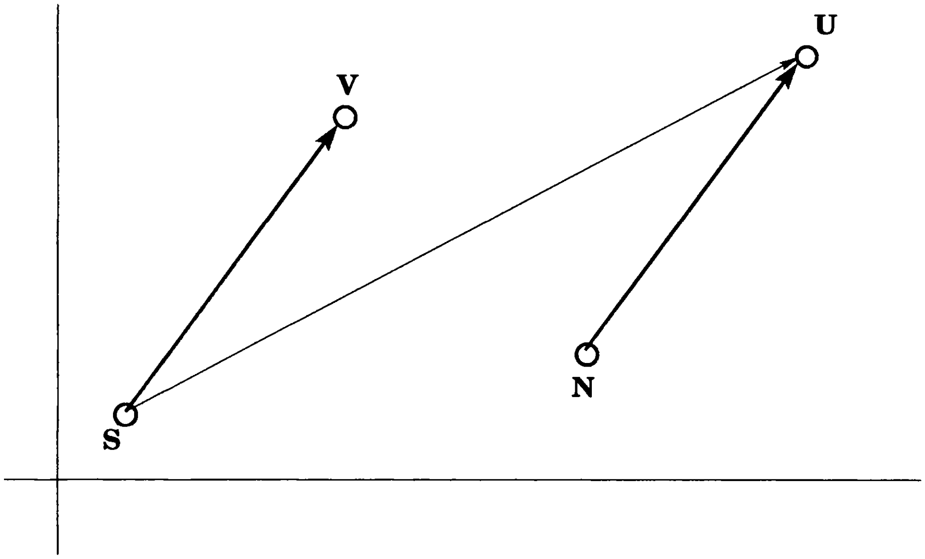


Figure 6.1: Graph demonstrating the gravity and vector models. S is a point on the original sound, U is the urgent sound. Vector **NU** defines the difference between the urgent sound and a typical non-urgent sound, and is applied to S to get vector **SV**. The gravity model morphs sound S along **SU** whilst the vector model morphs S along **SV**.

urgent' sound (N in the figure) and the characteristic 'urgent' sound is taken. The sum of this vector and the initial sound, S, provides the target, indicated by V in the figure, producing the path **SV**.

The non-urgent sound, N, is defined according to Edworthy *et al.* as a 150Hz tone with harmonics up to 4kHz where all harmonics fall on integer multiples of the fundamental. This replicates the least urgent sound from the set used in their experiment.

In both the urgent and non-urgent sounds, a point is generated in timbre space by constructing the sounds from Edworthy as specified above, lasting for 1 second each, with a 25ms linear onset and offset trapezoid envelope, to avoid transients at the start and end of the sound. For each sound, the path in the timbre space is generated using the analysis parameters for the timbre space itself, the result of which is multiplied by the output matrix from the PCA. The single point used is the mean of all the points in the path.

To generate the stimuli for the experiment, the sample "cello, martelé" (**Sm**) was taken from the set used to generate the timbre space. This was chosen as it was the shortest sound in the set at just under 1.5 seconds, and thus provided the shortest stimuli, placing fewer time demands on participants in the experiment. Each point on the path of **Sm** was translated along either the vector **SU** or **SV** such that the final stimuli contained a set of sounds representing paths with no translation, paths consisting entirely of U points or V points, and a number of paths between these extremes.

6.3 Hypotheses

The experiment attempts to test the following hypotheses, using the same data set for all the tests. The first hypothesis seeks to determine whether urgency is consistently perceived amongst the sounds both within and between participants. This hypothesis will be tested separately for each model.

The second hypothesis seeks to compare the models using these consistency ratings to determine is there is any difference in perception of urgency between the gravity and the vector model.

The third hypothesis seeks to explore a cultural phenomenon uncovered by McAdams and Cunibile [19] where musicians and non-musicians responded differently to changes in a human timbre space.

6.3.1 Consistency of rank

Hypothesis For all sounds in the set, there is a consistent ordering such that if a sound is judged to be more urgent than another, it will always be judged that way, across subjects and when the sounds are reversed. This result would support Edworthy *et al.* [31] who found an ordering of urgency amongst their sounds.

Null hypothesis There is no consistent ordering between sounds in terms of their perceived urgency.

6.3.2 Comparison of vector and gravity models

Hypothesis The vector model will produce a better correlation than the gravity model since the parallelogram model on which it is based has been shown to have perceptual significance.

Null hypothesis The vector model and the gravity model will have equal correlation within and between subjects.

6.3.3 Comparison of musicians and non-musicians

Hypothesis Participants with a musical background will perform better on the vector model than participants with a non-musical background. Sounds in the vector model are distinguished by magnitude, which McAdams *et al.* found to be a better discriminator amongst musicians than non-musicians.

Null hypothesis There will be no difference between musicians and non-musicians on the vector model.

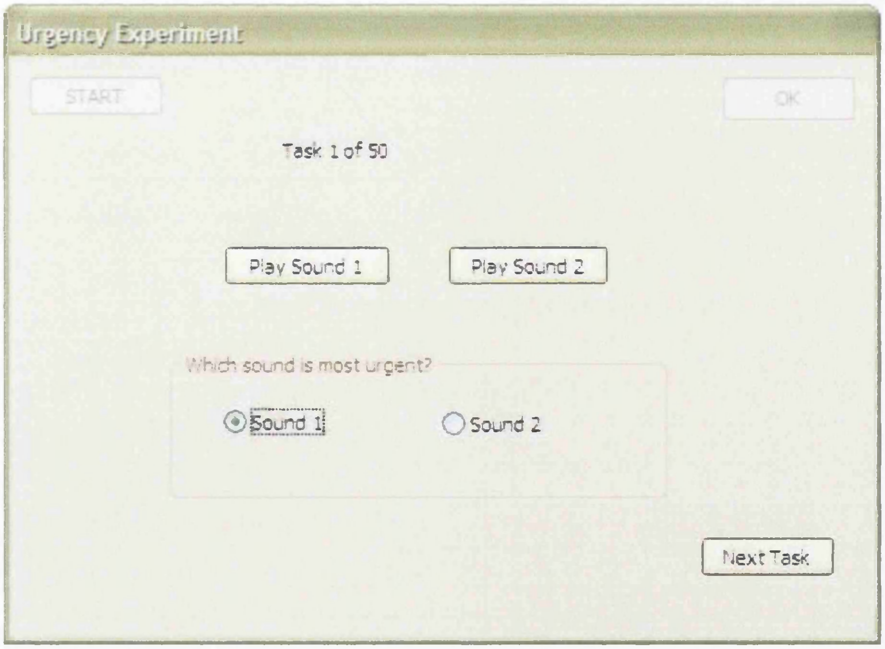
6.4 Experimental design

In this discussion, the sounds will be defined as $u1$ to uN where N is the number of sounds. The sounds will be chosen so that $u1$ is the original sound and uN is the urgent sound defined by the model. A distance rank $d(uM)$ will be taken such that $d(u1)$ is ranked 1, as the furthest from the urgent sound, and $d(uN)$ is ranked N, as the closest to the urgent sound, hence $d(n)$ increases with expected urgency.

The responses of the participants will be defined as the perception rank $p(uM)$ such that for each pair of sounds uX and uY, if the participant ranks uX as more urgent than uY, $p(uX) > p(uY)$. If commutativity of perception can be shown, the sounds can then be ordered according to this inequality so that the most urgent sound will have the highest rank.

Each participant is played every pair of sounds, in both (uX, uY) and (uY, uX) order. To provide statistically significant results, each participant will hear each pair a total of 4 times, twice in each order. A number of (uX, uX) pairs are also presented.

6.4.1 Experiment implementation



Task 1 of 50

Which sound is most urgent?

● Sound 1 ○ Sound 2

Figure 6.2: The main screen for the experiment.

The experiment is performed on a standard desktop PC. The participant uses headphones to listen to the stimuli. The experiment is divided into two trials, one for each model. The order in which the the models are presented is balanced across participants to avoid order effects. The instructions provided to participants prior to the experiment can be found in Appendix D. No participant was paid.

Once the experiment starts, the participants are presented with the screen shown in Figure 6.2 which offers the chance to listen to one sound or the other, an option box to select which sound to choose, and a 'next' button to enter their choice and move on to the next pair of stimuli. If the 'next' button is pressed before a selection is made, a dialog box informs the user of this and access to the next pair is prevented. Between trials, and at the end of the experiment, a dialogue box informs the participant to see the experimenter.

The participants are not trained on what an urgent sound is so that they are not influenced by the experimenter. For each pair they have the opportunity to hear each sound up to 4 times by clicking a button, after which the button is disabled until the next pair is presented.

At no point is any personal data held for the experiment, and all participants are given a unique, randomly assigned ID, which is linked to their results and demographic data. No personal details are recorded except age, gender and whether the participant has 5 or more years musical experience. Musical experience is recorded to compare with McAdams *et al.*'s parallelogram experiment since they found significant differences between musicians and non-musicians. Recording this data allows us to check for any differences under our conditions. As far as possible, the order was balanced among participants within a gender and a musician/non-musician group.

The participants are free to withdraw from the experiment at any time and are offered the opportunity for full debriefing and further contact after the experiment if they request it. This is explained to the participants in the consent form before the experiment starts. All participants were offered a copy of the results at the end of the experiment. For copies of the materials provided to the participants, and the experimental results, refer to Appendix D.

6.4.2 Statistical analysis

In order to determine the number of sounds needed for this experiment, the statistical tests must be considered and the results required for a significant result shall be decided. This must be balanced with the need to keep the experiment short enough for each participant to complete comfortably and quickly. If the number of sounds is small enough, it is easy to ask every participant to rank every possible pair of sounds.

Consistency of rank

For each sound uX, the hypothesis states that if $p(uX) > p(uY)$ for one trial, the participant will always rank uX higher that uY no matter in what order the two sounds are presented. To test this, a Wilcoxon analysis will be performed across all pairs uX, uY by subtracting the number of false (do not fit hypothesis, $p(uX) < p(uY)$) trials, from the number of true (do fit hypothesis, $p(uX) > p(uY)$) trials.

The categories are defined for each uX so that a positive result occurs when $p(uX) > p(uY)$ for cases where $d(uX) > d(uY)$ and when $p(uX) < p(uY)$ for cases where $d(uX) < d(uY)$ and a negative result occurs otherwise. If the participant cannot reliably determine uX and uY apart, the result will be zero.

Correlation between perception and timbre space

For each $p(uX)$ value that has a significant categorisation, the ranking $p'(uX)$ will be calculated across this subset. This will be compared against the ranking $d'(uX)$

across the same subset using Spearman's rank correlation co-efficient ρ_s.

Both $p(uX)$ and $d(uX)$ have been defined such that a strong positive rank correlation shows that perception of urgency grows as the sound moves closer to the urgent sound.

Selection of number of participants

For significance at the 99% level, for a 2-sample test, a sample size of at least 11 is required. This sample size assumes that in the 4 presentations of each pair of sounds, each participant chooses one from the pair as most urgent at least 3 times, indicating that sound is more urgent than the other. This provides a difference of at least 2 between the sounds in each pair, i.e. 3 choices of the urgent sound minus one choice of the other sound.

Selection of number of sounds, N

In order to create a tractable experimental set that will produce usable results, $N = 5$ is a good compromise. The Wilcoxon value required for 99.9% significance is reasonably low, although the ρ_s value does require a high level of proof. Smaller values of N require too large an effect to be witnessed for a significant ρ_s and larger values require too many pairs to be presented to the participant which could allow fatigue to affect the results.

For $N = 5$, there are $5 * 5 = 25$ pairs of sounds per model, taking order into account. To present each pair twice in each order requires 50 trials, or 100 sounds to be presented to the participant. If each sound is listened to the maximum 4 times, there are 400 stimuli, where each sound is 1.5s in length, giving a total time of 600s, or 10 minutes of sound for each model, or 20 minutes over the full experiment. Even after taking into account the time taken for the participant to select a sound and click next, and the time to introduce and debrief the participant, a complete experiment time of 30 minutes is reasonable.

In practice, no participant took longer than 20 minutes to complete the experiment as there were many pairs where the participants did not listen to each sound 4 times before making their decision.

6.5 Results

The following results are presented such that each trial is divided into pairs of sounds, presented in the format (1 v 2), which represent the statistics over all trials where sound 1 and sound 2 were presented together, in either order. In all the following data, sound 0 is taken to be the original sound and sound 4 is the most urgent sound.

The demographics of the participants is presented in Table 6.1. All participants were students or staff at Glasgow University and none reported any hearing problems. Participants who spoke English as a second language who asked for clarification were asked to think about an urgent situation and compare sounds depending on which sound was more appropriate for that situation. No native English speaker asked for clarification of the term 'urgency'.

The difficulty of defining participants as musicians or non-musicians was discussed in McAdams *et al.* [19], and the definition used here is similar, although the difficulty of finding participants who practice 3 times a week, as used in that paper, meant that this requirement was dropped. Consequently, a musician is defined as someone with at least 5 years musical background who described themselves a currently active player. The measurement of musical ability by the Musical Quotient developed by Challis *et al.* [149] was not considered since the evaluation time of the measure, at 45 minutes, was longer than the expected experiment time.

				Age		
	Total	Male	Female	Range	Mean	Std. Dev.
All	11	7	4	23-42	28.91	5.22
Musicians	6	3	3	-		
Non-Musicians	5	4	1	-		

Table 6.1: Demographics of experimental participants, values rounded to 2dp where required.

	Gravity Model				Vector Model			
Pair	N	Wil. Statistic	p	Median	N	Wil. Statistic	p	Median
All Participants: N = 11								
3 v 4	*11*	*59.5*	*0.021*	*4.000*	9	23.0	1.000	0.000
2 v 4	*11*	*60.0*	*0.018*	*4.000*	*11*	*60.0*	*0.018*	*4.000*
1 v 4	*11*	*60.0*	*0.018*	*4.000*	**10**	**55.0**	**0.006**	**4.000**
0 v 4	*11*	*59.5*	*0.021*	*4.000*	**11**	**65.0**	**0.005**	**4.000**
2 v 3	*11*	*60.0*	*0.018*	*4.000*	**11**	**66.0**	**0.004**	**4.000**
1 v 3	**10**	**53.5**	**0.009**	**4.000**	**11**	**66.0**	**0.004**	**4.000**
0 v 3	*11*	*59.5*	*0.021*	*4.000*	**11**	**66.0**	**0.004**	**4.000**
1 v 2	*10*	*47.5*	*0.047*	*3.000*	*9*	*43.0*	*0.018*	*2.500*
0 v 2	9	39.5	0.051	3.000	9	36.0	0.124	1.000
0 v 1	6	16.0	0.295	1.000	7	22.0	0.205	1.000
Musicians: N = 6								
3 v 4	6	17.0	0.208	3.000	5	9.0	0.787	0.000
2 v 4	6	17.5	0.173	4.000	6	17.5	0.173	4.000
1 v 4	6	17.5	0.173	4.000	5	15.0	0.059	4.000
0 v 4	6	17.0	0.208	3.000	6	20.0	0.059	4.000
2 v 3	6	17.5	0.173	4.000	*6*	*21.0*	*0.036*	*4.000*
1 v 3	5	13.5	0.138	2.000	*6*	*21.0*	*0.036*	*4.000*
0 v 3	6	17.5	0.173	4.000	*6*	*21.0*	*0.036*	*4.000*
1 v 2	5	11.0	0.418	2.000	6	19.5	0.075	3.000
0 v 2	4	7.5	0.465	2.000	5	10.0	0.590	1.000
0 v 1	3	3.0	1.000	0.000	3	4.0	0.789	0.000
Non-musicians: N = 5								
3 v 4	5	15.0	0.059	4.000	4	4.0	0.855	0.000
2 v 4	5	15.0	0.059	4.000	5	15.0	0.059	4.000
1 v 4	5	15.0	0.059	4.000	5	15.0	0.059	4.000
0 v 4	5	15.0	0.059	4.000	5	15.0	0.059	4.000
2 v 3	5	15.0	0.059	4.000	5	15.0	0.059	4.000
1 v 3	5	15.0	0.059	4.000	5	15.0	0.059	4.000
0 v 3	5	15.0	0.059	4.000	5	15.0	0.059	4.000
1 v 2	5	15.0	0.059	3.000	3	6.0	0.181	2.000
0 v 2	5	15.0	0.059	4.000	4	9.0	0.201	2.000
0 v 1	3	6.0	0.181	2.000	4	8.5	0.273	2.000

Table 6.2: Wilcoxon Results. Table shows results grouped by vector/gravity model across rows and grouped by all, musician and non-musician participants by column. All tests were two-tailed, comparing the median to zero. **Bold** entries are significant at the 99% level and *italic* entries are significant at the 95% level. A positive estimated median indicates the second of the pair was rated more urgent than the first, and has a maximum value of 4.000. For each pair, values of N less than the number of participants indicate that a number of participants rated the two sounds equally (i.e. of the 4 trials, each sound was classified as more urgent twice.)

