Audio Segmentation, Classification and Visualization

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy (PhD)

Jessie Xin Zhang

2009

School of Computing and Mathematical Sciences

Primary Supervisor: Dr. Jacqueline Whalley

Table of Contents

List of Figures

List of Tables

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted to the award of any other degree or diploma of a university or other institution of higher learning."

Signed: _____

Acknowledgements

I am very grateful to the many people who encouraged me to begin, continue and finish my study at AUT. First and foremost, I am deeply grateful to my supervisors Dr. Jacqueline Whalley and Dr. Stephen Brooks, for their invaluable suggestions, support, and guidance throughout my period of research. The work presented in this thesis owes much to their enthusiasm and careful guidance.

Dr. Jacqueline Whalley has provided timely guidance and support since the initial stages of this research. Her supervision covered every detail of my study. She offered considerable extra effort as English is my second language. This thesis would not have been possible without her constant support, valuable discussions and detailed reviews. I appreciated all her effort and valuable time.

Thanks to Dr. Stephen Brooks for providing the initial inspiration for the research topic and for the opportunities he provided for me to undertake some of the research under his supervision at Dalhousie University. He has provided valuable guidance and direction at various stages of this research and has generously provided partial financial support towards my attendance at conferences.

David Whalley, I am deeply grateful for the considerable thought and time you put into proofreading my thesis. Thank you for being so patient in correcting any mistakes I've made. I would also like to acknowledge my gratitude to Dr. Qun Song for valuable discussions about the development of the system in this thesis. My thanks go to Felix Suwen Wang and Waseem Ahmad for spending much time reading my work.

Thanks to AUT and SERL for funding my study and providing financial support towards the presentation of papers at overseas conferences. I am grateful to the students in the PhD lab for providing a wonderful community in which to learn and conduct research.

Finally, I would like to add personal thanks to my mother Xuejun Li for her endless encouragement and support, and to my husband Peiliang Zhang for his love and support. Thank you, Mum, you have given me the strength to keep going and to get where I am today.

Abstract

This thesis presents a new approach to the visualization of audio files that simultaneously illustrates general audio properties and the component sounds that comprise a given input file. New audio segmentation and classification methods are reported that outperform existing methods. In order to visualize audio files, the audio is segmented (separated into component sounds) and then classified in order to select matching archetypal images or video that represent each audio segment and are used as templates for the visualization. Each segment's template image or video is then subjected to image processing filters that are driven by audio features. One visualization method reported represents heterogeneous audio files as a seamless image mosaic along a time axis where each component image in the mosaic maps directly to a discovered component sound. The second visualization method, video texture mosaics, builds on the ideas developed in time mosaics. A novel adaptive video texture generation method was created by using acoustic similarity detection to produce a resultant video texture that more accurately represents an audio file. Compared with existing visualization methods such as oscilloscopes and spectrograms, both approaches yield more accessible illustrations of audio files and are more suitable for casual and non expert users.

Preface

Organization of this Thesis

This thesis presents research in the field of content-based audio visualization. The research reported in this thesis is cross disciplinary so the in-depth review of literature of particular relevance to each specific aspect of the research is presented in the applicable chapter.

Chapter 1 introduces the reader to the existing literature in the field of general audio visualization. The motivation for this research is discussed and the research objectives are introduced.

Chapter 2 begins with a description of the methodology used. Then the reader is introduced to the framework of our audio visualization system and the development environment used.

Chapter 3 introduces relevant literature in the field of audio segmentation. Subsequently the development and evaluation of a novel 2-phase audio segmentation method is detailed.

Chapter 4 covers the current literature on audio classification and other relevant classification methods. The development and evaluation of an accurate, adaptive classification method with new class detection is described.

Chapters 5 and 6 describe two alternate and complementary methods for visualizing audio; using images and video textures. The chapters also review the relevant literature on image processing and video texture generation respectively.

Chapter 7 analyzes the audio visualization system and proposes potential avenues for future research and development.

Chapter 1
Introduction

This chapter introduces the motivation for this research and highlights the challenges involved. Prior work in audio visualization is also reviewed.

1.1 Motivation and Research Objectives

With the advance of digital media technologies, the use of digital audio is becoming increasingly widespread. Digital audio files have been used widely in a variety of fields including film, television, computer gaming, radio, website design, and audio book production. Extensive databases have been developed to store digital audio. Management of audio files is therefore becoming more and more important. Research on audio databases and audio file management has included aspects such as audio file classification and audio file search, retrieval and querying, as well as audio database navigation. Providing efficient and effective browsing and navigation tools still remains a major challenge.

Traditional text-based management is not effective with audio databases because an audio file is usually treated as an opaque collection of bytes with primitive fields such as name, file format and sampling rate [1] attached. For example, typical audio database query systems use simple keyword matching of pre-defined tags for each audio file. These results are often not satisfactory, especially when the user is not familiar with the tags for the audio database. Those accustomed to searching, scanning and retrieving text data can be frustrated by the inability to inspect the audio objects. Even if a previous user has assigned keywords or indices to the data, the labeling is often highly subjective and may not be useful for another user [1]. When a user submits a keyword search in an audio database all the audio files in the database are searched and a list of audio files whose indices contain the specified keyword is returned as the result. However, one keyword could be mapped to multiple audio files which may or may not be equivalent

depending on the context. For example, the keyword "rain" could be used to represent drizzle, shower, down pour, rainstorm and thunder storm. This may lead to unexpected query results. As a consequence users may need to spend a significant amount of time listening to the audio files in order to find the desired audio file. Therefore it can be difficult, even for experienced users, to find a specified audio file in a large audio database when using a text-based query system. The subjective nature of assigned keywords and the tedium of manual annotation have motivated researchers to find more efficient and automated methods for audio database management.

The management of small databases is not difficult as it is possible to listen to each file and to select the required sound. For large databases, this is ineffective. Users may find it difficult to distinguish and remember all the examined sounds. This is particularly true for non-expert users. For example, the same note sounds different when played on different instruments. Users may also forget the exact properties of a specific sound heard and may be unable to compare it with another sound from the set of audio files that is listened to at a later time. But even the untrained eye can easily recognize the differences between two images even when they are similar to each other (as in the game "critical seekers" [2]). If it is possible to relate audio files to images, audio visualization may provide a solution to the problem of effective and efficient management of audio databases.

The linear nature of audio files also makes them difficult to navigate and compare. For example, the navigation of an audio database is particularly difficult because the linear nature of audio files requires the sequential examination of each file and each of these examinations may require the same period of time as it takes to play the audio. Even the inspection of query results from an audio database requires considerable time and, as discussed above, provides unreliable results for non-professional users.

Figure 1-1 shows a query for an audio database. The names of resulting audio files do not provide enough information about either the database or the query candidates for viewers to be confident about what the files contain. For example, although viewers can know that all the returned sounds are frog croaks they cannot know which one is louder or which one is competing against background noise.

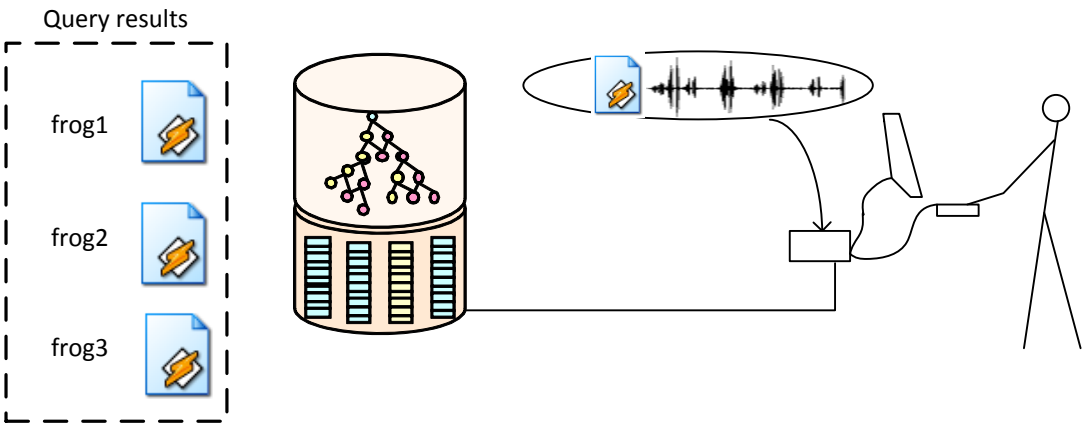

Figure 1-1: Audio database query.