	Gravity Model				Vector Model			
Pair	N	Wil. Statistic	p	Median	N	Wil. Statistic	p	Median
Female: N = 4								
3 v 4	4	10.0	0.100	4.000	3	4.0	0.789	1.000
2 v 4	4	10.0	0.100	4.000	4	10.0	0.100	4.000
1 v 4	4	10.0	0.100	4.000	4	10.0	0.100	4.000
0 v 4	4	10.0	0.100	4.000	4	10.0	0.100	4.000
2 v 3	4	10.0	0.100	4.000	4	10.0	0.100	4.000
1 v 3	3	6.0	0.181	2.500	4	10.0	0.100	4.000
0 v 3	4	10.0	0.100	4.000	4	10.0	0.100	4.000
1 v 2	4	6.5	0.715	2.000	3	4.5	0.593	1.000
0 v 2	2	3.0	0.371	2.000	3	0.0	0.181	-2.000
0 v 1	2	3.0	0.371	1.000	2	0.0	0.371	-1.000
Male: N = 7								
3 v 4	7	23.5	0.128	4.000	6	9.0	0.834	0.000
2 v 4	7	24.0	0.108	4.000	7	24.0	0.108	4.000
1 v 4	7	24.0	0.108	4.000	*6*	*21.0*	*0.036*	*4.000*
0 v 4	7	23.5	0.128	4.000	*7*	*27.0*	*0.035*	*4.000*
2 v 3	7	24.0	0.108	4.000	*7*	*28.0*	*0.022*	*4.000*
1 v 3	*7*	*27.0*	*0.035*	*4.000*	*7*	*28.0*	*0.022*	*4.000*
0 v 3	7	23.5	0.128	4.000	*7*	*28.0*	*0.022*	*4.000*
1 v 2	*6*	*21.0*	*0.036*	*3.000*	*6*	*21.0*	*0.036*	*4.000*
0 v 2	7	23.5	0.128	4.000	*6*	*21.0*	*0.036*	*3.000*
0 v 1	4	7.0	0.584	1.000	5	15.0	0.059	2.000

Table 6.3: Wilcoxon Results by gender. Table shows results grouped by vector/gravity model across rows and grouped by gender along the columns. All tests were two-tailed, comparing the median to zero. *Italic* entries are significant at the 95% level.

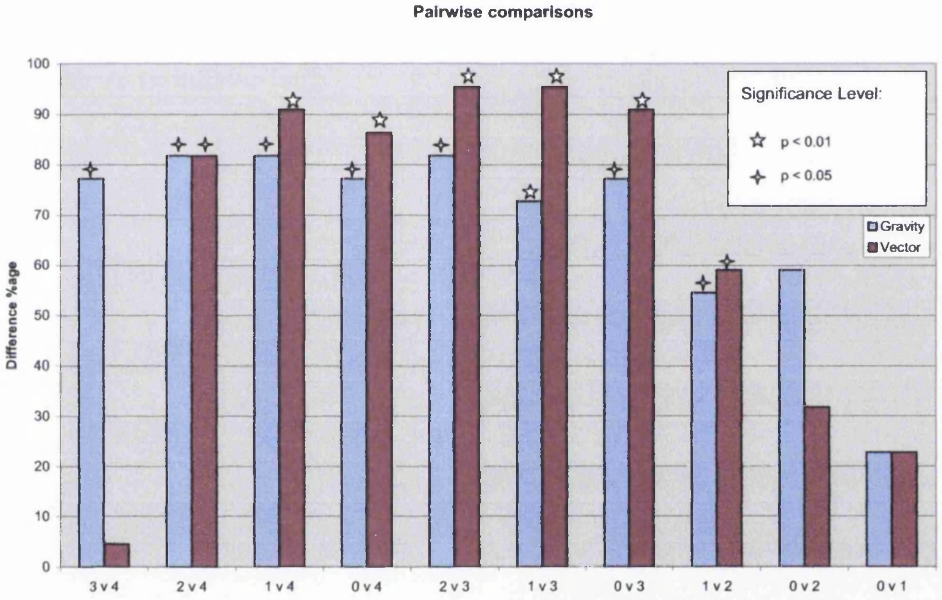


Figure 6.3: Results of the urgency perception experiment. Sound 0 was the original sound and sound 4 was the most urgent of the set. For each pair of sounds, the height of the bar represents how many times the second was classified more urgent than the first. Pairs that are significant under Wilcoxon are marked with a star.

None of the results of the Wilcoxon analysis shown in Table 6.2 are significant at the 99.9% level so no rank correlation analysis has been performed.

6.5.1 Consistency of rank

Figure 6.3 demonstrates that for every pair, the sound defined as most urgent in the timbre space is rated most urgent by the participants. In 15 out of the 20 pairs, this preference is significantly different from chance at the 95% level. The pairs that are not significant are those between the original sound and its two nearest neighbours and the result between the most and next most urgent sounds in the vector set.

This anomalous result from the vector set stems from the fact that of the 11 participants, 4 classified sound 4 as more urgent than sound 3 every time (i.e. in the same direction as the timbre space), whereas 4 classified sound 3 as more urgent than sound 4 every time. The remaining 3 participants showed a slight tendency towards sound 4 as being more urgent, hence providing a positive score overall, but the Wilcoxon classifies the mean of this pairing at zero suggesting overall participants ordered the pairing no better than chance.

As can be seen from Table 6.2, this result is represented by an estimated mean of zero across all participants but also within both the musician and non-musician groups suggesting that this effect is independent of musical training. Similar analysis, shown in Table 6.3, has also shown the effect is independent of gender.

The fact that 8 of the 11 participants classified this pairing consistently however does suggest the two sounds are distinctive, although it should be noted that one musician and one non-musician were unable to find a consistent ordering for this pair.

The gender differences in Table 6.3 show that the results for males show similarity with the overall results in terms of the number of significant pairs at the 95% level.

Although there are no significant pairs for the female participants, the trend for the male participants suggests there is little if any difference in performance between genders, however there are not enough female participants to ensure the validity of this trend.

6.5.2 Difference between Vector and Gravity model

Of the 10 pairings in each group, the vector group has 5 pairings that are significantly ordered at the 99% significance level whilst the gravity group only has one, which was not rated significant for all participants (one participant could not rate the pair according to the Wilcoxon). At the 95% significance level however, the vector group has 7 significant pairings whilst the gravity group has 8. This is due to the poor results discriminating between the two most urgent sounds in the vector case as discussed above.

In Figure 6.3, it can be seen that in 6 out of the 10 pairs, the vector group shows a more consistent ordering of urgency. I.e. a higher percentage of the pairwise trials rate the expected urgent sound more urgently than the other sound in the pair. These results are strengthened by the fact that 5 of these pairings are significant at the 99% level.

In 2 out of the 10 pairs, the least urgent pair (0 v 1) and the most urgent pair (3 v 4), the gravity group shows a more consistent ordering of urgency. In the remaining 2 pairs, no difference between the groups is observed.

Taking the results for musicians and non-musicians separately, there are fewer significant pairs within each group, as is to be expected, but the musician group has 3 significant pairings (at the 0.05 level) in the vector sounds whereas the non-musician group shows no significant pairings. This relates to the parallelogram trials [19] where musicians were found to perform better than non-musicians. No significant pairs were found in either group for the gravity model.

Looking in more detail it can be seen that the non-musicians group is more consistent than the musician group across both models and across more pairs, achieving the maximum median of 4.0 across 14 of the 20 trials compared with 10 out of 20 trials for the musician group although none of the non-musician results are significant. The closeness of the results across the models means that the significant results from the musician group allow for the significant pairings found in the overall results. The inability to discriminate between the two models on the non-musician data suggests a much larger study is required to determine the best model for non-musicians, although a model based on direction rather than magnitude may perform better, based on McAdams *et al.*'s results [19].

The musicians and non-musicians appear to perform similarly on the vector model, but the musicians have a much worse performance than the non-musicians on the gravity model with no results close to significance. For the non-musician group, the gravity model has more pairs with a median of 4.0 than the vector model, and also more pairs with a median of 4.0 than the musicians with the gravity model.

6.6 Discussion

Overall, it appears that both the gravity and the vector models can be used to modify sounds perceptually. For most cases, the vector model seems to provide a more consistent ordering for urgency, although this breaks down for the most urgent and least urgent pairs. Discussions with the participants indicated that certain features of the sound become more important at these extremes, especially in the gravity case, and it may be that the lack of cues in the vector case prevents a consistent ordering.

In the debriefing session after the experiment, participants were asked how they felt the two models compared. For the gravity model, participants talked about the 'hum' or sinusoid as being the essence of urgency in the sound, the hum being the pure urgent sound used as the basis of the model. As there was no similar hook to listen out for in the vector case, it could be that at the extremes, the hook in the gravity model overpowers the other qualities of urgency present in the sound and is very noticeable whereas there is no similar quality to be found in the vector model. Comments on the gravity model suggested that hearing two sounds at once sounded more urgent, that the gravity model had more urgent sounds amongst the 5 used, and that the sounds in the gravity model were "fuzzier" and hence easier to distinguish from each other.

During the debriefing, all the users who expressed a preference, including some of the musicians, found the gravity model easier to use, and described the 'hum' or 'tone' as the hook they used to help them when using that model. This would suggest that users might select the gravity model in a free choice despite performing better on the vector model.

The difference in performance between the two groups, although not significant, suggests that the two groups prefer different models with the musician group preferring the vector model and the non-musician group preferring the gravity model. This supports McAdams *et al.*'s data showing a difference in performance between musicians and non-musicians with respect to detecting the relative magnitude and direction of vectors.

Whilst the results here show some support for the hypotheses, it should be noted that the results may not generalise to different conditions. Where a audio perceptual quality is not as well defined as 'urgency', such as 'age', there may be a greater variance in perception between participants than demonstrated here. The variance between participants will also be increased where a perceptual quality has context sensitive associations, such as the quality of 'tallness' possibly being associated with fear when connected with falling, yet also easily associated with achievement when connected with rising.

In reference to Edworthy *et al.* [30], it should be noted that there is no guarantee that perceptual parameters are orthogonal and adjusting one can have a confounding affect on another desired property. The timbre space could make such effects clearer, provided the interface can model them, but also offers a wide variety of possibilities for modelling such interactions, none of which currently seem to indicate a preference for either of the two models tested.

For example, the gravity model offers a target at the mean of the two perceptual properties, or a two stage process whereby the sound is moved towards one parameter then this result is moved toward the other parameter, with the order being significant. The vector model offers the mean of the two target vectors, their addition, or even their cross-product. As the cross-product of two vectors will be perpendicular to both of them, this cross-product will define an orthogonal dimension along which the sound can be changed. This vector may or may not be perceptually significant but it is possible that changes along this vector will not affect the perceived urgency of the sound. This model is novel and untested so further experiments should be done with a much larger set of perceptual parameters in order to explore the perceptual effect of this model.

The vector model has another advantage over the gravity model not tested by the experiment. In the gravity model, all sounds that have been made urgent will sit in the same region of space, which could lead to diminished ability to discriminate between sounds that have been made urgent. In the vector model, the urgent sounds will be as well spaced as the original sounds and will have an identical path to the original sound, although it will be transposed. This should lead not only to better discrimination between urgent sounds but also to greater identification of an urgent sound to its original non-urgent sound.

All of these suggestions should however be taken in the context that although performance on the vector model is similar between musicians and non-musicians, tools aiming to capitalise on this model by seeking to provide a musician-friendly interface may not produce the optimal interface for non-musicians, who appear to prefer the gravity model.

Given so many possibilities, it appears that the superior choice of model depends not on the modelling of a single perceptual parameter but in which model best presents and mitigates the problems of interactions between such parameters. The scale of investigation required to examine these interactions is beyond the capabilities of the current study.

Looking at the problem in another way, it could be argued that both models are valid given a certain cultural background. This experiment indicates a possible cultural difference between musicians and non-musicians in terms of their preferred model. With this in mind, it may be necessary to adapt an audio interface to match these cultural biases, perhaps by offering a vector based model to musicians and a gravity based model to non-musicians. This model could be chosen by the user in much the same way as they currently choice a graphical theme. The designer, instead of suggesting a particular sound for a situation, would merely ask for a number of sounds between less urgent and more urgent and the system would generate the appropriate sound according to the chosen model.

The choice of urgency as a model adds a further complexity to the design and illustrates a more general design problem that a system like this cannot solve. Shannon's information theory [150] states that the most interesting signals are those that occur least frequently. Applying this concept to interface design, we should expect that if most sounds have a high urgency, it will be sounds with a lower urgency that stand out more as they are less familiar. Designers should therefore take learning and psychological effects into account on top of whatever perceptual models exist in the design system.

The complexity of adding such cultural sensitivity is increased by the knowledge that each model will have its own pitfalls, such as the gravity model losing distinction between sounds as they close in on the target. To avoid such complex, culturally sensitive design issues, the number of cultural models should be kept to a minimum, sacrificing the needs of the user to make the design tractable for the interface designer.

6.7 Conclusions

The gravity and the vector models both demonstrate their effectiveness in modelling urgency. In both models, over half the trials showed a significant preference among the participants to rate sounds in line with the model. Sounds defined as more urgent by the model were rated as more urgent by the participants. The two models are both novel additions to automated timbre spaces and the addition of perceptual information to each transformation is also novel.

The overall trend suggests the vector model is better than the gravity model as it has far more pairings that are rated significant at the 99% level. This trend breaks down at the extremes, however, so it appears that if a high density of sounds is required in the vicinity of the original or the urgent sound, the gravity model should be used.

The two models each provide a new way of looking at audio interface design, since the designer can develop sounds based on the perceived urgency directly within the timbre space. This is in contrast to existing design systems that work at the signal and musical levels, and is a step towards a generalised ecological sound design environment. The difference in performance between the two models suggests that the audio interface may need to be tailored for each user, by creating an audio theme based on the users abilities. This would need a much larger study to be explored further.

The timbre space has been shown to capably model urgency amongst the participants. To complete the generalised audio interface system, a number of other perceptual descriptions must be modelled, using the vector, gravity, or some other model. Useful descriptions for a desktop based interface may include size, which would map directly to file size, or age, which could be defined by asking users to rate sounds based on how old they sound and generating a perceptual parameter from that.

In conclusion, the vector model seems to provide a better overall basis for sound design than the gravity model, and should be explored further through perceptual testing over a larger number of participants. In light of the possible differences between musicians and non-musicians in terms of model preferences however, it may be wise for others to investigate this discrepancy further using more thorough tests, with more participants, to determine if musicians and musical tools would help or hinder audio design for a non-musical user base.

Chapter 7. Discussion and Conclusions

7.1 Introduction

This chapter looks at the thesis as a whole and evaluates what has been done and outlines possible avenues for future research. Whilst the timbre space approach offers many promising directions for audio design, there are a few problems that still need to be addressed. The theoretical foundations for possible solutions are outlined here so that others can implement those solutions.

Section 7.2 reviews the aims of this study and highlights the novel contributions made in researching those aims. The relevant results are reviewed to provide recommendations to others who wish to extend this work.

Section 7.3 assesses the methodology and technical choices made during the research and considers what improvements can be made considering the results obtained and the new research and technology made available since those choices were made.

Section 7.4 discusses the work needed to extend this proof of concept system into an audio design tool and other directions for future research. A number of avenues for further research are discussed, with some ideas presented for initial work. Section 7.5 summarises the thesis aims and contributions.

7.2 Contributions of the thesis

This research has made a number of novel contributions to the field. These are detailed below followed by a summary of how these contributions fit with the original aims of this research. These contributions extend the existing work by applying techniques from another field to audio interface design, or by finding solutions to problems unique to audio interface design. These two areas are detailed within the relevant contributions.

7.2.1 Choice of timbre space

This section reviews the choice of timbre space in light of the literature review and looks at new additions that could be made with reference to the results found in the Chapter 5 and Chapter 6. The addition of new data and new analysis techniques discussed here may lead to problems in the size or the operational speed of the timbre space, so require the resolution of the problems discussed in Section 7.2.2.

A methodology for comparing timbre spaces was developed that did not require lengthy subjective perceptual tests. This was required due to the large size of the data sets. The objective comparisons were designed to cover the range of timbres produced without prejudice, unlike some tests which focus on timbre in the speech range. The suitability of these tests is discussed in the next section.

The comparison of spaces is a novel contribution of this work, which has led to a number of criteria on which timbre spaces can be compared. The chosen timbre space is believed to be the best from those tested for reconstructing timbres appropriate for Earcons, since the original data set contains many orchestral sounds, as used in many Earcon designs.

Summary of results

A number of timbre spaces were compared and one was chosen for use in further experiments. The chosen space was identified as producing reasonable quality output and offered a good balance between quality and size.

The aim of the timbre space comparison was to determine a good set of analysis techniques to use to generate a timbre space that would be useful for audio interface design. The speed of the analysis was a problem and prevented analysis of more techniques and parameters. Whilst more modern hardware, such as faster CPUs, would improve this speed a little, the computation speed of the timbre spaces used could be an important factor in the near future, and will influence the ability to further compare timbre space analysis techniques.

The analysis techniques used were unable to provide a perfect reconstruction of the original signals. Other techniques, such as Wavelets and Auto-Regression, may improve this, but are harder to implement alongside the multi-dimensional scaling used. This problem is discussed further in the next section.

Despite these complications, the timbre space seems to be a good platform for exploring audio, since it provides a geometric way to look at sounds, allowing new transformations to be explored. It can model synthesisers using a parameter map, although the current implementation requires a lot of storage. This concept can easily be extended to generate a parameter map for virtual synthesisers, to explore parameterisations that are important for a specific application before generating an optimised synthesiser to handle them. This is discussed further in Section 7.4.

The timbre space, in common with some analysis-synthesis techniques, allows a wide variety of sound manipulations such as pitch shifting without changing the length of a sound, or extending and reducing the steady portion of a sound easily without noticeable artifacts in the resultant sound, as demonstrated by Hourdin *et al.* [63].