It is much easier to query and navigate image databases because the human visual system can rapidly scan a structured page of thumbnail images. If images or video/video textures could provide accurate visual analogies of audio files, users could visually navigate the whole audio database and rapidly scan through a set of audio candidates returned from a query. Audio visualization would enable an audio database management system to mimic image/video database query and navigation and would therefore benefit from the associated effectiveness. For example, scanning a set of 10 image mosaics representing audio files would require at most a few seconds and is independent of the durations of the audio files they represent, whereas listening to each audio file sequentially would require the sum of the time required to play each file. It has been shown that the average human is in some ways better at seeing than hearing because there are more neurons in human's brains devoted to seeing than to hearing [3].

This research aims to encode audio files visually as images and/or video textures for a single audio input or all audio files in the database. This will allow the viewer to visually interpret an input audio file, browse through an audio database or rapidly scan through a set of audio candidates returned from a query through some defined interface.

Figure 1-2: An example of a time mosaic audio visualization.

Simply put, the objective of this system is to view what happens in the sound sequence. The resultant visualization is a composite image built from simple graphical elements driven by the audio data. The relative positions of image tiles in the resulting mosaic will show the time sequences of their corresponding audio clips (Figure 1-2).

In addition to identifying the component sounds, the visualization conveys more subtle audio features such as loudness, pitch, noise ratio, etc, on a sound-by-sound basis. We propose to achieve this by mapping audio properties to image filtering operations. Video textures will also be employed in this system in order to enhance the visualization of the audio content and properties, and also the visualization of the changes in audio properties over time.

The way in which the system will be used by various users, and the tasks users wish to perform, may vary. For an expert user visualizing audio files might provide rapid access to a subset of audio that they then analyze using expert tools to identify and manipulate the audio files. In the longer term the modular system described in Chapter 2 might readily be extended to accommodate tools specifically for use by experts.

For the novice user this system should provide a means of searching for specific audio files rapidly, either by supplying a query audio or a query image or by browsing using the ontological database structure. For example, a novice might wish to add sound to a digital photo album or select sounds for their cell phone by browsing series of images to select the desired audio. A semi-professional user might wish to identify and locate sounds required for a website or a piece of software that they are developing. Although these users may be computer literate and perhaps even software developers, they are not experts in sound processing. It is anticipated that the system will be used by a broad range of users, from passionate experts to amateurs.

1.2 Challenges in Audio Visualization

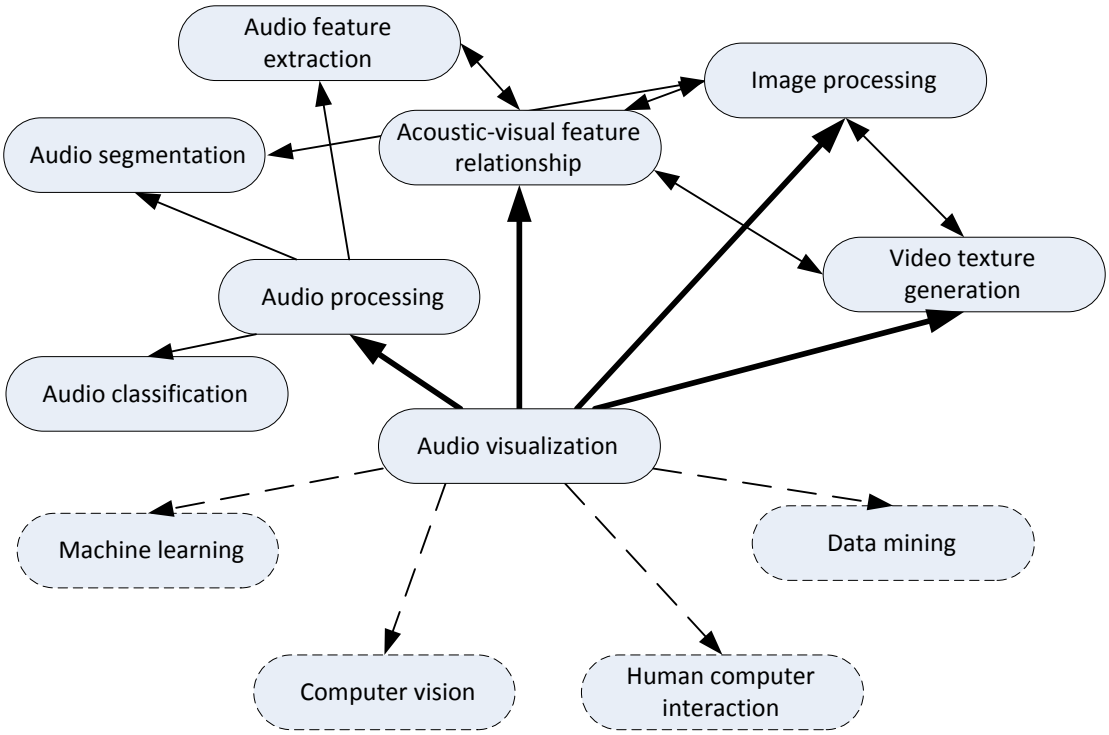

Figure 1-3: Relevant research areas of audio visualization.

Audio visualization research falls under the general area of audio processing and image/video processing. Figure 1-3 shows the main research areas that are relevant to audio visualization. The thickness of the lines is indicative of the importance of each area to this particular research. Reciprocal relationships are indicated in Figure 1-3 by the use of connecting arrows pointing in both directions. So the challenges for audio visualization also come from different fields.

In this domain, a primary requirement for successful visualization is the correct extraction of audio features. The audio features can be used for segmentation, classification, and the production of resultant image/video texture generation.

An audio input may contain more than one sound and an audio visualization system must be capable of analyzing the content of an input audio file while identifying the individual sounds. Segmentation provides a foundation for classification. The accuracy of segmentation limits the accuracy of the visualization results for heterogeneous audio inputs.

Image and video processing are two other important aspects of audio visualization research. As the result of any audio visualization will be an image or video/video texture which can represent the audio input(s), image processing or video texture generation is an essential part of the process.

Acoustic-visual feature relationship research works as a bridge which connects the areas of audio processing and image/video processing. It is based on audio feature extraction and the study of human perception. Although defining the relationship between audio features and the human perception of and reaction to those features is still an open problem, a number of experiments have demonstrated that it is possible for humans to associate auditory and visual percepts [4].

Other related fields include machine learning, computer vision, human computer interaction and data mining. For example, data mining techniques are needed in training a system using a given data set. Audio visualization will benefit from any improvements in these related fields.

To summarize, the audio visualization research reported in this thesis explores the following research questions:

1. Can we design a system to visualize heterogeneous audio inputs?
    a. Can we separate the different sounds and represent the contents accurately?
    b. How can we combine images or video to represent the content of a digital audio file?
2. Is it possible to develop an audio classification method that classifies general sounds accurately enough to allow for visualization of an audio file?
3. Is it possible to classify an unknown sound by integrating an adaptive new-class detection method into an audio classification method?
4. How can we construct an image or image sequence that represents a digital audio file?
    a. How can features of audio files be mapped to the features of images?
    b. How can we combine images to form a sequence that is representative of a digital audio file?
5. How can we synthesize new video sequences that map accurately to the content and sequence of a digital audio file?

6. Can meaningful images and video textures be automatically generated for digital audio files?

Addressing these issues will improve the management and intelligibility of audio databases.

1.3 Audio Visualization Literature Review

This section contains a review of existing approaches to audio visualization. The more detailed literature review for the specific methods employed in each module of our audio visualization system is presented in the corresponding chapter of this thesis.

A history of audio visualization is introduced and then existing approaches to audio visualization are reviewed thematically in the following order:

1. audio properties visualization
2. auditory-visual associations
3. visualizing the structure of an input audio file
4. music visualization
5. speech visualization
6. visualization of audio files with images
7. audio database visualization
8. visualizing similarity of different audio files

1.3.1 Audio Visualization Introduction

The term "audio visualization", also called "sound visualization" has been defined by Nomura, Shiose, Kawakami, Katai and Yamanaka as reading sounds [5]. They found that sound visualization is a viable alternative to listening as a way of understanding audio [5]. Various attempts have been made to visualize different kinds of sounds, some of which were made prior to the invention of the computer. The first attempt at visualizing sound can be traced back to the development of the phonautograph (Figure 1-4). A phonautograph is a device for converting sound into visible traces [6]. Four wave shapes were examined for flute, clarinet, oboe and saxophone tones (C3, "middle" C) using the phonodiek (an advanced phonautograph). The images from a phonodiek

illustrated that the differences of the sounds could be visually presented using different wave shapes. The phonautograph and phonodiek are similar to modern oscilloscopes in that both the phonautograph and modern oscilloscopes depict the wave shape along the time axis.

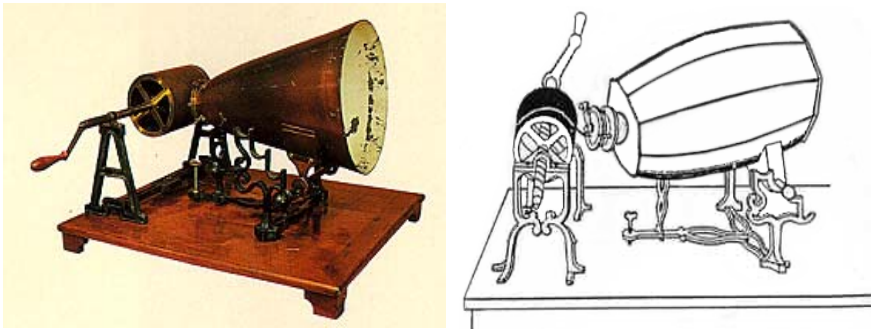

Figure 1-4: The first audio visualization device – a Phonautograph.

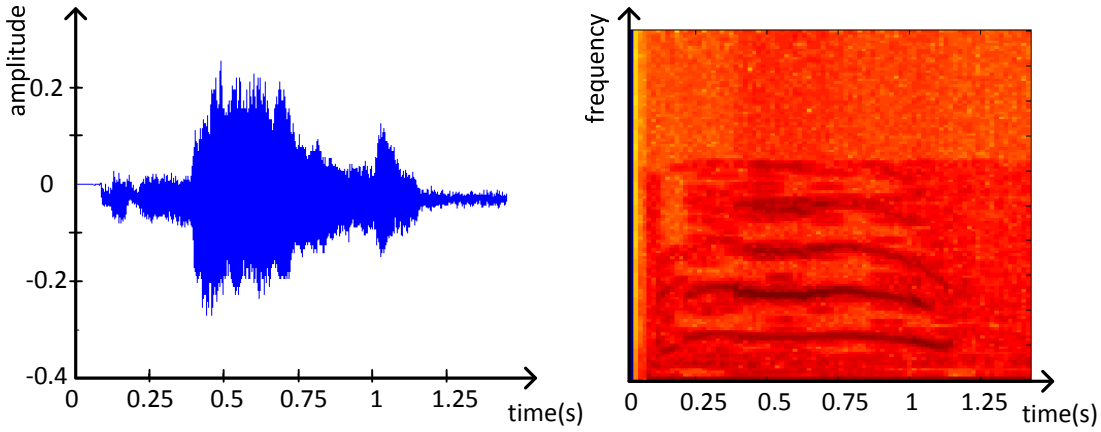

Figure 1-5: An Oscilloscope (L) and a spectrogram (R) for a cat's meow.

The result from an oscilloscope, as shown in Figure 1-5 (L), is a very common representation that expresses the audio signal as amplitude along a time axis. Oscilloscopes are commonly used in modern audio signal processing and can be considered to be the most direct form of visualization. From the oscilloscope, users are able to gain some basic understanding of a sound such as its duration, amplitude, and power (power is positively correlated to amplitude).

Besides representing sound as waves in the time domain, sounds may be visualized and analyzed using spectrograms. The spectrogram represents an audio signal in the frequency domain, as shown in Figure 1-5 (R). The magnitudes of the windowed

discrete-time Fourier transform are shown against the two orthogonal axes of time and frequency. Experts may directly derive information from sound spectrographs such as bandwidth (wide band or narrow band), or even recognize certain words by reading their spectrographs [7].

Oscilloscopes and spectrograms are two widely used methods in audio analysis and signal processing but neither oscilloscope wave-shapes nor the spectrogram of a sound enables non-professional users to understand its content or to perceive its audio features. More accessible methods are needed to visually represent audio inputs.

1.3.2 The Visualization of Audio Properties

In audio processing, a piece of audio can be represented by a vector of audio feature values. Tzanetakis and Cook proposed a method for visually representing audio files by images called *TimbreGrams* [8], in which colour perception and the pattern recognition capabilities of the human visual system are exploited to interpret timbral and temporal information.

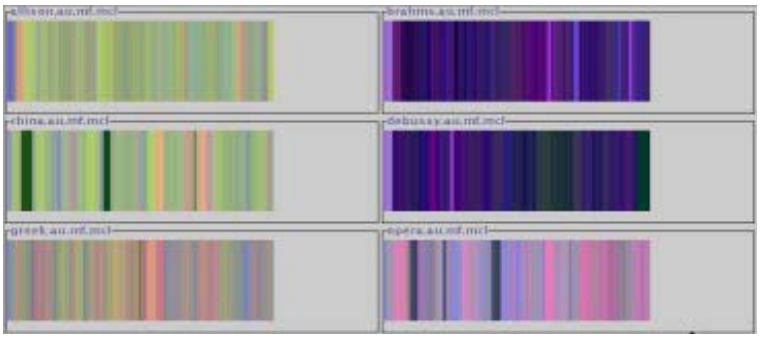

Figure 1-6: TimbreGrams for speech and classical music [8].

In TimbreGram images (Figure 1-6), light and bright colours typically correspond to speech and singing, while purple and blue colours represent classical music segments. Users can easily distinguish speech from classical music by the colours in the images. This approach can be used as an effective tool for speech/music discrimination as the resultant images can illustrate audio properties even to non-professional users. However, this method does not meet our audio visualization requirements because it does not give enough information about the content of an input file.

Similar work has also been explored by Politis, Margounakis and Karatsoris [9]. But in this case the input files are limited to music and the visualized features are limited to a specific feeling they describe as the "chromatic of music", which is sometimes subjective and cannot be used effectively on general audio files[1] or by non-professional users. A chromatic is adopted from colour models and used to describe a user's feeling for a piece of music. Audio properties have also been represented by other visual features in other applications, such as loudness by height of a sphere [10], reverberation by colour [11], and pitch by light intensity [4].