Signal and dimensional analysis

Chapter 5 looked at a range of timbre spaces and the optimal one from the set tested was the STFT / PCA based analysis with 2ms Gaussian windows and overlap such that each window started 1ms after the last. The Fractional Fourier Transform was no better than the STFT for greater computational cost, the auto-regression analysis was incompatible with the PCA technique and the CQT / PCA analysis produced lower quality sounds than the STFT / PCA analysis.

Chapter 3 argued that the PCA algorithm is not suited to analyse the TFRs resulting from a wide range of other signal analysis techniques such as wavelets, correlograms and cochleagrams, so timbre spaces based on these techniques were not tested. Wavelets may be useful as they can easily capture both the onset and the frequency content of transient portions of a sound more accurately than the Fourier and so may be useful for accurately analysing the full range of instruments. The correlogram and cochleagram attempt to model different aspects of the human ear and so may create timbre spaces where perceptual information is more easily modelled. None of the PCA family of analysis techniques, such as FAC or SPCA is suitable for such analysis, but extensions such as kPCA [118] or alternatives such as the Self-Organising map [119] may be able to model the TFRs concerned, or a pre-processing step to convert the TFR into a suitable matrix may provide a suitable basis for building a timbre space.

The more advanced processing allowed by these techniques increases the variety of available timbre spaces. Use of alternative analysis techniques allows the timbre space to model a wider range of timbres accurately, since the time-frequency resolution trade-off is different from the standard Fourier-based techniques. Such techniques may allow timbre spaces to model transient sounds such as Gaver's Auditory Icons more accurately.

The advanced techniques listed above could also be extended using methods borrowed from the Timbre Engine [127] or Terasawa *et al.* [44]. These both constructed a timbre space with pre-defined axes, designed to capture the important perceptual parameters defined for a particular interface. Terasawa *et al.*, for example, chose parameters suited for timbres found in speech interfaces. Careful choice of these parameters prevents the over-fitting problem mentioned by Keller and Berger [72] that prevents the timbre space generalising to new sounds, which is discussed further in Section 7.2.2.

7.2.2 Including synthesis models and new timbres

For timbre spaces to be useful for audio design, it is necessary to produce output from the space. In this work, a number of synthesisers and mapping methods have been studied. Their relative effectiveness and usefulness is described here. In addition, to study the effectiveness of the design tool to work with novel sounds, a number of new timbres were introduced to test how effectively these new timbres, and hence the parameter maps, are modelled within the existing timbre space.

The novel contributions discussed below are the addition of parameter maps to a timbre space to allow a variety of synthesisers to be used as inputs and outputs to the timbre space. The addition of new timbres from Auditory Icons to the automated timbre spaces is also novel. The addition of phase to the timbre space output also presents a novel contribution to the work on automated timbre spaces.

In Section 5.5, the quality of the output was discussed, including ways of smoothing the sound to produce good quality output whilst keeping the output sound close to the input sound. The inclusion of the FM synthesiser with the k-d tree mapping was also reviewed and extensions to this are detailed below.

The addition of Auditory Icons investigates the ability of the timbre space to model other synthesisers, not just as extra output devices, but also to provide another set of inputs into the space alongside the perceptual maps discussed in Chapter 6. Auditory Icons are an example of a range of physical modelling synthesisers that can be used to create realistic sounds within the computer, giving a wide range of expression without requiring a large number of wave files. Adding these to the timbre space not only allows it to generate realistic sounds efficiently, but also allows those sounds to be modified by the perceptual map. As the results below show, the addition of physical modelling synthesisers cannot be done within the existing spaces.

Keller and Berger's hypothesis [72], which states the timbre space is much worse at modelling timbres other those used in its construction, was supported by the results showing the addition of Auditory Icons to the timbre space produces poor results. This means that the timbre space must be constructed with all the possible required timbres, or the timbre space must be constructed in a different way to allow it to generalise more easily to new timbres.

Addition of new timbres

New timbres were added to the space to test Keller and Berger's hypothesis [72] that the timbre space does not generalise to new timbres. This has implications for how well a synthesiser can be mapped into the space, since if a timbre is not modelled well by the space, the synthesiser parameters used to generate that timbre will similarly be poorly modelled in any mapping between the timbre space and the parameter space.

Additional timbres were added from the MUMS CDs [131] and from a synthesiser based on Gaver's Auditory Icons [14]. The results presented in Chapter 5 support Keller and Berger's hypothesis as the errors for the Auditory Icon timbres were significantly different to the errors for the MUMS sounds at the 99.9% level. The errors for the Auditory Icons were greater than the errors for the original MUMS sounds. Results for the MUMS sounds that differed in pitch from those used to construct the space showed some difference in errors, but these were not significant at the 99.9% level. This suggests that a method for normalising pitch of input sounds, as used by Terasawa *et al.* [44], for example, may improve the ability to synthesise a range of timbres suitable for Earcon design, but the addition of new timbres from Auditory Icons and other physical models requires a greater change in the construction of the timbre spaces, as discussed in the "Signal and Dimensional Anaysis" Section above (Section 7.2.1).

7.2.3 Validity of perception

In order for the timbre space to be an effective tool for audio design, it should be able to encode perceptual information, so a designer can increase, for example, the "urgency" of a sound for actions that have more significant effects (such as deleting more important files). The models are both novel within automated timbre spaces, as is the addition of perceptual information to the automated timbre space. In this section the urgency experiment presented in Chapter 6 is reviewed and the timbre space is evaluated in light of similar perceptual research.

The concept of urgency was used to define a perceptual dimension inside the timbre space, using a definition from Edworthy *et al.* [31]. Two novel models were compared for their ability to define this concept and both showed promise as a method for defining perceptual dimensions, although there are signs that there is a cultural bias between the two models in terms of musical background of listeners, indicating there may not be one way to define an audio interface for all users. The implications of these results for existing and future work is discussed below.

Perception of urgency

Twenty pairs of sounds were presented to the participants, ten each for the gravity and the vector models. Out of the 20 pairs, 15 showed a significant difference at the 95% level under a Wilcoxon analysis, indicating that discrimination within the pairs was good. In addition, every pair rated showed that participants agreed with the rating of urgency defined by the model. This suggests that either model can be used to apply perceptual parameters to a timbre space.

There are limits to perception in the models, seen in the results as the original sound had no significant difference from either of the next two least urgent sounds, as shown in Figure 6.3. There was also an unexpected result for the vector model where the most urgent sound had no significant difference from the next most urgent. Since the gravity model does not exhibit the same behaviour, it is unclear whether the discrepancy is due to problems with perception in the space, problems with the model or another, currently unexplained, phenomenon.

The vector model, despite the anomaly, has shown that it can be a useful model for perception, although it may not be the best model for all users. The vector model is important as it can only be implemented using a timbre space, whereas the gravity model can be implemented in any existing analysis-synthesis technique, although with greater complexity than the geometric representation of the timbre space. The vector model provides geometric exploration of combinations of sounds, and could therefore provide some insight into the interactions between perceptual parameters uncovered by Edworthy *et al.* [30]. In this way, the automated timbre space has enabled the development of a novel tool for exploring perception. This thesis has demonstrated the effectiveness of these two models in the timbre space, and considered some of the issues multiple perceptual models can address, such as cultural sensitivity.

Extension of existing perceptual work

In the course of developing the audio design models, a number of techniques from the literature were applied. In the following discussion, the useful additions to those techniques are highlighted.

Perceptual parameters Perceptual parameters, such as those defined by experiments like Edworthy *et al.* [31], have been shown to be representable within a timbre space in such a way that a given sound can be modified in relation to that parameter. The two novel models used show no clear listener preference, although there is a suggestion that musicians and non-musicians prefer different models, which has implications for culturally sensitive design.

There may be other models more suited to other perceptual parameters, such as those only representable as paths or those that are very context sensitive, as discussed in Section 6.6. The problems of combining perceptual parameters, as discussed in Edworthy *et al.* [30] were not resolved here, and further psychology research will need to be undertaken to determine the best model for resolving such issues.

Human timbre spaces The parallelogram model of changes in timbre developed for human timbre space has been validated in the automated timbre space. The application of this model to automated timbre spaces is a novel contribution of this research. The parallelogram model demonstrates that timbre can be considered as a geometric entity with differences in timbre having a noticeable magnitude and direction that can be mapped in a human timbre space. The experiment performed in Chapter 6 extends that idea to show that magnitude is also relevant to timbre differences in an automated timbre space. The use of 'urgency' to construct the vector adds a name to perceptual changes in the timbre space which extends the parallelogram model by demonstrating that the timbral analogies can be named by perceptual concepts.

In agreement with McAdams *et al.* [19], there is a difference in performance between musicians and non-musicians using the model. In addition, the experiment presented in Chapter 6 suggests that cultural background may influence the best way of encoding perceptual parameters for an individual user. The gravity and vector models are both focussed on magnitude of change in the timbre space, but it is possible that direction, path shape or other geometric constructs may prove useful for encoding perceptual parameters.

Further studies are needed to determine whether there is a single best way of encoding perceptual parameters within the timbre space, but the results here and from McAdams *et al.* suggest that audio perceptual parameters, like graphical icons, need to be understood within a cultural context, and the perceptual encoding may need to adapt to this context, for example, allowing musicians to choose a vector-based interface whilst non-musicians can choose a gravity-based interface.

Automated timbre spaces To extend the existing automated timbre space work, a range of timbre spaces were evaluated to determine which techniques were most suitable for audio design. As there is a wide range of techniques available, only a small proportion could be tested. As demonstrated in Chapter 6, a good technique has been found that provides reasonable output quality and supports at least two encodings of the perceptual map.

The automated timbre spaces axes have already been shown to be compatible with the human timbre space axes by Hourdin *et al.* [12], but the urgency experiment demonstrates that ideas developed in human timbre spaces, such as the parallelogram model, can be applied directly to automated timbre spaces.

Hourdin *et al.* [63] have shown that automated timbre spaces can be used for re-synthesis, and they demonstrated simple manipulations in the space have a noticeable effect on the re-synthesised timbre. This work adds a number of new manipulations through the development of a mathematical model of the sounds within the parser, through the addition of object models via the Auditory Icon synthesiser, and also through the demonstration of two models of perceptual information.

7.2.4 Advantages and disadvantages over existing work

Section 2.3 dealt with the existing audio interface design methodologies. In Section 3.6, the timbre space was compared with a variety of other audio design methods. In this section, the results from the experiments on timbre spaces are compared to the alternatives discussed in those two sections to evaluate whether the timbre space offers sufficient advantages over other methods to be worth considering for future research.

Firstly, the timbre space is shown to incorporate, or be able to incorporate, the ideas from Auditory Icons and from Earcons. It is then compared with the alternative audio design techniques discussed in Section 3.6 to see how well it addresses the concerns raised there.

Comparison with Granular Synthesis

In Section 3.6, Keller and Berger's [72] assertions about timbre spaces were analysed. They state:

> "(1) timbre space dimensions should be orthogonal, i.e. changes on one dimension should not affect the other dimensions; (2) the distance between samples A and B has to be equivalent to the distance between B and A; (3) the topography of the timbre space must be independent

from the number and type of classes it contains." [72]

The first two issues were addressed in Section 3.6. The third relates to the ability of the timbre space to incorporate new sounds and concepts and to express the full timbral range of an instrument.

In the urgency experiment, new sounds were added to the space based on definitions developed by Edworthy *et al.* [31]. In both the gravity and the vector models, their urgent sound was judged to be more urgent in all but one of the pairings, indicating that the timbre space captured the timbre of urgency from that sound accurately despite not being created with that sound in its analysis set. In addition, for the vector model, the difference between two sounds that were not in the original MUMS sound set were used a basis for modifying existing sounds. Since five out of the ten pairings in the vector model showed significant differences at the 99% level, it can be argued that the timbre space provides a good model of the timbres for both the urgent and non-urgent sounds used, and can also incorporate good models of sound generated between sounds used to create the timbre space and sounds added to the timbre space after the analysis. The successful addition of the urgent and non-urgent timbres is novel, as is the demonstration of changes according to the two models.

The timbral range of an instrument has been measured by taking the highest and lowest pitch available from the MUMS CDs [131], and adding those to the timbre space. A few of the measurements showed significant differences between errors in the pitched and original sounds at the 95% level, but there was no overall trend for a difference in errors between the pitched sounds and the original sounds, suggesting that there is no significant difference between errors in modelling pitched sounds and errors in modelling the original sounds. This is a novel result, since previous automated timbre spaces have normalised pitch and not explored the pitch range.

The coverage of the timbre space was tested by adding timbres from a new source, the Auditory Icon synthesiser. The errors in the reproduction of these sounds was significantly different from the reproduction of the original sounds at the 99.9% level across all but two of the measures used. The timbre space is therefore poor at modelling the timbres produced by the Auditory Icon synthesiser.

Keller and Berger's hypothesis is supported by the results in Chapter 5 since the Auditory Icon timbres are not modelled as successfully as the orchestral sounds in the timbre space. Therefore, the simplest way to accurately model Auditory Icons is to add Auditory Icon timbres to the construction set of the space. Other physical models would have to be added the same way, which would add a large amount of processing time and storage requirements for each space. The complexity introduced by such methods suggests that extending the timbre space by methods discussed in Section 7.2.1 would provide results that generalise to more timbres without greatly increasing the time and space requirements for the timbre space.

7.2.5 Comparing against Auditory Icons and Earcons

The results from the experiments presented in Chapter 5 to add new timbres to the space suggest that sounds separated mainly by pitch from those used to construct the space are modelled much more successfully than sounds separated mainly by timbre from the construction sounds. This suggests that Earcons, which mainly use the orchestral sounds used for constructing the timbre spaces used in this research, can be successfully modelled within the timbre space, provided the pitch range of each instrument is stored in the timbre space. Auditory Icons are not successfully modelled by the current timbre space, demonstrating that it does not generalise well to new timbres. The changes suggested in Section 7.2.1 should be considered so that future automated timbre space are able to handle Auditory Icon timbres alongside orchestral timbres.

7.2.6 Evaluation of the thesis aims

In the introduction, the aims stated for this research were as follows:

1. To compare a variety of timbre spaces on a range of measures important for audio design.
2. To implement Earcons and Auditory Icons as models within the timbre space, in order to draw on previous knowledge of audio design.
3. To compare two ways of modelling perceptual knowledge within a timbre space, providing the system with new design possibilities.
4. To discover the advantages and disadvantages of the timbre spaces for audio design compared against existing audio design systems within the computer music and HCI communities.

This thesis has demonstrated that timbre spaces are a viable tool for audio design, with a number of novel audio transformations available in the timbre space, some of which have been shown to be suitable for audio design. In particular, the two novel models used for the perceptual map demonstrate that perceptual information for "urgency", which is useful for the design of alarms [31] in audio interfaces, can be encoded into the timbre space.

Timbre spaces provide a platform for analysing and manipulating sounds. There are a variety of different spaces that can be used for audio design, and each offers a different balance between the size of the audio representation, the accuracy of the time and the frequency resolutions and the ability of the space to generalise to new timbres. The chosen timbre space provides one balance, but other applications may require different spaces. The comparison of timbre spaces is a novel contribution, as is the investigation of using the timbre space for audio design.

The addition of Earcons and Auditory Icons was investigated. Whilst a methodology for including these was proposed, the inclusion of these models is not complete. The addition of Earcons requires the adoption of a path morphing technique to model pitch, but duration, timbre, volume and start time can all be modelled within the timbre space framework. The addition of Auditory Icons cannot be completed with the current timbre spaces due to large differences between the timbres of the Auditory Icons and the sounds used to construct the spaces. This problem can be alleviated by adding Auditory Icons to the set of construction sounds, but a more general solution, such as those discussed above, is required to allow the full range of physical models to be incorporated efficiently into a timbre space. The addition of Earcons and Auditory Icons into the same design framework has not been previously undertaken, and the incorporation of Earcon parameters into the timbre space framework is novel, as is the investigation of adding Auditory Icon timbres to the space.

Two models for perceptual parameters were investigated to determine their validity for audio design. The gravity model and the vector model were compared, and the results suggest a difference between musicians and non-musicians, supporting McAdams *et al.* [19], with important implications for audio designers. The incorporation of a perceptual map within an automated timbre space is a novel contribution, as are the two models used to encode the parameters within the automated timbre space.

The attempt to add Auditory Icons into the timbre space shows some support for Keller and Berger's hypothesis that Granular Synthesis presents a better audio design model than timbre spaces [72]. However, timbre spaces have the ability to encode perceptual parameters, either through the novel models shown here, or encoded in their construction, as in Terasawa *et al.* [44]. Timbre spaces also allow geometric manipulation of parameters in a multi-dimensional space, which provides an advantage over the complex parameter space of granular synthesis, which Keller and Berger [72] and Roads [58] note requires abstracting many parameters to create a usable design space.

The timbre space builds on the analysis-synthesis techniques to build a geometric model of timbre providing novel techniques such as the vector and gravity models. These techniques indicate that there may be areas of audio design that timbre spaces can expose more easily than other design systems. Timbre spaces should therefore be considered as an important tool for extending knowledge of audio design.

7.3 Limitations of the research

This research has attempted to answer a number of questions regarding audio design and timbre spaces. Some of these questions remain unanswered due to limitations in the research and other questions have been raised that the current research is unable to answer. This section will look at how the research could be improved.