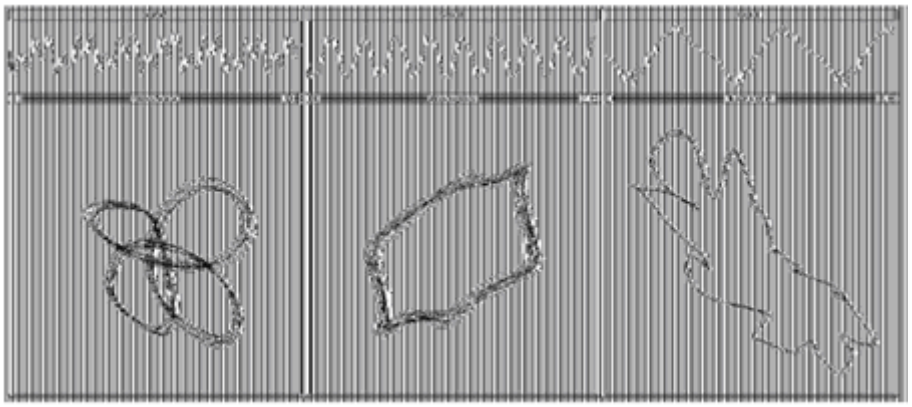

Figure 1-7: Visualization results for single notes on various instruments in phase space [3].

The shape of an object is a visual feature that has been used in some audio visualization approaches that generate black-and-white images [3]. In this algorithm, any sound signal $f(t)$ is transformed into a three-dimensional phase space $\Phi_3(f)=(f(t),f'(t),f''(t))$. Then in a discrete form of this transformation, periodic signals are represented by simple shapes, for example a sinusoid represented by a circle or an ellipse. Natural sounds which are comprised of different sinusoidal signals have more complicated shapes. The images in Figure 1-7 represent the same note played on three different instruments. The sounds produce different shapes in phase space. In addition, the consonance of a chord has been visualized using the roughness of the curve as a visual cue [12].

These approaches were developed for varying purposes and use different visual features to visualize audio features. But none of these systems can visualize both the content and audio features of sounds in a way that is accessible to non-professional users. Users

[1] In this context, the term general audio files defines a limited subset of short duration speech and music sounds as well as environmental and animal sounds.

cannot grasp the content and audio properties by rapidly visually scanning the results. Moreover the visualization of the audio features is only suitable for audio files of the same type because the audio features of different types of sounds are quite different. The comparison of audio features is only meaningful when it is made between sounds of the same type. However, these studies are important because they offer information about auditory-visual associations that could be employed in our audio visualization system.

1.3.3 Auditory-Visual Associations

The perception of auditory or visual features is subjective and can differ from one person to another. However, the study of associations between auditory and visual features can provide information about the most commonly experienced associations and this can be used to develop audio visualization systems.

Giannakis and Smith discovered auditory-visual associations such as pitch-lightness and loudness-colourfulness [13]. Evans also generated heterophonic maps to connect visual features to musical features [14]. Smith and Williams [15] noted connections between audio properties and human feelings. The pitch of tone defines the perceived volume of a sound and in their visualization is represented by the location of a sphere in a three dimensional space. The size of the sphere indicates the level of volume and the colour maps to the timbre of the music. However, it has been found that the relationships between audio and visual features vary from one individual to another and this poses a difficulty for any audio visualization system that is dependent on these relationships. For example, the commonly used visual feature, colour, does not generate the same reaction in all people. Kaya and Epps tested the emotion-colour relationship and found that the responses to the same colour by different people were quite varied [16]. The details of visual-acoustic relationships will be discussed further in chapter 5.

Consistent relationships between audio and visual features are needed in order to represent audio features visually when employing image processing filters. The most commonly accepted audio-visual pairs can be employed in an audio visualization system as long as viewers are made aware of how they are to be interpreted and these interpretations are consistent. However, for our purposes, the visualization of an audio input by the visualization of audio features alone, and without description of its content,

is not sufficient. When this approach is used, users can only view the differences between audio inputs when they are the same type of sounds, because audio properties from different types of sounds are diverse and incomparable.

1.3.4 Visualizing the Structure of an Audio File

Some audio visualization systems aim to represent the structure of sounds, especially music, using images [17]. Audio pieces are parameterized into a pre-designed acoustic feature space and then the similarity or dissimilarity between two audio "instants" are calculated as an element in a matrix and represented by a pixel in the final image. The resultant image for a piece of music is a checkerboard image, intended to show the resemblance among instants of the music input. This method is useful for the analysis of music structure but the images produced using this method do not provide useful visualizations of general audio files.

Foote's visualization of music using self similarity [17] has been extended to structural analysis for indexing and thumb nailing [18] [19]. It is worth exploring the possibility of adapting this technique for use in the audio segmentation stage of our visualization system. It is conceivable that image processing based edge detection and the resultant similarity matrix of an image could be used to detect abrupt changes of audio properties between different sounds in an audio file.

More recently, Bergstrom, Karahalios and Hart introduced a new method for visualizing the structure of music showing consonant intervals between notes and common chords [20]. The results can offer information to experienced musicians about the structures of music but they are hard to interpret for non-professionals.

All of these methods are intended for knowledgeable users and are strictly designed to visualize musical files. As songs often have repeating regions, from the resultant structure image, viewers can find the repeating patterns which are important for music summarization. But they do not help in the understanding of the content beyond the structure.

1.3.5 Music Visualization

In this subsection, "music visualization" refers to a 2D or 3D image that represents music. Non-professional users may first encounter music visualization as a secondary output of audio media player software [21]. These systems typically generate and render animated imagery that is based on certain audio features in real time and is synchronized with the music when it is played.

Music visualization is the most commonly studied topic in audio visualization. Most research in audio visualization handles music input only and could be more accurately described as music visualization.

Some music visualization research has resulted in methods that are used to represent or describe a piece of music [22] [23]. Others have developed ways in which music can be generated to represent given visual features [5] [9] [24]. Because this work concerns ways in which music and visual features can be related it provides some helpful information about visual and acoustic feature relationships.

The audio features that best describe music are not necessarily the best for the description of general audio files. Some audio features of music, such as tempo, are not likely to be relevant to general audio files so the approaches for visualizing music are not suitable for general audio visualization. However, because general audio databases are not limited to certain types of audio files they may contain a considerable number of music files. Therefore any system for the visualization of general audio files must be able to visualize music as well as other sounds.

A tool named MELIRIS was developed to visualize the feelings that musical pieces may generate in the audience [22]. It is a chromatic analysis tool as well as a standard media player. It presents chromatic information in colourful stripes and tracked pixels. MELIRIS also visualizes sound attributes in sonograms, frequency distributions and frequency-amplitude graphs. The system provides a music classification process based on the chromatic index.

Some music visualizations are derived from note-based or score-like representations of music, typically from MIDI note events [17]. The traditional method for visualizing

music is music notation [15] but unfortunately, many people cannot read music scores so Smith and Williams used colour and three-dimensional space to visualize music instead of standard music notation [15]. In addition, Malinowski [25] introduced "The Music Animation Machine" (MAM), that displays the music's structure by using bars of colour to represent the notes in the music.

Music visualization methods have been categorized into two types, augmented score and performance visualization, by Hiraga and Matsuda [23]. The former was intended to assist composers in documenting expressive intentions on a musical score or to assist performers in learning a piece of music [26] [27]. Some were developed to assist musical performances [28] [29]. Hiraga and Watanabe [30] generated a system to illustrate any change in performance using a series of Chernoff faces that may be used in music training or practice.

In some approaches, the result is not limited to the visualization of audio features but also includes their changes. For example, Hiraga and Matsuda visualized tempo change, dynamic changes and the articulation of music pieces with vertical lines, horizontal intervals and the height and width of bars [23]. This kind of research concentrates more on music analysis than on visualization. For example, Politis et al. argue that the song "How you gonna see me now" (Alice Cooper) is most similar to "The trooper" (Iron Maiden) and both belong to the category of Metal songs [9]. They visualize these two songs by using chromatic graphs (Figure 1-8) and use these to compare their similarity.

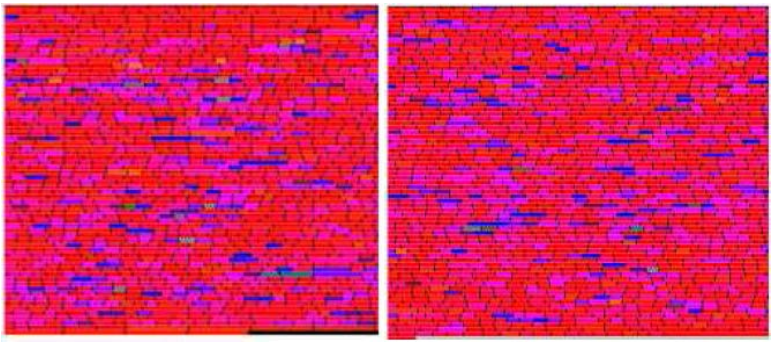

Figure 1-8: Visualization of two songs using chromatic graphs (in terms of colouring) [9].

It is also important to note that the reasons for music visualization vary. For example, music visualization has been used to support music learning [30] [31]. It has also been used to analyze performance. Mood was used instead of content and tags for a musical data mining interface [31]. This system has been tested using several instruments and

provided visualizations that facilitate the analysis of expression, principally in classical music performances.

Rather than visualizing audio inputs with images, some researchers have attempted to use visual characteristics to affect sounds [5]. Lubar argued that some elements in music could be applied to visual arts such as paintings [24]. Besides proposing an algorithm to visualize music according to its melodic structure, Politis et al. invented "chromatic bricks" to create a complete music piece from colours [9]. It is a mapping from colour to melody – an inverse process to the analysis and visualization of music.

1.3.6 Speech Visualization

Speech visualization systems concentrate on content description or position illustration rather than feature illustration.

Hailpern et al. hypothesized that speech visualization techniques can both act as a means to support communication and a method to help autistic children develop speech skills by visualizing vocalization [32] (see Figure 1-9 (L)).

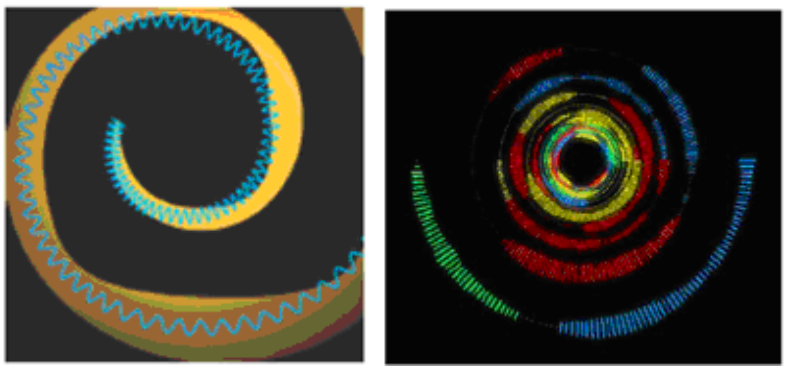

Figure 1-9: (L) Graphic matching in speech learning [32]; (R) Clock visualization for conversation [33].

Karahalios and Bergstrom visualized audio conversation using simple graphical elements to identify the current speaker [33]. Different colours were used to represent different participants around the table and the length of each rectangular slice, which forms a section of the circle, corresponds to the average amplitude of voice (Figure 1-9 (R)). Similar approaches can be found in Donath, Karahalios and Viegas' research into the visualization of conversation [34] and the work of Bergstrom and Karahalios who visualized speakers in conversations around a table using a clock-like image [35].

Simunek [36] developed a system for human face animation driven by phonemes that produces a kind of visualization for speech by animating lip movement. Bregler et al. created a new video of a person mouthing words that he did not speak in the original footage [37]. Bregler's work could be useful when trying to represent an audio file using archetypical video templates. His methods could be adapted to generating new video sequences that accurately represent the time sequence of sounds in a general audio file.

The purpose of speech visualization may be different from the purpose for our audio visualization system but the approaches used in speech visualization and the use of video represent commonalities.

1.3.7 The Visualization of Audio Files using Images

Audio visualization results are most commonly static 2D or 3D images. Visualizations of audio files have also been constructed in three-dimensional space by Smith and Williams [15], Chaudhary and Freed [38], Kaper et al. [11] and Hiraga et al. [31]. Smith and Williams present a method for visualizing the audio properties of MIDI music by using colour and three-dimensional space [15]. The mapping function is defined by the musical characteristics and the piece of music is transformed into three dimensional graphical representations. Tones are represented by coloured spheres and the pitch, and volume and timbre are the audio properties that define the spheres. Kunze and Taube also generated a 3D graphical tool that could be used by composers to write music [39].

The time, amplitude and features (such as frequency, channel number, etc.) of sounds are visualized in 3D in Figure 1-10. Hiraga et al. [31] produce 3D images in which the depth axis stacks multiple channel layers. Three properties, pitch, volume and tempo were represented by the height, diameter and colour saturation of cylinders.

Either 2D or 3D images can be employed to visualize sounds as long as there are enough visual features available to represent the selected audio features. The advantages of 3D over 2D become apparent when more than one sound is visualized at the same time, for example, visualizing all the audio files in an audio database.

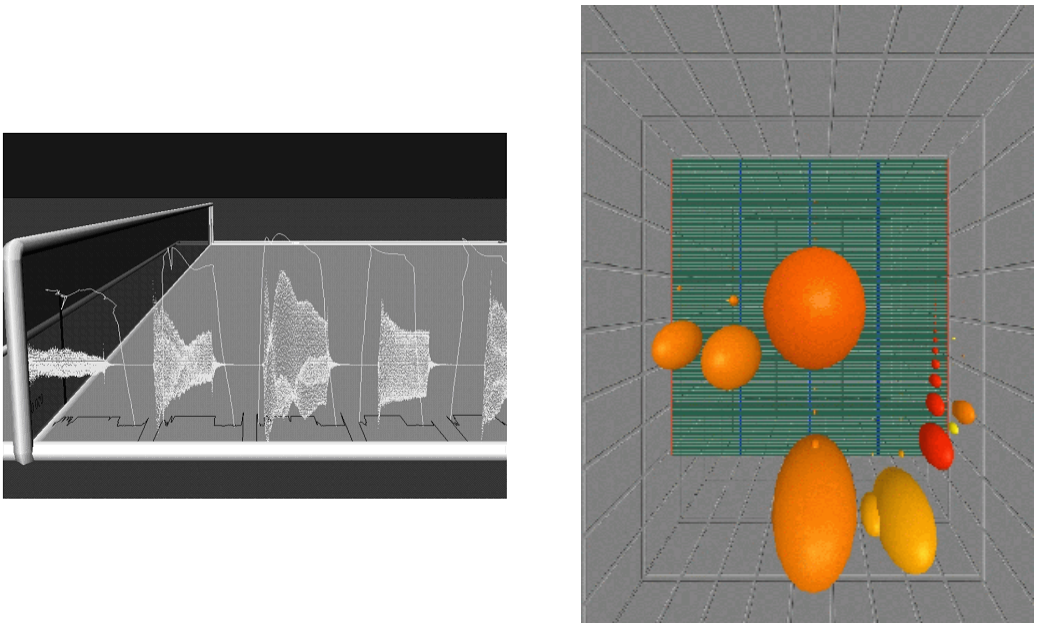

Figure 1-10: Sound visualizations: (L) from method in [38]; (R) from method in [11] .