7.3.1 Limitations of the methodology

The research attempted to compare a number of different timbre spaces for their suitability for audio design. This section looks at how effective the comparison was and how it could be improved. This section also looks at the limitations of the study into the perception of urgency.

Objective measurement of quality

The quality of the timbre spaces and the reconstruction of new timbres was evaluated by a statistical model of errors between the input and output signals. This method was chosen as a full scale trial of subjective quality measures was impossible since there were about 2500 separate output sounds from the 27 original sounds evaluated across all 96 timbre spaces generated with the STFT and CQT. Further sounds were generated to test the addition of new pitches and timbres and to test the addition of pitch to the analysis.

Measuring quality is complicated by individual differences between humans. Some of these differences were highlighted in the experiment on the perception of urgency, where individuals with a musical background perceived sound in the two models differently from individuals without a musical background. This suggests there may be no true objective measure of subjective difference between two sounds, a fact highlighted by the large variety of human timbre spaces that attempt to measure these subjective differences.

The objective measures chosen compare the error between input and output sounds for various models. Some error is to be expected since information is lost in the construction of the timbre space. The phase information of the sound is one piece of lost information. As Chapter 5 showed, two sounds which are perceptually similar can have a large signal error when their harmonics have significant phase differences, but the addition of phase information to the sounds does not improve the output quality of the timbre spaces.

Comparing the errors between sounds generated by different timbre spaces reduces the effect of loss of information on the comparisons, but a full study into the effectiveness of timbre spaces would require a more accurate objective measure of perception. Whilst the ITU has defined measures for the loss of quality between audio encoding methods, leading to the development of PEAQ [123], there are problems with the standard that have delayed implementation of a standardised audio quality measure [124]. Until such a measure exists, no objective comparison of timbre space output can be perceptually valid. Asking participants to rate similarity of a random selection of input and output sounds would however provide some basis of perceptually valid comparison, provided a reasonable coverage of the 2700 output sounds could be assessed, either through a large number of participants rating a random small subset each, or a smaller number of participants rating a large random subset of the comparisons, over a series of sittings to avoid fatigue effects.

Objective comparison of timbre spaces

The comparison of timbre spaces has focussed on the speed of construction of the timbre spaces and the resultant size of the sounds within that space. Analysis has been performed on the reproductive quality of timbre spaces, as discussed above, but there is still no clear guidance on what makes a timbre space good for audio design.

Time constraints meant that only a small number of possible spaces could be tested, and only a small range of values were considered for each independent variable. Improved computing power since the start of this thesis, and an improvement in speed and recoverability of the algorithms used, allow a much wider range of options to be tested now, which would provide a much stronger basis for comparison. Other options to improve the coverage of the tests included a distributed solution for compiling and testing the various spaces, and the investigation of a number of perceptual comparisons, including PEAQ and perceptual encoding comparisons.

The size of the timbre space is important since a number of the STFT spaces could not be analysed due their high memory requirements, and so these spaces could not be used. The reproductive quality is important too, since a number of the CQT spaces produced sounds unrecognisable as the original sounds. These spaces were constructed from CQT analyses that violated the minimum time-frequency resolution.

Apart from timbre space size and reproductive quality, a number of other measures may be important in comparing timbre spaces. The orthogonality of a timbre space may be considered, although many useful dimensions are not orthogonal, for example, pitch and loudness, as shown by the equal-loudness curves [21]. Orthogonality is useful for geometric construction as it simplifies many transformations, but it is an open question as to how useful such transformation are. Urgency was encoded within an orthogonal space, but better results may be obtained from more exotic timbre spaces.

There may well not be a single best timbre space for audio design. Terasawa *et al.* [44] have demonstrated a good timbre space for speech design. Marentakis and Jensen [64] have shown a good timbre design tool for harmonic tones. The results presented here show that different users perform better with different audio models, and there may be a corresponding relation between users and timbre spaces based on which audio features are extracted in the timbre space construction.

Any timbre space that meets the size constraints and quality constraints used to dismiss some of the STFT and CQT timbre spaces can be useful for audio design. The STFT timbre space that was used to construct the urgent sounds did not provide perfect fidelity of sound production, but still provided significant results across many pairings of sounds. Even though no best space can be determined with the measures used, timbre spaces are still useful for exploring audio design.

Perception of urgency

The results of the timbre space comparison were not available so the urgency experiment was performed using a timbre space built with a Kaiser 8 window instead of the Gaussian window recommended by the analysis. Further experiments on urgency should use the Gaussian window to improve the output quality. It is unclear what effect this might have on the differences between sounds, as used in this experiment, but it is unlikely that improving the sound quality would lead to poorer identification or ordering, since such a result would also contradict earlier work on urgency [31] and the parallelogram model [19].

The experiment detailed in Chapter 6 was performed using a single source sound modified by a single perceptual parameter, "urgency". To validate the results for the two models of urgency that were considered, a larger scale study is required that covers more instruments and more parameters. The suggestion of a difference between musicians and non-musicians implies that a larger cross-section of participants should be considered in order to fully investigate cultural and individual differences in the perception of the models in timbre space. The study presented here was unable to investigate these problems due to limitations on the number of participants and the amount of time those participants were available for the experiment. Since each participant took between 15 and 30 minutes to complete the experiment, any larger study would need to consider revising the methodology to allow more trials to be presented in the same time to prevent participants suffering from experimental fatigue.

The results show no indication of why the least urgent pairs and the vector model's most urgent pair were not distinguished whereas the other pairs could be. A further study could investigate these by generating more pairs at the extremes to determine the resolution required to distinguish a pair, and should also consider reducing the effect of the "hum" reported by participants, possibly by artificially adding a hum to all sounds, or reducing the effective amplitude of that portion of the sound.

7.3.2 Limitations of the technology

The development of the concept has led to a few realisations of short-comings with the original design. Whilst some of those could be addressed during the project, there were other problems which need to be fixed before the system could be used for audio design. This section addresses those problems and shows the scientific basis for their resolution whilst leaving their implementation as an engineering exercise.

Parsing improvements

The parser currently converts the entire input file into a set of tokens then converts the tokens from the file into a path in either timbre space or parameter space. This decision was taken to simplify the creation of the `[MonoOut]` and [=*MatrixVariable*] sections. However, after completing experiments on the parser, it appears that this solution has too high memory requirements, as it needs to store the original file, the tokens and the final data in memory at once. It also impedes real-time changes as it does not store file state (synthesiser parameters, system variables, etc.) between generations of the parameter data. Storing the state is required if a stream of data is generated from a real-time source such as the slider GUI presented in Section 4.10.

To overcome these problems, the parser should be re-written to hold an internal state indicating the section currently being read, the synthesiser parameters and any other variables. Each line would be sent to this new parser from a file or real-time interface and it would be tokenised and stored in the internal state or sent to the synthesiser as required. This process is fairly straight-forward but presents a large engineering task since the code requires refactoring and re-testing which is a time-consuming process.

Such a model would require that each line could be read, processed and stored fast enough for real-time generation and further requires that the system status is able to reside in the working memory of the system. The advantages of this model include faster response to input, since changes can be made without requiring parsing of a complete file. This model also allows streaming of audio data from external devices, such as a sound server, as in ENO [151]. This model also avoids the requirement for the entire file to be read, then tokenised, then parsed. Processing each line in turn saves memory as only the global variables and the current line needs to be stored, rather than the entire file, token list and parse tree. The overall effect is to reduce the memory requirements.

Choice of technology API

Since the AGNULA project [144] completed in 2004, specifications for an Open Source audio plugin API have been released under the name *LADSPA* (Linux Audio Developer's Simple Plug-in API) (`http://www.ladspa.org`). As the DXi specification used for this project is no longer supported by Cakewalk, LADSPA looks to be a good alternative. It addresses two of the main issues with DXi for this project.

LADSPA has its own parameter format that allows plugins to specify types and ranges for their values, allowing the parameter specification to be encoded in the plugin instead of in a parameter file. The LADSPA specification also allows hard real-time plug-ins with specific processing speed requirements. These plug-ins can provide instant feedback from the timbre space, provided the timbre space manipulations could be performed fast enough.

Unlike the DXi format, LADSPA does not yet include native support for MIDI signals so is less suited for representing Earcons. The modifications made to extend the DXi to handle Auditory Icons can be used to port a MIDI style interface onto a LADSPA synthesiser, but doing so would require the parameter queue to be more complicated. Alternatively, the DSSI (DSSI Soft Synth Interface) (`http://dssi.sourceforge.net`) extension to the LADSPA format can be used. DSSI is at version 0.9 as of 26th July 2005, but offers standard GUI controls such as knobs and sliders, a MIDI interface and the advantages of LADSPA listed above, with the addition of having dynamic ports so that a synthesiser can be reconfigured whilst in use, for example to switch between FM and Auditory Icon synthesis. Further work on the timbre space framework should be done after porting the synthesis engine to DSSI so that the framework can make use of the real-time capabilities and the more complete parameter definitions. Porting the framework to the DSSI also avoids the problem of obsolescence since the standard is openly available and supported.

Mapping to synthesiser parameters

The k-d tree, in common with other proximity detection methods, encounters problems when multiple sounds all map to the same point in timbre space. For the FM synthesiser, there are a number of settings that result in silence, which is mapped to the origin of the timbre space. The FM synthesiser can be restricted so that it only produces one silent output by careful control of the parameters, but in a general mapping from unknown inputs, this may be harder to achieve, particularly where the common timbre space point is not due to silent output. There is currently no automated way of picking up or rectifying this problem as it is unclear which set of input parameters should be used when mapping from the target point, and interpolation of the set of all parameters is not guaranteed to produce the correct result. This is hence left as an open question to be answered when a new synthesiser is added to the space.

The number of points in the parameter map grows exponentially as a new parameter is added and grows linearly as the resolution of each parameter increases. The point-cloud matching of the k-d tree, whilst simple, is an inefficient way to represent a high-resolution parameter map. For a faster response with a high-resolution map, it would be necessary to generate a function or similar map, perhaps using kPCA [117]. The downside to such an approach would be that points that are in the point cloud for the k-d tree, and are hence represented accurately, may not be as accurately represented in such a map. More complex maps also require careful consideration in their construction in order to accurately model points not in the parameter map point cloud. A loose fitting model will have errors in mapping points both within and without the parameter map point cloud. A tight fitting model more accurately models points within this cloud at the cost of increasing errors modelling the rest of the map.

Efficiency of synthesis

The FM synthesiser itself is capable of real-time synthesis via the parser, but the addition of the k-d tree has slowed the overall execution down to less than real-time, despite promising results from the Matlab implementation of the algorithm. Whilst this suggests an inefficiency in the C++ implementation, even the speed of the algorithm in Matlab is not fast enough for multiple sounds or for real-time reproduction alongside real-time manipulation of the timbre space paths. Possible methods for improving efficiency are discussed in Section 7.4.1.

7.4 Implications for future work

A number of issues have been highlighted by this work that may affect the adoption of timbre spaces for audio interface design. In addition, a number of issues relevant to designers may have solutions due to the timbre space concept. The extensions and possibilities to the timbre spaces developed in this research are discussed in this section.

7.4.1 Improvements to the system

To use the timbre space for audio design, a number of extensions must be included to improve the performance and quality of the timbre space output. The performance improvements described here depend on the available hardware, so a review of possible improvements is presented, but no decision can be made here on which improvements should be used.

Hardware optimisation

Apart from implementing a faster, but less accurate, algorithm than the k-d tree, there are a number of hardware based optimisations that can be performed. The exact optimisation depends on the available hardware and so the implementation developed here does not include them to maximise compatibility.

The timbre space manipulations are all specialisations of vector and matrix transformations. Modern 3D graphics cards are optimised especially for such transformations. Algorithms that use the GPU (Graphical Processing Unit) on such cards for general processing rather than graphical processing have become popular in the last two years, known by the term *GPGPU* (General Processing on Graphics Processing Units) [152].

GPUs could therefore be used to speed up the generation of sounds where the GPU is otherwise idle. To accommodate this approach, the timbre space transformations would need to be broken down into 2D or 3D components for processing then recombined afterwards.

GPUs have also been used for audio processing, with the company BionicFX (`http://www.bionicfx.com`) announcing software to do complex audio processing on graphics chips and Whalen [153] demonstrating tools to perform audio transforms such as filters and chorus effects by transforming sounds into textures that the optimised GPU hardware can manipulate much faster than the CPU. This could be used to add environmental effects to the sound or to do the smoothing of the paths in space, although Whalen notes that there is processing overhead converting the sounds to textures and transferring them to the graphics hardware that may mean that the overall operation is no faster than when performed on the CPU. The advantages of freeing up the CPU from some of the processing burden should be considered, particularly where the sounds or timbre space paths can be stored on the graphics hardware to avoid the conversion and transfer costs during real-time manipulations.

On mobile phones and related devices which are unlikely to possess such dedicated graphics hardware, particularly on low-end devices, the CPU would still be required to do most of the matrix processing, but such devices all have dedicated DSP hardware for encoding and decoding speech, which is idle when not on a call. Such hardware could easily be commissioned to perform the synthesis in real-time, shifting the burden away from the CPU. If the timbre space paths were sufficiently simplified, for example, by extending the size of the analysis window, or using fixed-point rather than floating-point calculations, and if the synthesiser could be made more complicated to simplify the parameter map, such as by using an additive synthesis approach, real-time synthesis on such devices could be possible.The use of the DSP means that the sounds would have to be greatly simplified if the user was on a call.

The two major chip manufacturers in the desktop market, Intel (`www.intel.com`) and AMD (`www.amd.com`), both have extensions to their 80x86 range of chips to allow faster processing for certain types of calculation. These optimisations are designed mainly for speeding up multimedia applications, and are only available on chips released since this research started so have not been incorporated into the system. One such technology, SSE2 [154], supported by both manufacturers, supports SIMD (Single Instruction, Multiple Data) codes that allow, for example, multiple points on a path to be operated on at the same time. Writing code for these architectures allows the timbre space to perform fast morphing between points in the timbre space by performing a scaling operation simultaneously on each dimension before adding the resultant vectors together. The addition of such optimisations however would require the parser to have more knowledge of the available hardware, and to decide the best optimisations given the input code. These compiler optimisations are beyond the scope of the current research.

Better output quality

The use of techniques such as Wavelets would allow the space to capture transients in the sound more accurately due to the increased time resolution at high frequencies. This information could be used to more accurately track the evolution of a sound and more accurately represent percussive sounds in the space.

Increases in frequency resolution can be achieved in similar ways, but also via techniques such as phase-vocoding, allowing each harmonic of the sound to be fine-tuned within its frequency bin. Such increases in frequency resolution however require a different dimensionality reduction technique to avoid the problems encountered whilst using the Auto-Regression analysis technique.

To improve the output quality to the synthesiser, a more advanced mapping is required than the proximity detection methods already used. For improving the quality of the FM synthesiser output, without adding a significant number of points to the parameter map, requires a targeted non-linear mapping, such as those discussed in Section 3.5.2.

7.4.2 Extending timbre spaces

This research has developed a novel system to explore the use of timbre spaces for audio design. If this system is to be useful to researchers and designers, a number of additions are required to enable them to explore the space without requiring the knowledge of how to build and extend the space. New methods for constructing timbre spaces can provide designers with a more intuitive space and a wider timbral range. This section discusses some of these extensions.

Analysis techniques

The most interesting area for future expansion is to explore further analysis techniques to highlight certain features of sounds, or to improve the output quality. The use of a combined signal analysis and dimensionality reduction step may also produce better results and allow other analysis techniques to be used. This idea has already been used for human timbre spaces [44] where the analysis is targeted to a specific set of axes by tuning the parameters used, and such an idea could be easily extended to an automated timbre space.

Alternatively, the two stages of the analysis could be combined by using the EM algorithm (see Section 3.5.2) to tune the filters and windows used in the analysis to achieve the best reconstruction of the sound, with suitable safeguards to prevent overfitting the data to ensure that new timbres are represented without prejudice. This technique would require a good estimate of the error introduced by the analysis.

Other signal analysis and dimensionality reduction techniques can be explored, such as kPCA, Wavelets and the Wigner-Ville distribution, but extra processing steps may be required to ensure the signal analysis output is suitable as input to the dimensionality reduction stages.

Data capture of perception - Extending Urgency

The addition of urgency as a perceptual dimension in the timbre space was discussed in Chapter 6, as were the complications uncovered by Edworthy *et al.* where perceptual dimensions are not orthogonal and interact in unpredictable ways [30].

In order to fully explore the possible perceptual dimensions and their interactions, it is necessary to capture those dimensions from further experiments. These experiments can be performed as Edworthy *et al.* did by generating a set of changes to a sound and identifying which of those changes influenced the perceived urgency of that sound.

If a perceptual parameter is not well defined or the set of viable changes are otherwise non-obvious, the timbre space itself offers a possible solution to capturing the perceptual data required to generate a perceptual dimension. Generating sounds across the timbre space provides a base line which can be refined using human participants until the relevant regions in the space are identified. The general procedure for generating a new perceptual dimension for some perceptual construct, p, is as follows:

1. Generate random points in space;
2. Get participants to order points as less or more p;
3. Generate new set of points around least and most p;
4. Get new participants to order the new set of points;
5. Generate description of the perceptual dimension and the transformation.

The ordering comparisons required for such an experiment can be done either pairwise, as in Chapter 6 and to develop the phon curve, as discussed in Section 2.2.2, or in groups of three or more, as in Edworthy *et al.* [31] where sounds were rated in groups of four.