1.3.8 Audio Database Visualization

The approaches mentioned thus far are for the visualization of individual audio files but some research has also been undertaken regarding the visualization of audio collections. This research has been aimed at the visualization of the distribution of audio files, or the similarities or dissimilarities between audio files. Audio collections have been visualized according to selected or pre-defined audio properties, rather than the content or properties of a specific audio file. This subsection explores existing techniques and solutions for browsing and searching large audio databases, and visualizing audio similarities in audio clusters.

The visualization of an audio database generally involves representing audio file clusters. It is a useful way of displaying the whole structure of an audio database especially when the audio files change frequently or they need to be classified in different ways. For example, a database of songs may have more than one way in which it can be clustered. A view that shows all the clusters may provide a better insight into their properties as well as the opportunity to explore their relationships. Research into audio database visualization has led to the development of automatic methods for clustering audio files and showing their similarities. The visualization of similarities between audio pieces can provide assistance with the exploration of audio collections.

Conventional tools for audio database management are based on directory-based tree structures and lists of textual information [40] [41]. A two-dimensional "landscape" is the prevalent representation. The central idea is to generate a geographic view of audio collections by using the distance between two audio files to roughly represent their differences. It can be employed to visually cluster audio, for example, in the classification of music according to genre.

Pampalk proposed an approach named "Islands of Music" which aims to provide an intuitive interface that can be used to analyze, organize and visualize digital music libraries based on perceived acoustic similarities of music pieces [42]. In this approach, the similar music pieces are clustered together like islands on a map. This can be used for music databases and the island maps can be based on a set of selected features such as instruments used, melody or rhythm of music. A geographical map indicates the relationships between music files. A genre pre-classification is not required in the "Islands of Music" because the system, based on psychoacoustic models, can estimate the acoustic similarities of any given music in raw audio format to other audio files in the database [43]. The left image in Figure 1-11 is a geographic map generated to represent the similarities between audio files.

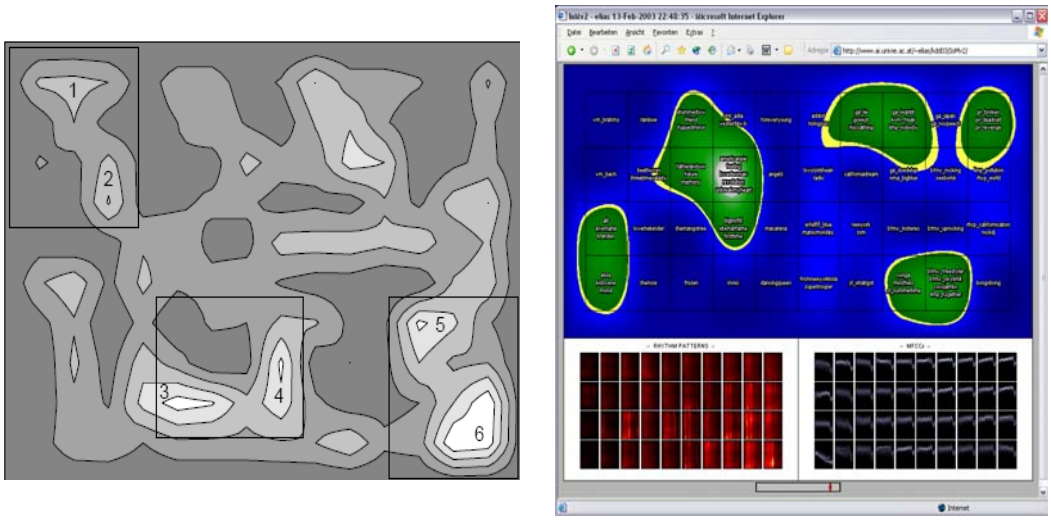

Figure 1-11: (L) Music collection visualization result by "Islands of Music" [43];
(R) Content-based exploration of music archives [44].

The concept of Islands of Music has been extended in [41], [44], [45] and [46]. Pampalk, Goebl and Widmer extended the Islands of Music to the visualization of changes of the cluster structure [44]. The resultant map illustrates how modification of the audio features with respect to weighting or normalization changes the cluster

structures (Figure 1-11 (R)). Schedl visualized a very large audio database employing hierarchical components [45]. An interface was also proposed by Zhu and Lu to browse, search and navigate personal digital music collections [41].

Lampropoulos and Tsihrintzis [47] visualized the organization of music files with a dendrogram. Leaves of the tree represent music files and the height illustrates the similarities between two songs (see Figure 1-12 and Figure 1-13). This is an efficient way of visualizing the differences between audio files within a class but if a database contains multiple classes these visualizations provide no information about the contents of audio files.

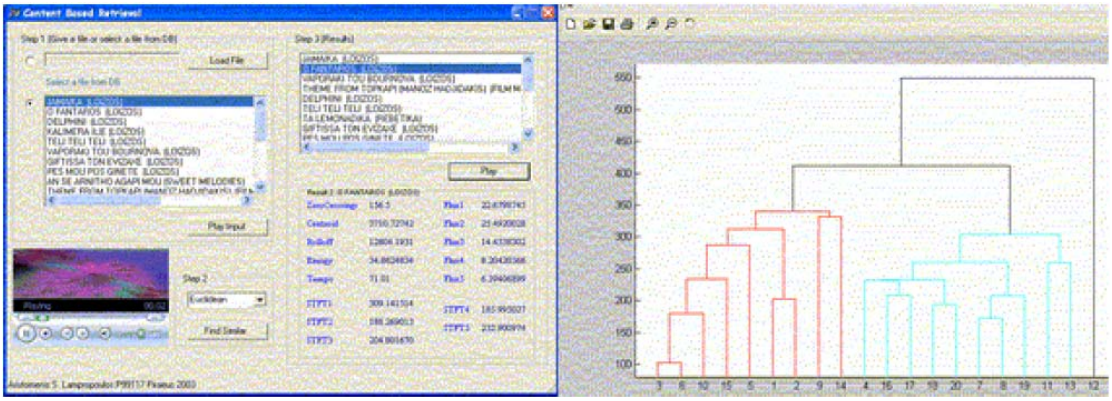

Figure 1-12: Resultant visualization of audio file query results from system in [47].

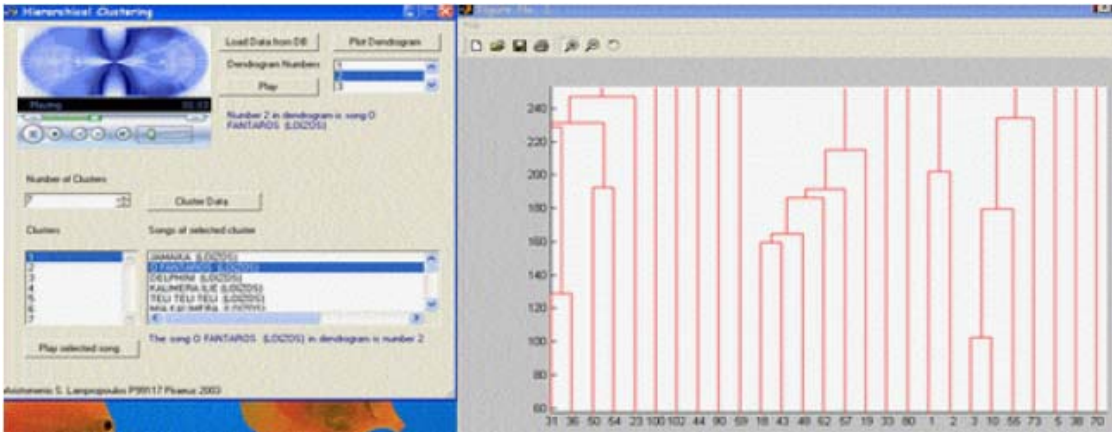

Figure 1-13: Browser of selected audio files to show their distances (differences) in [47].