The two rounds of ordering for the dimension allow the model to be verified by the second set of participants, for whom the sounds less associated with the dimension are removed, and also allows for a measure of how strong the perceptual dimension is by comparing performance between and within participants to determine if p is widely recognised across cultures and if p can be consistently ordered by individuals.

In addition to the perceptual dimensions covered by Edworthy *et al.* [30], perceptual dimensions of a more ecological nature may also be modelled. Edworthy *et al.* looked at 'heavy' and 'tall' sounds, but other dimensions from and inspired by physical models, including Auditory Icons, can be considered.

Abstracting pitch

For timbre spaces to be truly useful as an audio design tool, there are some issues that still need to be addressed. Timbres that are not used in the generation of the space are not represented well within the timbre space. New sounds whose pitch is outside the range of those used to generate the space, but with similar timbres otherwise show much smaller errors than where the timbre of the new sound is not represented in the construction of the space. Accordingly, the timbre space requires little adjustment to incorporate new sounds that differ mainly by pitch from those used to construct the space.

Two methods exist for adapting the timbre space for new pitches. The naive approach simply adds more sounds to the space representing the full range of pitches, but there are storage implications in this approach. Piano sounds, for example, will require paths for each semitone across their 96-key range, rather than the 1 path currently used. The timbre space may also be less effective as it may be modelling the variance in pitch as opposed the variance in timbre.

If the pitch is abstracted out of the timbres, as in Terasawa *et al.* [44], the timbre space will model the timbre alone. One method of doing this is to collapse all the pitches down to the lowest possible frequency bin after the TFR has been generated and before the multi-dimensional scaling analysis is performed. The paths for each pitch can be combined using the path morphing technique described in Section 5.5.6. Lowering the pitches prevents overflow of frequencies close to the Nyquist limit, which could occur if the pitch of the sounds is raised. Equalising the pitches requires the timbres to be tagged with their original pitch, perhaps by adding a new dimension to the resultant timbre space to indicate how many frequency bins the timbre should be shifted by. Adding the ability to abstract pitch would complete the incorporation of Earcons into the timbre space.

7.4.3 Extending audio design

The use of the timbre space as a platform for audio design creates new possibilities for designers. Three of these possibilities are discussed below. Firstly, the opportunities for abstracting individual differences away from the designer are discussed. The multi-dimensional nature of the space presents a challenge for manipulating the sounds, which has been addressed so far in this research by using mathematical tools for manipulation. Designers may feel uncomfortable with such an environment, so an alternative interface is discussed.

The other design possibility introduced here allows designers to use the timbre space to prototype new audio interface designs without having to construct novel synthesisers, as Gaver had to do for his Auditory Icons [14]. In this way, the design possibilities of the timbre space can be explored without requiring real-time interaction with the space, since the sounds can be exported as samples and used in most existing audio interface frameworks, such as the Windows or Linux desktop, or within a Web3D virtual environment.

Design abstraction

The urgency experiment exposed a problem common in user interface design, that of individual differences. It suggested that musicians and non-musicians prefer different encodings of urgency. This problem is an example of differences in preferences and performance that exists between users with different cultural backgrounds. These differences can arise either through technical experience in a specialised field, such as music, or through the general cultural background defined by language and experience.

The experiment fortunately offers a possible solution to this problem. With urgent sounds, the most important feature is ranking of urgency so that less urgent messages are always perceived to be less urgent than more urgent messages. Provided the higher level design respects the uniqueness and importance of the urgency levels, as noted in the Section 6.6, it should be possible to add some level of individual sensitivity to the space, provided a model suited to that individual can be determined.

This cultural model could be defined such that, for example, given an input sound and an urgency rating, out of 100 say, a model for musicians would generate the required urgent sound using the vector model and a cultural model for non-musicians would use the gravity model. The model used could be selected by the users when they first use the system, in much the same way as a colour scheme or graphical theme. Alternatively, an analysis of how well each user performs on each model could be performed to select the appropriate model.

Interactions - manipulating in 8D

Direct manipulation of paths in the timbre space is complicated by the 8-dimensional nature of the timbre space. Whilst the output of any manipulation can be presented as audio, the input and any non-audio feedback of changes is not as easily mapped to the space.

Hermann and Ritter [54] have developed a multi-dimensional haptic input device for their visualisation tool, and some of their ideas could be applied to working in the timbre space. Translation and rotation through a 3-dimensional space provides six degrees of freedom, each of which could be mapped to a dimension in space. Hermann and Ritter added to this a device which could be shaken, where the shake frequency was mapped to excitation of their audio model. Adding a shake or a squeeze dimension to an input device could provide the necessary extra input dimensions, particularly since each of these additional inputs could provide 3 degrees of freedom.

If the input devices are limited to those commonly available, there are some graphical alternatives that provide methods to manipulate the timbre space data. In many 3D modelling packages, design is aided by generating 2D projections of the front, side and top to give a better overview. This idea could be extended to provide projections of the 8-dimensional timbre space, for example, four 2D projections of the space, or three 3D projections, with two projections sharing a dimension, or interpreting time as a ninth dimension to be modelled within one of the projections.

However, the simplest modelling builds upon the prototypes developed in Section 4.10, which used linear and triangular sliders to model perceptual constructs within the space. The eight dimensions could easily be represented by eight sliders, whose position changed over time, or by eight line graphs representing change over time, as used by Marentakis and Jensen's Timbre Engine [64].

Modelling the timbre space directly in any of these methods provides a certain way to look at the timbre space, but each of these methods could also be modified to map onto perceptual dimensions such as that of urgency, provided interactions between non-orthogonal dimensions could be modelled intuitively.

Using timbre spaces as a prototyping system

Although the timbre spaces that have been investigated are too slow for real-time parameterisation and synthesis of sounds, they can be used as a testing platform to prototype audio interfaces that do allow full interaction. The timbre space opens up the possibility of prototyping or simulating specialised synthesisers through the use of the parameter map. Just as the FM synthesiser and the Auditory Icons have a parameter map that can be created in the timbre space, parameter maps specialised for certain information can also be created.

Using urgency as an example, one may wish to develop a range of sounds to signify a variety of e-mail messages, urgency being used as a sonic indicator of the priority assigned to the message by the sender, whilst other properties of the sound could be used to specify whether the sender was a work colleague, a family member or a stranger and another property could be used to specify the age of the message. This type of classification has commonly been used in Earcon design, but may be harder to develop for Auditory Icons.

By generating a parameter map for these sounds, the three parameters could be chosen to be orthogonal in order to prevent unwanted interactions between them. There is an infinite choice for the three parameters even under this condition, and experiments using the timbre space would allow designers to narrow down the parameters to those most natural for expressing the required concepts, without having to consider how the parameters might be represented within a synthesiser. Once the parameters have been decided, the timbre space provides an easy mapping either to a synthesiser or to a set of sounds that can be used as the basis of a new Auditory Icon.

Where the set of possible parameters is sufficiently small, the timbre space can output the sounds directly to be stored and then replayed as necessary by the relevant interface. This solution is compatible with most modern systems, as it simply produces standard audio files. This prototyping ability is a novel design method made available by the timbre space, and allows the timbre space to be used to design sounds for devices that are not capable of performing the required manipulations and synthesis in real-time.

7.5 Conclusions

The experiments performed in this thesis demonstrate that the timbre space is capable of being a basis for audio design. It offers a wide range of options for manipulating sounds, a small subset of which has been explored. The important novel contributions of this thesis are the comparisons of a number of spaces, the incorporation of synthesis and perceptual models in the space and the development of a framework that allows audio to be manipulated geometrically, as demonstrated by the models that have been created.

The most important addition to the timbre space has been the perceptual map, demonstrating that perceptual qualities, once defined, can be encoded in a number of ways in the timbre space such that sounds can be altered with respect to a perceptual quality. However, there is a large body of work which still needs to be completed in order to find the best model for representing the perceptual map and for dealing with the inherent conflict between perceptual parameters, e.g. between pitch and loudness or "urgency" and "high". The addition of perceptual parameters allows interface designers to think about the response they want from a sound rather than having to think about a set of synthesiser parameters.

The timbre space provides audio interface designers with a new set of tools for exploring timbre without sacrificing the knowledge gained from Earcon design, and with enough scope to include Auditory Icons and other physical models into the design space with the appropriate analysis. The timbre space can be used either as the primary focus for design, or can be used to prototype design concepts and perceptual frameworks without requiring designers to work at the signal or musical levels of description as current tools require.

The automated timbre space has been shown to exhibit many properties of the human timbre space. In particular, the parallelogram model from human timbre spaces has been extended to create the vector model in the automated timbre space, demonstrating that the perceptual significance is not lost between the two types of timbre space. This addition of perceptual information is a novel contribution to sound design, building on the transformations used for computer music with concepts more suited to interface design. A number of different automated timbre spaces have been evaluated and this has led to a number of ideas for future work to improve and extend the capabilities of timbre spaces within the flexible framework developed for this research.

Overall, the timbre space offers a useful platform for further experimentation and shows good promise in terms of developing a perceptual map, which presents a novel model for audio design. The full implications of this model remain unclear since there are cultural differences that may affect its usefulness. The timbre space shows promise as a platform for Earcon design using perceptual maps to influence the timbre, although extra research is required to make the timbre spaces suitable for Auditory Icon design.

With reference to the original goals of the project, the timbre space does present new audio interface design possibilities and does capture some existing knowledge, allowing Earcons and the perceptual property of "urgency" to be manipulated within the same system. The vector model presents one advantage over existing audio design systems, and is an example of the geometric transformations of audio that are made available by the timbre space. Therefore, this research has demonstrated that timbre spaces are worthy of further study for audio interface design and has provided a number of avenues for such study.

Bibliography

[1] Don Cross, "Reading and writing WAV files," http://www.intersrv.com/~dcross/wavio.html, Last updated 26 June 1999, Last visited: Jun 17 2002.

[2] A-V.I. Rosti, "Linear gaussian models for speech recognition," http://svr-www.eng.cam.ac.uk/~avir2/, Last Visited 18 June 2003.

[3] Judy Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. Am.*, vol. 89, pp. 425–434, 1991.

[4] David Mount and Sunil Arya, "ANN: Library for approximate nearest neighbor searching," http://www.cs.umd.edu/~mount/ANN/, Last Visited 07 July 2003.

[5] Craig Nicol, Stephen A. Brewster, and Philip Gray, "A system for manipulating audio interfaces using timbre spaces," in *Computer-Aided Design of User Interfaces IV (CADUI'2004)*, R. Jacob, Q. Limbourg, and J. Vanderdonckt, Eds., Madiera, Portugal, 2004, pp. 359–371, Kluwer.

[6] Craig Nicol, Stephen A. Brewster, and Philip Gray, "Designing sound: Towards a system for designing audio interfaces using timbre spaces," in *Tenth International Conference on Auditory Display (ICAD 04)*, Sydney, Australia, 2004, ICAD.

[7] William W. Gaver, "The SonicFinder: An interface that uses auditory icons.," *Human-Computer Interaction*, vol. 4, no. 1, pp. 67–94, 1989.

[8] M M Blattner, D A Sumikawa, and R M Greenberg, "Earcons and icons: Their structure and common design principles," *Human Computer Interaction*, vol. 4, no. 1, pp. 11–44, 1989.

[9] William W. Gaver, "Synthesizing auditory icons," in *ACM Interchi '93*. 1993, ACM.

[10] Elizabeth D. Mynatt, Maribeth Back, Roy Want, Michael Baer, and Jason Ellis, "Designing audio aura," in *CHI '98*. 1998, ACM.

[11] Graeme W. Coleman, Chris Hand, Catriona Macaulay, and Alan F. Newell, "Approaches to auditory interface design - lessons from computer games," in *Proceedings of ICAD 05*, Limerick, Ireland, July 6-9 2005, Eleventh Meeting of the International Conference on Auditory Display, ICAD.

[12] Christophe Hourdin, Gérard Charbonneau, and Tarek Moussa, "A multidimensional scaling analysis of musical instruments' time-varying spectra," *Computer Music Journal*, vol. 21, no. 2, pp. 40–55, 1997.

[13] J. Grey, "Timbre discrimination in musical patterns," *Journal of the Acoustical Society of America*, vol. 64, pp. 467–72, 1977.

[14] William W. Gaver, "How do we hear in the world? Explorations in ecological acoustics," *Ecological Psychology*, vol. 5, no. 4, pp. 285–313, 1993.

[15] Barry Vercoe, "CSound front page," `http://www.csounds.com`, Last visited: 20 Jun 02.

[16] Paul Masri, Andrew Bateman, and C. N. Canagarajah, "A review of time-frequency representations, with application to sound/music analysis-resynthesis," *Organised Sound*, vol. 2, no. 3, pp. 193–205, 1997.

[17] Rudolf Rasch and Reiner Plomp, "The perception of musical tones," In Deutsch [155], chapter 4, pp. 89–112.

[18] H. L. F. von Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, Dover (1954), New York, 4th edition, 1877.

[19] Stephen McAdams and Jean-Christophe Cunibile, "Perception of timbral analogies," *Philosophical Transactions of the Royal Society of London, Series B - Biological Sciences*, vol. 336, no. 1278, pp. 383–389, 1992.

[20] Oxford University Press, "Oxford English Dictionary," `http://www.oed.com`, 2001.

[21] International Organization for Standardization, *Normal equal loudness contours for pure tones and normal threshold of hearing under free-field listening conditions*, ISO, second edition, 2003, Recommendation R 226.

[22] T. H. Andersen and K. Jensen, "Importance of phase in sound modeling of acoustic instruments," in *Mosart Midterm Meeting Report.* Aalborg University, 2002.

[23] Jean-Claude Risset and David L. Wessel, "Exploration of timbre by analysis and synthesis," In Deutsch [155], chapter 5, pp. 113–171.

[24] R. Wegel and C. Lane, "The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear," *Physics Review*, vol. 23, pp. 266–285, 1924.

[25] Norman M. Weinberger, "Music and the auditory system," In Deutsch [155], chapter 3, pp. 47–88.

[26] Oscar S. M. Marin and David W. Perry, "Neurological aspects of music perception and performance," In Deutsch [155], chapter 17, pp. 653–724.

[27] J. J. Gibson, *The Ecological Approach to Visual Perception*, Houghton-Mifflin, Boston, 1979, repr. 1986, Hillsdale, NJ: Lawrence Erlbaum.

[28] Elizabeth D. Mynatt, "Designing with auditory icons," in *CHI '94*, Boston, Massachusetts, 1994, ACM.

[29] N. Vanderveer, *Ecological acoustics: Human perception of environmental sounds*, Doctoral thesis, Dissertation Abstracts International, 1979.

[30] Judy Edworthy, Elizabeth Hellier, and Rachael Hards, "The semantic associations of acoustic parameters commonly used in the design of auditory information and warning signals," *Ergonomics*, vol. 38, no. 11, pp. 2341–2361, 1995.

[31] Judy Edworthy, Sarah Loxley, and Ian Dennis, "Improving auditory warning design: Relationship between warning sound parameters and perceived urgency," *Human Factors*, vol. 33, no. 2, pp. 205–231, 1991.

[32] Kees van den Doel and Dinesh K. Pai, "The sounds of physical shapes," *Presence: Teleoperators and Virtual Environments*, vol. 7, no. 4, pp. 382–395, 1996.

[33] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai, "FoleyAutomatic: Physically-based sounds effects for interactive simulation and animation," in *ACM SIGGRAPH*, 2001.

[34] Matti Karjalainen, Vesa Välimäki, and Tero Tolonen, "Plucked-String models: From the Karplus-Strong algorithm to digital waveguides and beyond," *Computer Music Journal*, vol. 22, no. 3, pp. 17–32, 1998.

[35] Matti Karjalainen, "New techniques and effects in model-based sound synthesis," in *2nd COST G-6 Workshop on Digital Audio Effects (DAFx99)*, Trondheim, 1999.

[36] Stephane Conversy, "Ad-hoc synthesis of auditory icons," in *ICAD 98: Fifth International Conference on Auditory Display*, S. A. Brewster and A. D. N. Edwards, Eds., Glasgow, UK, 1998.

[37] Elizabeth D. Mynatt, Maribeth Back, Roy Want, and Ron Frederick, "Audio aura: Light-weight audio augmented reality," in *UIST*, Banff, Alberta, Canada, 1997, ACM.

[38] W. Dixon Ward, "Absolute pitch," In Deutsch [155], chapter 8, pp. 265–298.

[39] Jessica J. Baldis, "Effects of spatial audio on memory, comprehension, and preference during desktop conferences," in *SIGCHI'01*, Seattle, WA, USA, 2001, vol. 3:1, pp. 166–73, ACM.

[40] Gavin Andresen, "Playing by ear: Creating blind-accessible games," `http://www.gamasutra.com/resource_guide/20020520/andersen_01.htm`, 2002.

[41] Diana Deutsch, "Grouping mechanisms in music," In *The Psychology of Music* [155], chapter 9, pp. 299–348.

[42] David McGookin and Stephen A. Brewster, "An investigation into the identification of concurrently presented earcons," in *Proceedings of ICAD 2003*, Boston, MA, 2003, ICAD.

[43] T. Dutoit, *An introduction to Text-to-Speech Synthesis*, Kluwer Academic Publishers, 1997.

[44] Hiroko Terasawa, Malcolm Slaney, and Jonathan Berger, "Perceptual distance in timbre space," in *Proceedings of ICAD 05*, Limerick, Ireland, July 6-9 2005, Eleventh Meeting of the International Conference on Auditory Display, ICAD.

[45] Damiàn Keller and Barry Truax, "Ecologically-based granular synthesis," in *ICMC 1998*, Ann Arbor, Michigan, 1998.

[46] D. Rocchesso, R. Bresin, and M. Fernstrom, "Sounding objects," *IEEE Multimedia*, vol. 10, no. 2, pp. 42–52, 2003.

[47] S.A. Brewster, P.C. Wright, and A.D.N. Edwards, "A detailed investigation into the effectiveness of earcons," in *First International Conference on*

Auditory Display, G. Kramer, Ed., Santa Fe Institute, Santa Fe, NM, 1992, vol. Auditory display, sonification, audification and auditory interfaces., pp. 471–498, Addison-Wesley.

[48] Stephen A. Brewster, P.C. Wright, and A.D.N. Edwards, "An evaluation of earcons for use in auditory human-computer interfaces.," in *InterCHI'93*, S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White, Eds., Amsterdam, 1993, pp. 222–227, ACM Press, Addison-Wesley.

[49] Judy Edworthy and R. Hards, "Learning auditory warnings: The effects of sound type verbal labelling and imagery on the identification of alarm sounds," *International Journal of Industrial Ergonomics*, vol. 24, no. 6, pp. 603–618, 1999.

[50] David K. McGookin and Stephen A. Brewster, "Space, the final frontearcon: The identification of concurrently presented earcons in a synthetic spatialised auditory environment," in *Proceedings of ICAD2004*, Sydney, Australia, 2004, ICAD.

[51] William W. Gaver, Randall B. Smith, and Tim O'Shea, "Effective sounds in complex systems: the ARKOLA simulation," in *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems.* 1991, pp. 85–90, ACM Press.

[52] NSF/ICAD, "Sonification report: Status of the field and research agenda," in *The International Community for Auditory Display*, G Kramer and B. Walker, Eds., Santa Fe, 1999.

[53] Florian Dombois, "Using audification in planetary seismology," in *International Conference on Auditory Display (ICAD)*, Espoo, Finland, 2001.

[54] T. Hermann and H. Ritter, "Sound and meaning in auditory data display," *Proceedings of the IEEE*, vol. 92, no. 4, pp. 730–741, 2004.

[55] Rooms3D, "Rooms3d desktop interface," `http://www.rooms3d.com`, Last visited: 20 Jun 2002.

[56] Web3D, "Web3D consortium: VRML and X3D," `http://www.web3d.org`, Last visited: 20 Jun 2002.

[57] MIDI Manufacturers Association, *MIDI – Musical Instrument Digital Interface*, International MIDI Association, Los Angeles, CA, 1988, Specification 1.0.

[58] Curtis Roads, *The Computer Music Tutorial*, Massachusetts Institute of Technology, 1996.

[59] R.C. Maher, "Wavetable synthesis strategies for mobile devices," *Journal of the Audio Engineering Society (JAES)*, vol. 53, no. 3, pp. 205–212, March 2005.

[60] Thomson multimedia, "MP3 specification and licensing," `http://www.mp3licensing.com`, Last visited: 18 Jun 2002.

[61] Karl Geiringer, *Musical Instruments. Their History from the Stone Age to the Present Day*, New York, 1946.

[62] T. Cahill, "Patents for the telharmonium," Washington, D.C.: U.S. Patent office, 1897, U.S. Patents 580,035 (1897);1,107,261 (1914);1,213,803 (1917);1,295,691 (1919).

[63] Christophe Hourdin, Gérard Charbonneau, and Tarek Moussa, "A sound-synthesis technique based on multidimensional scaling of spectra," *Computer Music Journal*, vol. 21, no. 2, pp. 56–68, 1997.

[64] Georgios Marentakis and Kristoffer Jensen, "The Timbre Engine - progress report," in *Workshop on Current Research Directions in Computer Music*, `http://www.iua.upf.es/mtg/mosart`, Ed., Barcelona, Spain, 2001, p. p07, MOSART IHP Network.

[65] John Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society (JAES)*, vol. 21, no. 7, pp. 526–534, 1973.

[66] John Chowning and David Bristow, *FM Theory And Applications: by Musicians for Musicians*, Yamaha Music Foundation, Tokyo, Japan, 1986.

[67] James Beauchamp and Andrew Horner, "Spectral modelling and timbre hybridisation programs for computer music," *Organised Sound*, vol. 2, no. 3, pp. pp253–258, 1997.

[68] J.-F. Allouis, "The use of high-speed microprocessors for sound synthesis," *Computer Music Journal*, vol. 3, no. 1, pp. 14–16, 1979.

[69] J.-F. Allouis and J.-Y. Bernier, "The SYTER project: sound processor design and software overview.," in *Proceedings of the 1982 International Computer Music Conference*, J. Strawn and T. Blum, Eds., San Francisco, 1982, pp. 232–240, International Computer Music Association.

[70] J. L. Flanagan, *Speech Analysis, Synthesis, and Perception*, Springer-Verlag, New York, 1972.

[71] R. Kronland-Martinet, Ph. Guillemain, and Sølvi Ystad, "Modelling of natural sounds by time-frequency and wavelet representations," *Organised Sound*, vol. 2, no. 3, pp. 179–91, 1997.

[72] Damiàn Keller and Jonathan Berger, "Everyday sounds: synthesis parameters and perceptual correlates," in *Proceedings of the VIII Brazilian Symposium*

of Computer Music, Fortaleza, 2001, CE: SBC.

[73] Kevin Karplus and Alan Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, 1983.

[74] Tero Tolonen, Vesa Välimäki, and Matti Karjalainen, "Modelling of tension modulation nonlinearity in plucked strings," in *IEEE Transactions on Speech and Audio Processing*, vol. 8, p. 3. IEEE, 3 edition, 2000.

[75] Joel Laird, Paul Masri, and Nishan Canagarajah, "Modelling diffusion at the boundary of a digital waveguide mesh," in *International Computer Music Conference (ICMC)*, 1999.

[76] Joel Laird, Tom Huns, Paul Masri, and Nishan Canagarajah, "WaRM: A framework for modelling resonant acoustic environments," in *International Computer Music Conference (ICMC)*, Beijing, 1999, pp. 545–548, The International Computer Music Association (ICMA).

[77] Marco Palumbi and Lorenzo Seno, "Metal string: Physical modelling of bowed strings - a new model and algorithm.," in *JIM '98: Journées d'Informatique Musicale*, La Londe-les-Maures, France, 1998.

[78] I Kaminskyj, "Multidimensional scaling analysis of musical instrument sounds' spectra," in *Australasian Computer Music Conference (ACMC)*, Wellington, NZ, 1999, pp. 36–9.

[79] Pierre-Yves Rolland and François Pachet, "A framework for representing knowledge about synthesizer programming," *Computer Music Journal*, vol. 20, no. 3, pp. 47–58, 1996.

[80] Stephen McAdams, Suzanne Winsberg, Sophie Donnadieu, Geert De Soete, and Jochen Krimpoff, "Perceptual scaling of synthesised musical timbres:

common dimensions, specificities, and latent subject classes.," *Psychological Research*, vol. 58, pp. pp177–192, 1995.

[81] Matthew Wright, James Beauchamp, Kelly Fitz, Xavier Rodet, Axel Röbel, Xavier Serra, and Gregory Wakefield, "Analysis/synthesis comparison," *Organised Sound*, vol. 5, no. 3, pp. 173–189, 2000.

[82] Paul Masri, Andrew Bateman, and Nishan Canagarajah, "The importance of the time-frequency representation for sound/music analysis-resynthesis," *Organised Sound*, vol. 2, no. 3, pp. 207–14, 1997.

[83] SDIF, "Sound description interface format: Specification and home page," `http://cnmat.cnmat.berkeley.edu/SDIF/`, Last visited: 18 Jun 2002.

[84] Matthew Wright, A. Chaudhary, A. Freed, D. Wessel, X Rodet, D. Virolle, R. Woehrmann, and Xavier Serra, "New applications of the sound description interchange format," in *Int. Computer Music Conf.*, Ann Arbor, Michigan, 1998, pp. pp. 276–9.

[85] Matthew Wright, A. Chaudhary, A. Freed, S. Khoury, and D. Wessel, "Audio applications of the sound description interchange format standard," in *Audio Engineering Society 107th Convention*, 1999.

[86] Matthew Wright, A. Chaudhary, A. Freed, S. Khoury, and D. Wessel, "An XML-based SDIF stream relationships language," in *Int. Computer Music Conf.*, Berlin, Germany, 2000.

[87] F.J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, January 1978.

[88] J.F. Kaiser, "Nonrecursive digital filter design using the sinh window function," in *Proc. 1974 IEEE Symp. Circuits and Systems*, April 1974, pp. 20–23.

[89] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, April 1928.

[90] E.P. Wigner, "On the quantum correction for thermodynamic equilibrium," *Phys. Rev.*, , no. 40, pp. 749–759, June 1932.

[91] J. Ville, "Thorie et applications de la notion de signal analytique," *Cables et Transmission*, , no. 2A, pp. 61–74, 1948.

[92] R. Kronland-Martinet, "The wavelet transform for analysis, synthesis, and processing of speech and music sounds," *Computer Music Journal*, vol. 12, no. 4, pp. 11–20, 1988.

[93] M. D. Freedman, "Analysis of musical instrument tones," *Journal of the Acoustical Society of America*, vol. 41, pp. 793–806, 1967.

[94] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of the complex fourier series," *Math. Computation*, vol. 19, pp. 297301, Apr 1965.

[95] J. B. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-25, pp. 235–238, June 1977.

[96] H. M. Ozaktas, Z. Zalevsky, and M. A. Kutay, *The Fractional Fourier Transform, with Applications in Optics and Signal Processing*, Wiley, New York, 2000.

[97] Tony S Verma and Theresa H. Y. Meng, "Time scale modification using a sines+transients+noise signal model," in *Proceedings of First COST-G6 Workshop on Digital Audio Effects (DAFX98)*, Barcelona, Spain, November 19-21 1998.

[98] Ye Wang and Miikka Vilermo, "The modified discrete cosine transform: Its implications for audio coding and error concealment," in *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio.* 2002, Audio Engineering Society.

[99] T. L. Petersen, "Acoustic signal processing in the context of a perceptual model," Tech. Rep. UTEC-CSc-80-113, University of Utah, Department of Computer Science, Salt Lake City, 1980.

[100] Judy Brown, "Constant Q transform for MATLAB," `http://web.media.mit.edu/~brown/cqtrans.htm`, Last visited 04 Feb 02.

[101] H. Dudley, "Synthesising speech," Tech. Rep., Bell Laboratories Record, December 1936, 98-102.

[102] N. Bailey and D. Cooper, "Perceptually smooth timbral guides by state-space analysis of phase-vocoder parameters," *Computer Music Journal*, vol. 24, no. 1, pp. 32–42, 2000.

[103] R. Kronland-Martinet and Ph. Guillemain, "Towards non-linear resynthesis of instrumental sounds," in *1993 International Computer Music Conference*, San Francisco, 1993, pp. 86–93, International Computer Music Association.

[104] J. Buckheit, S. Chen, D. Donoho, I. Johnstone, and J. Scarg le, "Wavelab reference manual," 1995.

[105] Mathworks, *Matlab*, Mathworks, Inc., Natick, MA, 1999.

[106] Sam Roweis and Zoubin Ghahramani, "A unifying review of linear gaussian models," *Neural Computation*, vol. 11, no. 2, 1999.

[107] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[108] R. Penrose, "A generalized inverse for matrices," *Proc. Cambridge Phil. Soc.*, , no. 51, pp. 406–413, 1955.

[109] Addad Software, "Data analysis software," 1989.

[110] D. E. Rumelhart and A. A. Abrahamson, "A model for analogical reasoning," *Cognitive Psychology*, , no. 5, pp. 1–28, 1973.

[111] Michiel Smid, "Closest point problems in computational geometry," in *Handbook on Computational Geometry*, J.-R. Sack and Jorge Urrutia, Eds., pp. 877–935. Elsevier Science, Amsterdam, 1999.

[112] Thatcher Ulrich, "Loose octrees," in *Game Programming Gems*, Mark DeLoura, Ed. Charles River Media, Hingham, Massachusetts, August 2000.

[113] Jon Louis Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[114] Jerome H. Freidman, Jon Louis Bentley, and Raphael Ari Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[115] Rinie Egas, Dionysius P. Huijsmans, Michael S. Lew, and Nicu Sebe, "Adapting k-d trees to visual retrieval," in *Visual Information and Information Systems*, Berlin, 1999, pp. 533–540, Springer-Verlag.

[116] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[117] B Scholkopf, A Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, July 1998.

[118] Perry Moerland, "An on-line EM algorithm applied to Kernel PCA," Idiap research report, Dalle Molle Institute for Perseptual Artificial Intelligence, September 2000 2000.

[119] T Kohonen and E Oja, "Visual feature analysis by the self-organising maps," *Neural Computing and Applications*, vol. 7, no. 3, pp. 273–286, 1998.

[120] Perry R. Cook, "Physically informed sonic modelling (PhISM): Synthesis of percussive sounds," *Computer Music Journal*, vol. 21, no. 3, pp. 38–49, 1997.

[121] D.N. Uznadze, *Experimental basis of the psychology of set*, Academy of Science, Tbilisi, Georgia, 1961.

[122] D. W. Grantham, "Spatial hearing and related phenomena," in *Hearing*, B. C. J. Moore, Ed., pp. 297–345. Academic Press, San Diego, CA, 1995.

[123] Thilo Thiede, William C. Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G. Beerends, and Catherine Colomes, "PEAQ - the ITU standard for objective measurement of perceived audio quality," *J. Audio Eng. Soc.*, vol. 48, pp. 3–29, Jan.-Feb. 2000.

[124] P. Kabal, "An examination and interpretation of ITU-R BS.1387: Perceptual evaluation of audio quality," Tech. Rep., TSP Lab Technical Report, Dept. Electrical & Computer Engineering, McGill University, May 2002, `http://www.TSP.ECE.McGill.CA/MMSP/Documents`.

[125] N Delprat, Ph. Guillemain, and R. Kronland-Martinet, "Parameter estimation for non-linear resynthesis methods with the help of a time-frequency analysis of natural sounds," in *International Computer Music Conference*, Glasgow, 1990, pp. 88–90, International Computer Music Association.

[126] Ph. Guillemain, R. Kronland-Martinet, and Sølvi Ystad, "Physical modelling based on the analysis of real sounds," in *Proceedings of the Institute of Acous-*

tics, vol. 19, pp. 445–50. ICMA97, Edinburgh, 1997.

[127] Kristoffer Jensen, "The Timbre Model," in *Workshop on Current Research Directions in Computer Music*, `http://www.iua.upf.es/mtg/mosart`, Ed., Barcelona, Spain, 2001, MOSART IHP Network.

[128] Xavier Serra, *A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition*, Ph.d. dissertation, Stanford, 1987.

[129] Axel Röbel, "Adaptive additive synthesis of sound," in *ICMC*, Beijing, China, 1999, pp. pp. 256–9.

[130] Microsoft, "Microsoft interface definition language," `msdn.microsoft.com/library/en-us/rpc/rpc/the_interface_definition_langu%age_idl_file.asp`, Last visited: 23 Aug 2005.

[131] F. Opolko and J. Wapnick, "MUMS–McGill University Master Samples," (in compact discs), 1987, Montreal, Canada: McGill University.

[132] P. Bezier, *Mathematical and Practical Possibilities of UNISURF.*, Academic Press, New York, 1974.

[133] Steinberg, "VST specification and SDK," `http://service.steinberg.de/webdoc.nsf/show/development_e`, Last Visited: 17 Jun 2002.

[134] Steinberg, "Steinberg home page," `http://www.steinberg.net`, Last Visted: 17 Jun 2002.

[135] Cakewalk, "DirectX / DXi documentation and SDK," `http://www.directxfiles.com`, Last visited: Jun 17 2002.

[136] Cakewalk, "Cakewalk home page," `http://www.cakewalk.com`, Last visited: Jun 17 2002.

[137] Richard W. E. Furse, "Linux audio developer's simple plugin API (LADSPA)," www.ladspa.org, Last visited: 23 Aug 2005.

[138] Cakewalk, "SONAR," http://www.cakewalk.com/Support/SONAR/default.asp, Last visited: 23 Aug 2005.

[139] Microsoft, "Microsoft home page," http://www.microsoft.com, Last visited: 17 Jun 2002.

[140] "WINE HQ," http://www.winehq.com, Last visited: 12 August 2005.

[141] Transgaming Technologies, "Cedega," http://www.transgaming.com, Last visited: 12 August 2005.

[142] Mozilla Foundation, "XPCOM API," http://www.mozilla.org/projects/xpcom/, Last visited: 12 August 2005.

[143] "Simple Directmedia Layer," http://www.libsdl.org, Last visited: 12 August 2005.

[144] "Agnula," http://www.agnula.org, Last visited: 12 August 2005, European Commision contract: IST-2001-34879; key action IV.3.3, Free Software: towards the critical mass.

[145] David M. Howard and Andy M. Tyrell, "Psychoacoustically informed spectrography and timbre," *Organised Sound*, vol. 2, no. 2, pp. 65–76, 1997.

[146] C.P. Dancey and J. Reidy, *Statistics Without Maths for Psychology: Using SPSS for Windows.*, Pearson Education Limited, Essex, second edition, 2002.