Sounds are represented visually by Brazil and Fernström by *sonic objects*, which are objects used to represent an audio file, with specific properties in the Sonic Browser [48]. Users can define a position on a grid from which all sounds fall into a surrounding region. Figure 1-14 shows the visualization from the Sonic Browser. The left image in Figure 1-15 visualizes the genres in the audio database.

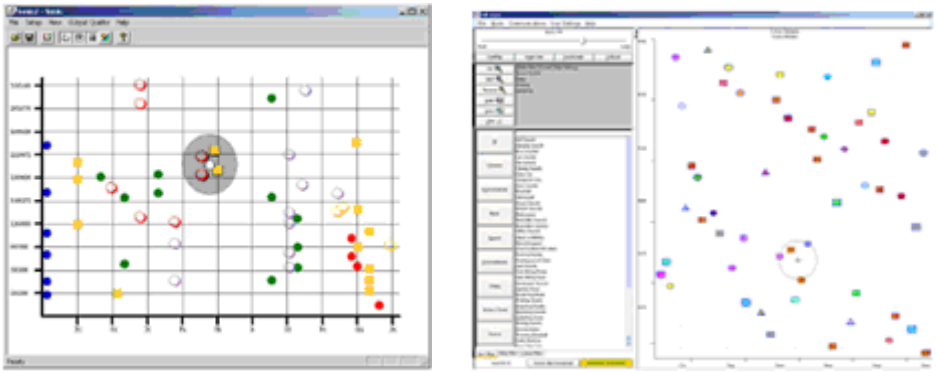

Figure 1-14: Visualizing audio database using the Sonic Browser [48].

Tzanetakis et al. developed an implement for visualizing sound collections to show the relative similarity within genres (the right image in Figure 1-15) [49]. Bainbridge et al. proposed a visualization for digital music libraries [50]. Koutsoudis et al. [51] proposed incorporating signal processing techniques and using psychoacoustic models for classification purposes to describe music.

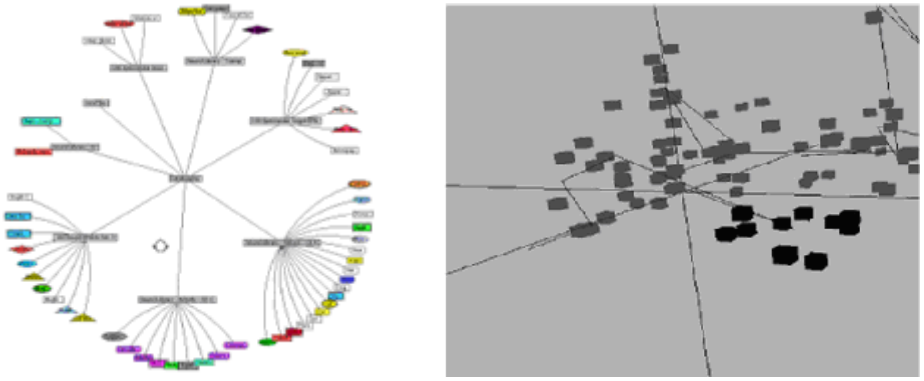

Figure 1-15: Audio database structure visualization results (L: in [48]; R: in [49]).

The above mentioned approaches can be used to cluster audio files with certain similar characteristics and to visually illustrate their distributions in a database. But non-musicians do not benefit from such visualization results.

1.4 Summary

A more general method is needed because the existing research related to audio visualization provides no approach that can be used to visualize the content of audio input as well as its audio properties. This thesis presents a general method for audio

visualization which is more accessible than the existing methods because the filtered images used in the application are directly representative of sounds in the real world.

Chapter 2
The Audio Visualization System Framework

This chapter discusses the methodology applied in this research. Then the framework of the proposed audio visualization system is presented. Modules in the system are introduced and the process of visualizing an audio file is explained. Finally there is a brief discussion of the development environment used.

2.1 Development Methodology

This research is founded in empirical, experimental based research. Specific experimental design is addressed in the relevant sections of the thesis. The proposed research involves developing methods, determining dependant and independent variables then experimentally evaluating these factors. Because of the iterative nature of the experimental process we adopted a traditional iterative software development approach, namely the Spiral model [52], when developing the system's segmentation and classification modules. We approached these experimental phases as a continual refinement of the algorithms and methods developed in order to optimize the performance and accuracy of the methods. In the case of the segmentation and classification experiments descriptive and comparative statistics were employed to inform the next cycle of experiments. The development of the visualizations was also an iterative experimental process where the resultant visualization was used as a subjective guide to further improvements. The pre and post module development phases were informed by the systems development research method described by Nunamaker et al. [53]. We have adapted their key phases as follows:

- *Phase 1* requires the construction of a conceptual framework for the system and represents a process that positions the research. The challenges, objectives and research questions are identified. As part of this process an in-depth literature review is undertaken.

- *Phase 2*: involves developing a unique architecture, defining the system modules and functionality, the components and their interrelationships.
- *Phase 3*: implements and tests the individual modules of the system. It is in this phase that the spiral model is utilized to guide development.
- *Phase 4*: requires the integration, fine-tuning and evaluation of the combined modules.
- *Phase 5*: documents the processes and findings of the research.

2.2 The Audio Visualization System Framework

Figure 2-1 shows the framework for this novel audio visualization system, which has a pre-built audio database and a visualization generation component to process audio input. A well-selected database should be able to accommodate any potential classes of sounds to ensure the accuracy of the classification and visualization processes.

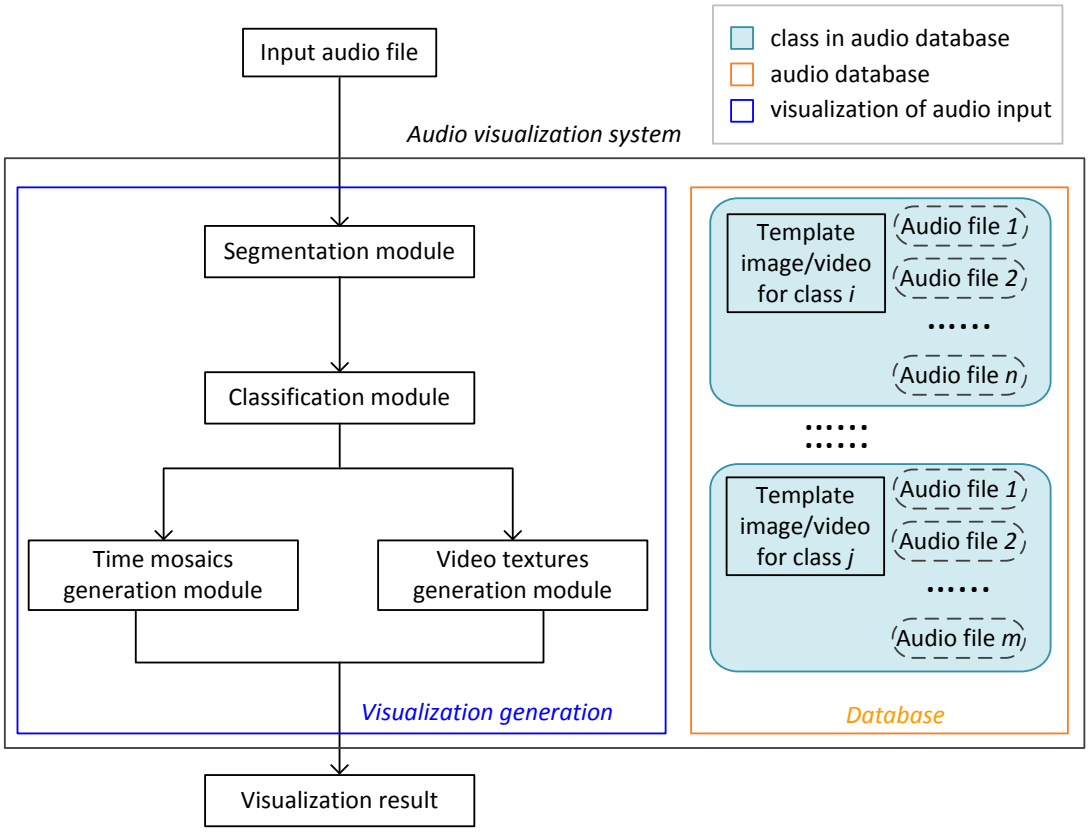

Figure 2-1: Framework of audio visualization system.