[147] Mike Brookes, "Voicebox: Speech processing toolbox for MATLAB," http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html, Last visited 23 September 2005.

[148] Malcolm Slaney, "Auditory toolbox, version 2," Tech. Rep. 1998-010, Interval Research Corporation, 1998.

[149] B. P. Challis, J. C. K. Hankinson, and A. D. N. Edwards, "Musical quotient: a measure of musical ability.," Tech. Rep., University of York, Department of Computer Science, 1999, York Computer Science report.

[150] Claude E. Shannon and Warren Weaver, *The Mathematical Theory of Communication*, Univ of Illinois Press, 1963, ISBN 0252725484.

[151] Michel Beaudouin-Lafon and William W. Gaver, "ENO: Synthesizing structured sound spaces," in *UIST '94*, Marina Del Rey, California, 1994, ACM.

[152] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krger, Aaron E. Lefohn, and Timothy J. Purcell, "A survey of general-purpose computation on graphics hardware," in *Eurographics 2005, State of the Art Reports*, Aug. 2005, pp. 21–51.

[153] Sean Whalen, "Audio and the graphics processing unit," `http://www.node99.org/projects/gpuaudio/`, March 10 2005.

[154] Shreekant (Ticky) Thakkar and Tom Huff, "The Internet Streaming SIMD Extensions," *Intel Technology Journal*, , no. Q2, pp. 8, May 1999.

[155] Diana Deutsch, Ed., *The Psychology of Music*, Series in Cognition and Perception. Academic Press, second edition, 1999.

Appendix A. COM interface definitions

```
//////////////////////////////////////////////////////////////////////////
// Timbre Control Interfaces here describe the COM
// connections in the Timbre Control library, allowing the analysis
// portions and the synthesis portions of the sound engine to
// communicate.
//////////////////////////////////////////////////////////////////////////
// There are 3 important interfaces to consider:
//
// TimbreSynth (sink), accepts a parameter list and synthesises a sound
//          from these.
//
// ParameterSource, generates a stream of parameters for the synth
//          either from a file or from the path in timbre space sent from
//          the local analyser or a remote process.
//          This can communicate with multiple synths, choosing the best
//          one depending on system parameters such as load and output
//          configuration, and on the portion of timbre space covered by
//          the path.
//          If a path is sent, this source must generate the set of points
//          possibly in real time (i.e. generate 1 point every 4ms, whilst
//          the synth is active).
//
// TimbreSource, generates a path in timbre space either from an input
//          file or via manipulations of the space, graphically or
//          mathematically.
//
//////////////////////////////////////////////////////////////////////////
// There are 2 important variable types sent between the COM objects:
//
// ParameterList, passed from ParameterSource to TimbreSynth.
//          Lists a set of parameters, in order, for the synth to
//          synthesize. This list is unique to each synth, but has common
//          components such as frequency, time and other MIDI-defined
//          parameters.
//
// TimbrePath, passed from TimbreSource to ParameterSource.
//          This is either an time-ordered list of points defining a path
//          in timbre space or a function (from a predefined set)
//          to generate such points, which must have a well-defined
//          bounding box so that ParameterSource can choose the proper
//          synthesis for the path.
//          Defined functions will include circles, bounded straight lines,
//          bounded polynomials, bezier curves (possibly).
//
//////////////////////////////////////////////////////////////////////////
```

```
[
     uuid(362AA78C-1609-429e-9C41-0620A0C2EE66),
     helpstring("Timbre Space Devices Type Library"),
     version(0.01)
]
library TimbreControl
{
     importlib("stdole32.tlb");
     importlib("stdole2.tlb");

     // We need midifilter for the definition of IMfxSoftSynth
     // from the DXi specification
     importlib("midifilter.tlb");

     struct parameterEntry
     {
          enum parType
          {
               parSigned,

               parSignedInt8,     // i.e. 8 bit signed int
               parSignedInt16,
               parSignedInt32,
               parSignedInt64, // Currently unused

               parFloat32,
               parFloat64,

               parUnicodeChar,

               parUnsigned, //0x08

               parUnsignedInt8,
               parUnsignedInt16,
               parUnsignedInt32,
               parUnsignedInt64,     // Currently unused

               parErrorType,

          } paramType;

          char paramaterName[128];

          union parameterValue // Current value
          {
               signed small sint8;
               signed short sint16;
               signed long sint32;
               signed hyper sint64;

               float sfloat32;
               double sfloat64;

               wchar_t sunicode;
```

```
            unsigned small uint8;
            unsigned short uint16;
            unsigned long uint32;
            unsigned hyper uint64;

            signed long paramErrorCode;
        } parValue;

    };

    typedef struct parameterEntry parEntry;

    struct TimbrePath
    {
  int Dimensions;     // Number of dimensions for timbre space
  int numParameters; // Number of function parameters
  int numPoints;      // Number of points/dimensions

        enum TypesOfPath
        {
            pathListOfPoints,      // Absolute points
            pathListOfVectors,     // Points relative to each other
            pathRandomGaussian,    // Random clustered points
                // Insert other path types in here

            pathError = 0x7FFF
        } PathType;

  // Will be a [Dimensions] or [numParameters] array
  // of types.
  parameterEntry* pointType;

        // This PathData will use pointType for type checking.
  // ListOfPoints should point to a [numPoints][Dimensions]
  // array of numPoints points in timbre space.
  // FunctionParameters should point to a [numPoints][numParameters]
  // array of numPoints parameters, and the function will
  // return [Dimensions] parameters.
        union PathData
        {
            void* ListOfPoints;
            void* FunctionParameters;
        };

  double[2]* BoundingBox; //List of (start, end) pairs for each dimension
    };

    struct ParameterList
    {
        LONG midiTime;
        float Frequency;
        char instrument;
        char masterVolume;
        LONG midiDuration;
        int NumOfParameters;
        parEntry * ListOfParameters;
    };

    [
        uuid(3987CF6E-7036-4353-8EE7-FF5D2AEFCF40),
        helpstring("Timbre Space Synthesiser Sink")
    ]
    interface ITimbreSynth : IMfxSoftSynth
        // Uses IMfxSoftSynth to accept MIDI data and output a PCM buffer
    {
        HRESULT getListOfParameters
                ([out, retval] struct ParameterList *parameterType);

        HRESULT getNumberOfParameters([out, retval] int* NumOfParameters);

        HRESULT setParameter
                ([in] int paramNum, [in] parEntry value, [in] LONG midiTime);

        HRESULT getParameter
                ([in] int paramNum, [out, retval] parEntry * value);

        // use set/getParameter when possible, as ByName is slower
        HRESULT setParameterByName
                ([in] char *paramName, [in] parEntry value, [in] LONG midiTime);
        // use set/getParameter when possible, as ByName is slower
        HRESULT getParameterByName
                ([in] char *paramName, [out, retval] parEntry * value);
        HRESULT checkParameterType
                ([in] int paramNum, [out] enum parType *parameterType);

        // Parameters should be listed in the same order
        // as getListOfParameters()
        HRESULT setAllParameters([in] struct parameterList * valueList);
    };

    // A parameterSource class also needs to inherit from IMfxTempoMap
    // if you would like to use it as an MFX/DXi host
    [
        uuid(77D6283A-E95F-473c-93D4-EC68D38CC964),
        helpstring("Timbre Space Parameter Source")
    ]
    interface IParameterSource : IUnknown
    {
        [helpstring("Prepare synth and memory")] HRESULT initialise();
        [helpstring("Read parameters from file")]
                  HRESULT sourceFromFile([in] char *filename);
        [helpstring("Accept parameters from external process")]
                  HRESULT onTimbreData([in] struct TimbrePath newPath);
        [helpstring("Generate the required output")] HRESULT transform();
    };

    [
        uuid(20752CC9-5CC0-4f4c-BEE8-65EF754D560C),
        helpstring("Timbre Space Path Source")
    ]
    interface ITimbreSource : IUnknown
    {
        HRESULT pathFromFile([in] char *filename);

        // Sample Buffer should be in a WAVEFORMATEX type
        HRESULT pathFromSampleBuffer([in] void * AudioBuffer);

        // Buffer type should be defined by a WAVEFORMATEX
        HRESULT setBufferType([in] void * bufferType);

        HRESULT getPath([out, retval] struct TimbrePath * outPath);
    };
}
```

Appendix B. Sample parameter file

```
% Parameter file format
%
% File input format for ParameterSpace parameters
% (% indicates a comment until the end of the line):
% [Header] is used to seperate sections of the file
% into header sections and performance sections
% where each performance is recorded into its own file.

[Synth]
name = synthName
path = synthPath
CLSID = synthClassID
% Only CLSID is really required, and
% this is the one to be used when files are computer generated
% Other two are more prone to error but more human-readable,
% so will be more prevalent in user-generated files. Editor will
% automatically add the CLSID if it successfully finds the synth.
% All 3 must agree for load to succeed

% We can then check parameters section against the returned
% parameters from the loaded synth.

[Parameters]
% i.e. the non-midi parameters that we manipulate via
% GetParameters and SetParameters methods
% Not all synth parameters need to be here
% (e.g. if we are only using the two-oscillator FM arrangement, there is no point
% adding details for modIndex of the third oscillator, and so its values
% will be left at default, which the synth should define as 'off'.)
% All parameters must exist in DXi, so we check this before we use it

% Example parameters 1
Param1 = carrierMultiplier
Param2 = modIndex
Param3 = modMultiplier

% Example parameters 2
% Param1 = modIndex

% Range default is 0..127
% Should check this too against DXi
Param1range = 1..50

% type is float | uint8 | uint16 | ... | sint8 | sint16 | ...
Param1type = float

% overflow = wrap
% will modulo values into the range
% overflow = clip
% is the default, will clip values to the desired range
Param1overflow = wrap

% Frequency can be one of:
% float
% note name and octave, e.g. A3, where A3 = MIDI note number 57 = 440 Hz
%     can also use A#3, or Ab3...
%     which is converted to a MIDI number via:
%         lookup table for note number per octave,
%         where: C = 0, D = 2, etc.
%         If sharp then:
%             notenum <- notenum + 1
%             if notenum = 12 then: (i.e. B# = C)
%                 notenum = 0
%                 octave += 1
%             endif
%         Else If flat then ...
%             :
%             :
%         endif
%
%     MIDI number <- ([octave + 1] * 12) + notenum
%
% MIDI note number, e.g. P60 (pitch 60 = middle C = C4)

[MonoOut]
%freq     inst     amp     p1     p2     p3
  440        0      50      1      1      1
% use character '.' to keep old value
   .         .      60      1      1      1
   .         .       .      2      .      .

% Each line = 1 time step
% (but can use a special char to say 'Same for next N lines'

% Use . to repeat a value from above.

% Use
% {N} or {N steps}
%     to repeat the last values for N steps
% {N sec} or {N s} or {N seconds}
%     to repeat the last values for N seconds
% {N ms} or {N msec} or {N milliseconds}
%     to repeat the last values for N milliseconds

% Use # lines to change variables.
% System variables all start with _
% e.g.
#_mspersample = 4
% is the default value, the variable name is equivalent to
% _timestepsize
% _timesteplength
% Only one variable can appear on left, any valid
% equation can appear on right, including variable
% names, and mathematical functions such as sin
% or exp and constants such as PI.

% _STEP is a special variable representing the current
% step number

% _NOW is a special variable representing the current
% time in ms

% To vary a parameter according to a function, the
% syntax
% #x = x + 1 | 1 step
% can be used, to increment x every time step.

% #x = x + 1 | 1sec
% will increment x every second

% #x = x + 1 | 4steps
% will increment x every 4 steps

% #y = y + 1 | 10x
% will increment y every 10th time x changes
% Cannot have same variable on left and right of |
% Cannot have circular references.

% #x = sin(_NOW) | _NOW
% will evaluate x as a sine wave varying against _NOW
% and will evaluate every ms, whenever _NOW updates

% Define a matrix with 2 rows and 4 columns
[=MatrixName@rows=2,columns=4]
1 2 3 4
5 6 7 8

% Use the matrix in an output section
[MonoOut]
% Full matrix:
MatrixName

% First row:
MatrixName@0

% Last 3 columns:
MatrixName@@1..4

% special section for generating a series of output files
% special variable here all begin with m_
[Multifile]

% m_lengthoffile takes units steps, sec, s, seconds, msec, ms, milliseconds
% This is the number of time steps recorded to each file
```

```
m_lengthoffile = 4 steps

% string containing output directory for files
% default is same directory as this file
m_filepath = "C:/my music/"

% filename generating string
% where p1, p2 and p3 are the first 3 parameters
% taken from the first time step for this file
% variables are quoted by //
% variable size is defined by . as in printf
% if a value is not unique, the filename is appended by
% "-N", where N is the lowest value such that
% "filename-N" is unique.
% default value for filename is same as this file
m_filename = "FM/p1//p2//p3/"

% Will output files like:
% FM112.wav
% FM113.wav

% m_filename = "FM/p1.2/-/p2.2/i/p3.3/"
% will output files like:
% FM01-01i002.wav
% FM01-01i003.wav

% maximum number of files to be output from this
m_maxfiles = 100
```

Appendix C. Timbre space comparison

C.1 Comparison of construction of timbre spaces

The following table lists the names of all the spaces tested along with the time to construct and the size of the input TFR. The 400ms STFT spaces, whose names begin ‘fl’, did not complete due to memory errors during their construction.

For a full explanation of the timbre space names see Table 5.2. For a graph of time versus size for all the spaces listed here, please refer to Figure 5.2.

name	Time								Timbre Space Size		
	Year	Month	Day	Hour	Minute	Second	sec	%age of max	# time steps	freq bins	Mb
chsb 2 8	0	0	2	18	38	59.43024	239,939.43	83.37%	66230	485	490.14
chsb 1 8	0	0	2	18	33	7.95759	239,587.96	83.25%	66230	485	490.14
cnsb 2 8	0	0	0	4	12	19.39243	15,139.39	5.26%	66230	121	122.28
cnsb 1 8	0	0	0	4	21	42.00153	15,702.00	5.46%	66230	121	122.28
clsb 2 8	0	0	0	4	10	35.57484	15,035.57	5.22%	66230	20	20.21
clsb 1 8	0	0	0	4	29	41.39875	16,181.40	5.62%	66230	20	20.21
chnb 2 8	0	0	1	10	4	35.59833	122,675.60	42.63%	33295	485	246.40
chnb 1 8	0	0	1	9	14	43.41508	119,683.42	41.59%	33295	485	246.40
cnnb 2 8	0	0	0	2	39	59.90942	9,599.91	3.34%	33295	121	61.47
cnnb 1 8	0	0	0	2	31	21.74888	9,081.75	3.16%	33295	121	61.47
clnb 2 8	0	0	0	1	58	25.63474	7,105.63	2.47%	33295	20	10.16
clnb 1 8	0	0	0	2	3	20.32459	7,400.32	2.57%	33295	20	10.16
chlb 2 8	0	0	0	0	34	37.31417	2,077.31	0.72%	322	485	2.38
chlb 1 8	0	0	0	0	34	41.50562	2,081.51	0.72%	322	485	2.38
cnlb 2 8	0	0	0	0	6	24.65626	384.66	0.13%	322	121	0.59
cnlb 1 8	0	0	0	0	6	19.5198	379.52	0.13%	322	121	0.59
cllb 2 8	0	0	0	0	1	45.44845	105.45	0.04%	322	20	0.10
cllb 1 8	0	0	0	0	0	42.47106	42.47	0.01%	322	20	0.10
chsg 2 8	0	0	2	16	44	58.4759	233,098.48	81.00%	66230	485	490.14
chsg 4 8	0	0	2	17	22	3.776982	235,323.78	81.77%	66230	485	490.14
cnsg 2 8	0	0	0	2	0	19.09696	7,219.10	2.51%	66230	121	122.28
cnsg 4 8	0	0	0	2	0	10.0065	7,210.01	2.51%	66230	121	122.28
clsg 2 8	0	0	0	0	56	45.80085	3,405.80	1.18%	66230	20	20.21
clsg 4 8	0	0	0	0	56	34.49407	3,394.49	1.18%	66230	20	20.21
chng 2 8	0	0	1	9	24	42.51925	120,282.52	41.80%	33295	485	246.40
chng 4 8	0	0	1	8	33	6.119095	117,186.12	40.72%	33295	485	246.40
cnng 2 8	0	0	0	1	15	12.66885	4,512.67	1.57%	33295	121	61.47
cnng 4 8	0	0	0	1	13	57.73943	4,437.74	1.54%	33295	121	61.47
clng 2 8	0	0	0	1	10	52.45876	4,252.46	1.48%	33295	20	10.16
clng 4 8	0	0	0	1	10	28.56088	4,228.56	1.47%	33295	20	10.16
chlg 2 8	0	0	0	0	29	58.68528	1,798.69	0.63%	322	485	2.38
chlg 4 8	0	0	0	0	29	49.52672	1,789.53	0.62%	322	485	2.38
cnlg 2 8	0	0	0	0	13	15.28134	795.28	0.28%	322	121	0.59
cnlg 4 8	0	0	0	0	13	20.48827	800.49	0.28%	322	121	0.59
cllg 2 8	0	0	0	0	0	59.41592	59.42	0.02%	322	20	0.10

cllg 4 8	0	0	0	0	0	59.59031	59.59	0.02%	322	20	0.10
chsk6 2 8	0	0	2	18	24	27.33656	239,067.34	83.07%	66230	485	490.14
chsk6 4 8	0	0	2	17	34	31.27735	236,071.28	82.03%	66230	485	490.14
cnsk6 2 8	0	0	0	3	57	48.51315	14,268.51	4.96%	66230	121	122.28
cnsk6 4 8	0	0	0	3	56	21.43805	14,181.44	4.93%	66230	121	122.28
clsk6 2 8	0	0	0	4	2	26.80527	14,546.81	5.05%	66230	20	20.21
clsk6 4 8	0	0	0	4	2	37.35862	14,557.36	5.06%	66230	20	20.21
chnk6 2 8	0	0	1	8	19	48.24905	116,388.25	40.44%	33295	485	246.40
chnk6 4 8	0	0	1	8	31	34.08966	117,094.09	40.69%	33295	485	246.40
cnnk6 2 8	0	0	0	2	43	41.38928	9,821.39	3.41%	33295	121	61.47
cnnk6 4 8	0	0	0	2	27	16.71679	8,836.72	3.07%	33295	121	61.47
clnk6 2 8	0	0	0	2	0	2.890591	7,202.89	2.50%	33295	20	10.16
clnk6 4 8	0	0	0	1	59	27.37767	7,167.38	2.49%	33295	20	10.16
chlk6 2 8	0	0	0	0	34	46.45407	2,086.45	0.72%	322	485	2.38
chlk6 4 8	0	0	0	0	34	0.115171	2,040.12	0.71%	322	485	2.38
cnlk6 2 8	0	0	0	0	6	23.01405	383.01	0.13%	322	121	0.59
cnlk6 4 8	0	0	0	0	6	17.37062	377.37	0.13%	322	121	0.59
cllk6 2 8	0	0	0	0	0	42.8312	42.83	0.01%	322	20	0.10
cllk6 4 8	0	0	0	0	0	43.1181	43.12	0.01%	322	20	0.10
chsk8 2 8	0	0	2	21	23	37.44872	249,817.45	86.81%	71884	480	526.49
chsk8 4 8	0	0	2	21	32	44.1195	250,364.12	87.00%	71884	480	526.49
cnsk8 2 8	0	0	0	2	47	2.063427	10,022.06	3.48%	71884	120	131.62
cnsk8 4 8	0	0	0	2	44	56.69099	9,896.69	3.44%	71884	120	131.62
clsk8 2 8	0	0	0	1	44	59.58245	6,299.58	2.19%	71884	20	21.94
clsk8 4 8	0	0	0	1	48	33.66745	6,513.67	2.26%	71884	20	21.94
chnk8 2 8	0	0	1	10	22	17.94395	123,737.94	43.00%	35936	480	263.20
chnk8 4 8	0	0	1	9	42	26.15918	121,346.16	42.17%	35936	480	263.20
cnnk8 2 8	0	0	0	1	53	35.14754	6,815.15	2.37%	35936	120	65.80
cnnk8 4 8	0	0	0	1	52	22.002	6,742.00	2.34%	35936	120	65.80
clnk8 2 8	0	0	0	2	7	16.17776	7,636.18	2.65%	35936	20	10.97
clnk8 4 8	0	0	0	2	9	36.9367	7,776.94	2.70%	35936	20	10.97
chlk8 2 8	0	0	0	1	14	51.65578	4,491.66	1.56%	346	480	2.53
chlk8 4 8	0	0	0	1	14	19.9	4,459.90	1.55%	346	480	2.53
cnlk8 2 8	0	0	0	0	22	7.457541	1,327.46	0.46%	346	120	0.63
cnlk8 4 8	0	0	0	0	22	9.996192	1,330.00	0.46%	346	120	0.63
cllk8 2 8	0	0	0	0	2	44.02773	164.03	0.06%	346	20	0.11
cllk8 4 8	0	0	0	0	2	44.52324	164.52	0.06%	346	20	0.11

fsb 1 8	0	0	0	8	18	44.29373	29,924.29	10.40%	66230	45	45.48
fsb 2 8	0	0	1	0	37	23.27149	88,643.27	30.80%	130973	45	89.93
fsb 4 8	0	0	3	7	56	27.84361	287,787.84	100.00%	256234	45	175.94
fmb 1 8	0	0	0	0	29	44.30233	1,784.30	0.62%	33295	89	45.22
fmb 2 8	0	0	0	1	4	34.46687	3,874.47	1.35%	66203	89	89.91
fmb 4 8	0	0	0	2	44	2.795383	9,842.80	3.42%	130920	89	177.79
flb 1 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flb 2 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flb 4 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
fsg 1 8	0	0	0	1	30	1.339834	5,401.34	1.88%	66230	45	45.48
fsg 2 8	0	0	0	3	18	54.79722	11,934.80	4.15%	130973	45	89.93
fsg 4 8	0	0	0	8	40	29.22991	31,229.23	10.85%	256234	45	175.94
fmg 1 8	0	0	0	1	18	5.791431	4,685.79	1.63%	33295	89	45.22
fmg 2 8	0	0	0	4	40	57.83425	16,857.83	5.86%	66203	89	89.91
fmg 4 8	0	0	0	5	31	20.05097	19,880.05	6.91%	130920	89	177.79
flg 1 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flg 2 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flg 4 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
fsk6 1 8	0	0	0	0	33	51.85699	2,031.86	0.71%	66230	45	45.48
fsk6 2 8	0	0	0	1	44	48.50646	6,288.51	2.19%	130973	45	89.93
fsk6 4 8	0	0	0	5	17	59.6609	19,079.66	6.63%	256234	45	175.94
fmk6 1 8	0	0	0	0	32	55.10444	1,975.10	0.69%	33295	89	45.22
fmk6 2 8	0	0	0	1	22	57.1065	4,977.11	1.73%	66203	89	89.91
fmk6 4 8	0	0	0	2	50	45.15149	10,245.15	3.56%	130920	89	177.79
flk6 1 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flk6 2 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flk6 4 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
fsk8 1 8	0	0	0	0	52	0.054427	3,120.05	1.08%	66230	45	45.48
fsk8 2 8	0	0	0	2	17	0.09349	8,220.09	2.86%	130973	45	89.93
fsk8 4 8	0	0	0	5	36	12.25861	20,172.26	7.01%	256234	45	175.94
fmk8 1 8	0	0	0	1	3	13.91335	3,793.91	1.32%	33295	89	45.22
fmk8 2 8	0	0	0	2	6	28.60761	7,588.61	2.64%	66203	89	89.91
fmk8 4 8	0	0	0	5	11	19.71954	18,679.72	6.49%	130920	89	177.79
flk8 1 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flk8 2 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-
flk8 4 8	-1	Out of memory. Type HELP MEMORY for your options.					-	0.00%	-1		-

Appendix D. Urgent sounds experiment

This appendix contains the materials provided to participants and the raw results for the urgency perception experiment described in Chapter 6.

D.1 Experimental materials

The following pages contain the consent form that all participants signed at the start of the experiment and the instruction sheet provided to all participants.

Participant Consent Form:

Urgency Perception Experiment

This experiment aims to discover how people perceive urgency in sounds. In this experiment you will be presented with 50 pairs of sounds and for each pair you will be asked to make a decision on which sounds more "urgent". Urgency is a subjective concept, so there is no right or wrong answer; we just want to know which sounds are more urgent to you.

Some of the pairs may sound very similar; if you find a pair which you feel unable to tell apart please just make a guess. You should not worry if you find it difficult to tell some pairs apart.

If you feel any discomfort or fatigue at any time please inform the experimenter immediately. You can stop and take a break at any time during tasks. The length of time taken to respond is not recorded, therefore you should feel free to take a break for as long as required.

During the experiment the computer will record data on your responses. This data will be stored anonymously using an ID number, rather than your name or any number that could identify you. All results will be held in strict confidence, ensuring the privacy of all participants. There will be no record kept that would allow your results to be traced back to you. All data will also be held in a password protected computer system (for electronic data) or a filing cabinet in a locked office (for paper based data). The data (including that from other participants) will be analysed to identify trends which may be generalised over the larger population. The results of this analysis may be published in appropriate scientific journals and conferences.

A feedback sheet (containing the summaries of the data mentioned above) will be sent to all participants who request it, after the data has been analysed.

Your participation in this experiment will have no effect on your marks for any subject at this, or any other university. You may withdraw from the experiment at anytime without prejudice, and any data already recorded will be overlooked.

If you have any further questions regarding this experiment, please contact:

Craig Nicol,
Department of Computing Science,
17 Lilybank Gardens, Glasgow G12 8QQ
e-mail: can@dcs.gla.ac.uk
tel: +44 0141 330 8430

I have read this information sheet, and agree to take part in this experiment:

Name: ______________________________ Date: ___________

Signature: ___________________________

I would like to receive a summary sheet of the experimental findings □

E-mail Address: ____

Urgency of sounds experiment

If you have any further questions regarding this experiment, please contact:

Craig Nicol *or Stephen Brewster*
Department of Computing Science,
17 Lilybank Gardens, Glasgow G12 8QQ
e-mail: can@dcs.gla.ac.uk *stephen@dcs.gla.ac.uk*
tel: +44 0141 330 8430 *+44 0141 330 4966*

Urgency Perception Experiment

In this experiment you will be presented with pairs of sounds and asked to make a decision on which sounds most "urgent". Urgency is a subjective concept, so there is no right or wrong answer; we just want to know which sounds most urgent to you. Some of the sounds may feel very similar; if you find a pair which you feel unable to tell apart please just make a guess. The experiment will not allow you to carry on if you do not make a selection.

Before you start the experiment please put on the headphones.

When you start the experiment you will see the dialogue shown in Figure 1. You will be able to hear the sounds by clicking on the buttons "Play Sound 1" and "Play Sound 2". You can play each sound up to four times in each task. After four presses, the button will be disabled and will appear greyed out. Once you have made a decision about which you feel is more urgent, make your selection by clicking on the radio button corresponding to your decision, e.g. in Figure 2 the user has selected sound 2.

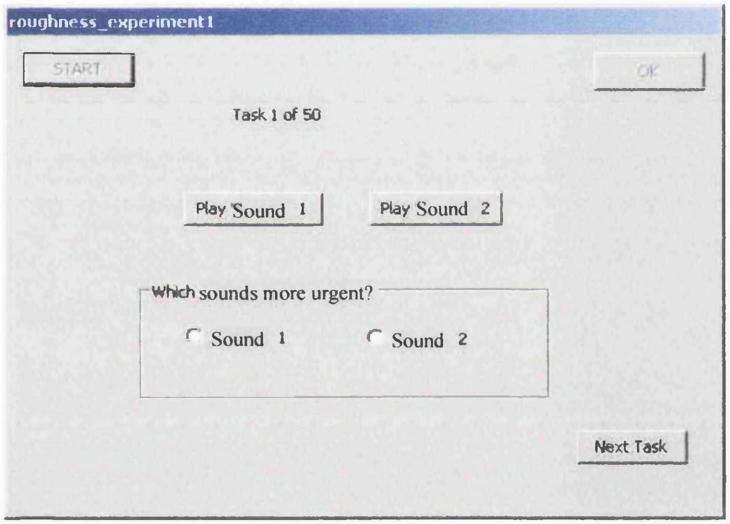


Figure 1: The experiment dialogue

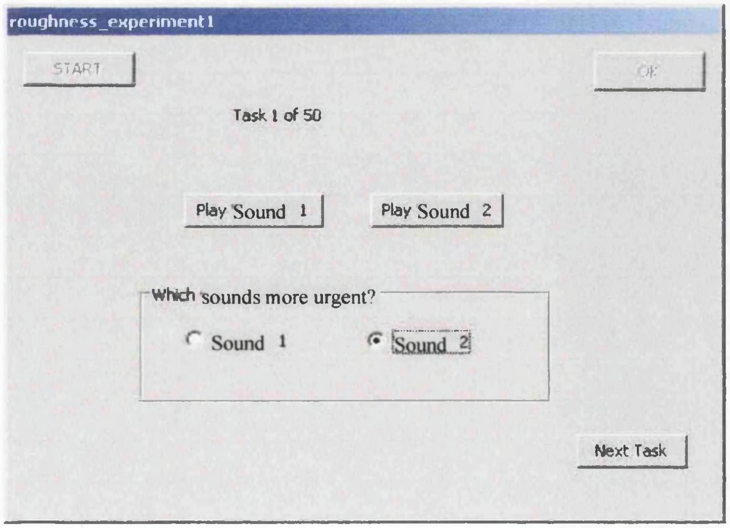
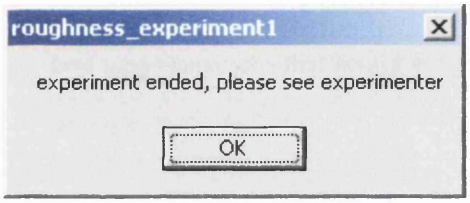


Figure 2: The user has selected "Sound 2" as the more urgent sound

Once you have made your selection, press the "Next Task" button to move onto the next task. You will see the Task Number display change to indicate which task you are currently on, e.g. "Task 2 of 50".

When you press next after the final task (Task 50), you will see the message shown in Figure 3. At this point, please inform the experimenter that you are finished.



experiment ended, please see experimenter

Figure 3: message indicating end of experiment

D.2 Raw data

The next page shows the raw data from the Urgency Perception experiment. Participants are ordered in lexographic order of participant ID. The numbers indicate how many times the first sound in the pair was rated more urgent than the second sound in the pair.

ID	gender	age	musician?	order	v																				g																			
					0>1	0>2	0>3	0>4	1>0	1>2	1>3	1>4	2>0	2>1	2>3	2>4	3>0	3>1	3>2	3>4	4>0	4>1	4>2	4>3	0>1	0>2	0>3	0>4	1>0	1>2	1>3	1>4	2>0	2>1	2>3	2>4	3>0	3>1	3>2	3>4	4>0	4>1	4>2	4>3
urg1250	f	25	y	gv	2	3	0	0	2	3	0	0	1	1	0	0	4	4	4	4	4	4	4	0	1	2	0	0	3	4	0	0	2	0	0	0	4	4	4	0	4	4	4	4
urg1442	m	28	n	vg	1	0	0	0	3	0	0	0	4	4	0	0	4	4	4	4	4	4	4	0	1	0	0	0	3	0	0	0	4	4	0	0	4	4	4	0	4	4	4	4
urg1952	f	27	n	vg	3	3	1	0	1	2	0	0	1	2	0	0	3	4	4	2	4	4	4	2	2	0	0	0	2	0	0	0	4	4	0	0	4	4	4	0	4	4	4	4
urg1984	m	34	y	gv	1	0	0	0	3	0	0	0	4	4	0	0	4	4	4	4	4	4	4	0	2	0	0	1	2	0	0	0	4	4	0	0	4	4	4	1	3	4	4	3
urg254	m	23	n	vg	0	0	0	0	4	1	0	0	4	3	0	0	4	4	4	4	4	4	4	0	0	0	0	0	4	0	0	0	4	4	0	0	4	4	4	0	4	4	4	4
urg2677	m	31	y	vg	2	1	1	3	2	0	1	2	3	4	1	4	3	3	3	2	1	2	0	2	4	4	4	4	0	2	3	4	0	2	4	4	0	1	0	4	0	0	0	0
urg394	m	27	y	vg	1	1	0	0	3	0	0	0	3	4	0	0	4	4	4	0	4	4	4	4	2	0	0	0	2	0	0	0	4	4	0	0	4	4	4	0	4	4	4	4
urg531	m	27	n	gv	2	2	0	0	2	2	0	0	2	2	0	0	4	4	4	1	4	4	4	3	2	0	1	0	2	1	0	0	4	3	0	0	3	4	4	0	4	4	4	4
urg548	f	26	y	gv	3	2	0	0	1	0	0	0	2	4	0	0	4	4	4	0	4	4	4	4	1	2	0	0	3	1	1	0	2	3	0	0	4	3	4	0	4	4	4	4
urg597	m	42	n	gv	0	0	0	0	4	0	0	0	4	4	0	0	4	4	4	0	4	4	4	4	0	1	0	0	4	1	0	0	3	3	0	0	4	4	4	0	4	4	4	4
urg920	f	28	y	vg	2	3	0	0	2	1	0	0	1	3	0	0	4	4	4	0	4	4	4	4	2	0	0	0	2	1	2	0	4	3	0	0	4	2	4	0	4	4	4	4

Appendix E. CD index

All the sounds on the CD are available via the `index.html` file in the root directory. The web pages offer full names for all the instruments and spaces for easier access and also collates important sounds that illustrate the points in this thesis.

All the code used in this thesis is available in the software directory of the CD in a series of ZIP files.

E.1 Timbre space comparison sounds

All sounds taken from the MUMS CDs start with their short name, as provided by Table 5.1 or Table 5.11. Sounds directly from the CD have '_mono' appended to their name whilst sounds that have been filtered through a timbre space have the timbre space name from Table 5.2 appended to their name. Sounds from the DXi Auditory Icon synthesiser have names of the form: 'Gaver' + (material name) + (Sine or String or Bar), where the material name matches those described in Table 5.12. All sounds are available in the 'sounds' directory on the CD, or through the 'Sound Samples' option from the website menu.

E.2 Urgency sounds

The sounds for the urgency experiment are available via the 'Sound Samples' link on the web pages. The sound names all begin with 'urgency_' and are appended by the model used in their construction ('g' for gravity, 'v' for vector), and a number indicating the urgency defined by the model, with 0 indicating least urgent and 4 indicating most urgent.