The system has four modules: a segmentation module, a classification module, a time mosaics generation module and a video textures generation module. An audio database

is employed as the training set for audio classification and visualization. In this audio database the audio files have been classified manually. Each class has an archetypical template image and template video, that are used to represent that class of audio files. The visualization generation algorithm takes the audio input and generates images or video texture to represent it.

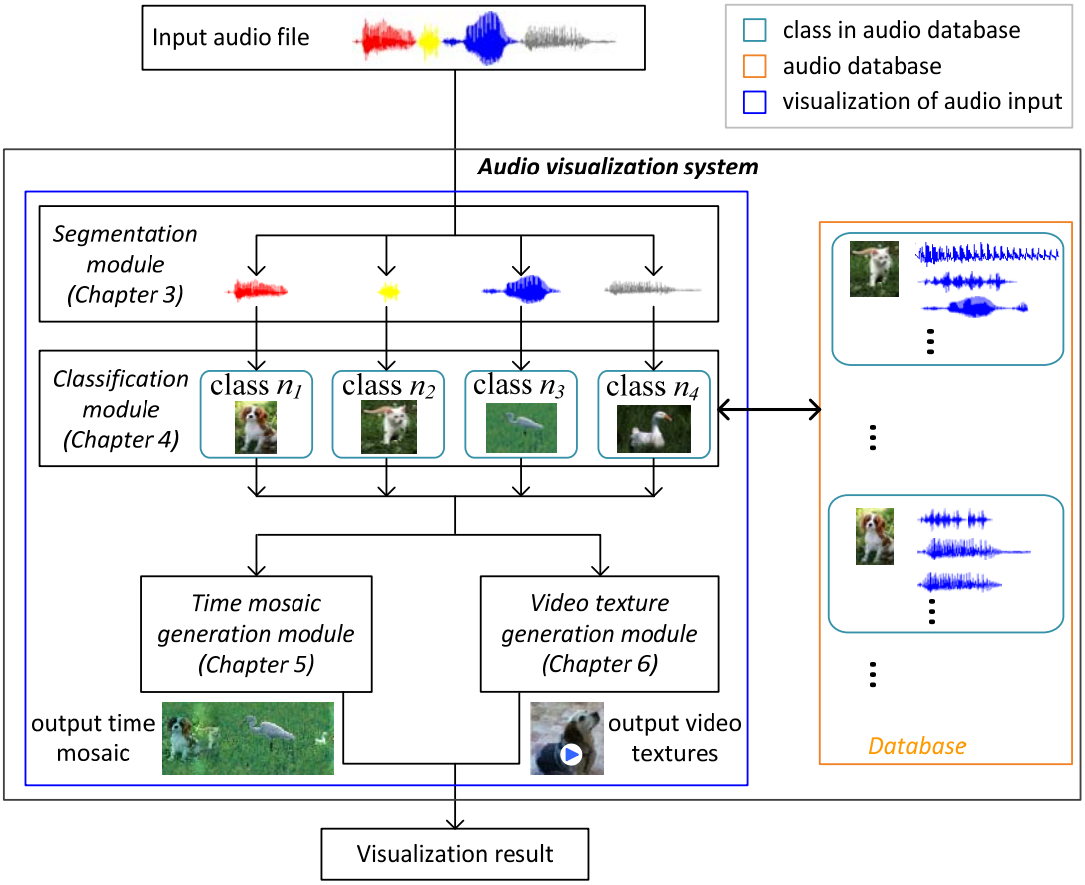

Figure 2-2: Processes of visualizing an audio input.

Figure 2-2 shows a detailed view of the framework. An audio file, which may contain a single sound or multiple sounds, is introduced as the file to be visualized. The segmentation module separates the input audio into component sounds. Then each audio segment is classified. The classification process selects the template image or video for each audio clip. A distinctive seamless sequence of images based on the computed audio features of each segment and the audio file is generated by the time mosaic module. In a similar way the video texture module generates a period of video texture.

2.3 The Development Environment

To implement the system, three propriety software systems were used. SQL Server was the database management system employed to store the audio files, each audio file's metadata and the template images and videos for each class of audio file. The user interface to the audio visualization system was developed using Java. MATLAB was employed to process the audio files, images and videos. MATLAB was chosen due to the built-in image filters available.

Chapter 3
Audio Segmentation

The goal of this research was to investigate and develop an accurate and robust, unsupervised segmentation method for partitioning a heterogeneous audio input stream into its composite segments, known as audio clips. The implementation of this segmentation method is the first module of the audio visualization system. The module separates an input audio stream into individual sounds (sound clips) and indicates the silence periods between the sound clips (which are not involved in the subsequent classification step).

Existing audio segmentation methods are reviewed and their limitations discussed. The existing methods are not suitable for the proposed application and therefore a novel two-phase segmentation method is presented. The experiments with, and analysis of, this method are detailed. The two-phase segmentation method offers accurate results comparable with other high performance segmentation algorithms and provides a general scheme that is appropriate for the audio visualization system.

3.1 The Audio Segmentation Module

Segmentation plays an important role in audio processing applications such as content-based audio retrieval and recognition, audio classification, and audio database management. For general mixed-sound audio files, segmentation is an especially important step because it is much easier and more accurate to undertake further analysis, such as query or classification, using segmented homogeneous audio clips than using a raw audio input stream.

Sounds that are considered homogeneous are sounds that are acoustically similar. Generally, the characteristics of a single sound tend to be constant over time. The characteristics of sounds in the same class tend to be acoustically similar as well. The

simplest boundary between two neighbouring sounds is an abrupt change, or transition, in acoustic features which occurs in a single frame or between two adjacent frames. If there are different sounds in an audio stream, they can therefore be separated by locating abrupt changes in audio features between two adjacent frames. Gradual transitions across sound segments are more complex and therefore more difficult to determine.

Audio segmentation identifies appropriate boundaries for different sounds in a single audio stream and partitions these continuous streams into homogeneous regions. The adjoining sounds are considered homogeneous if they are acoustically similar and could be classified as belonging to the same class in the ontology. After segmentation each resulting audio clip or segment contains only one sound, or acoustically similar sounds of the same class, that is acoustically different from other sounds in the audio stream. Where the audio is entirely homogeneous, the resulting output after segmentation is the same as the input. The output of a heterogeneous audio file is a series of audio clips where each contains one or more sounds of a particular class.

As the first stage of the audio visualization system, the segmentation process tries to detect the boundaries of each audio clip within the input audio stream, i.e. the frames where the acoustic characteristics change (Figure 3-1). The segmentation module output consists of the number of audio clips and each clip's start and end position relative to the start of the input audio file. Subsequently, the duration and variation of each audio clip can be easily extracted. From the perspective of the audio visualization system, the output of this module provides the input for the classification module, the second part of the system.

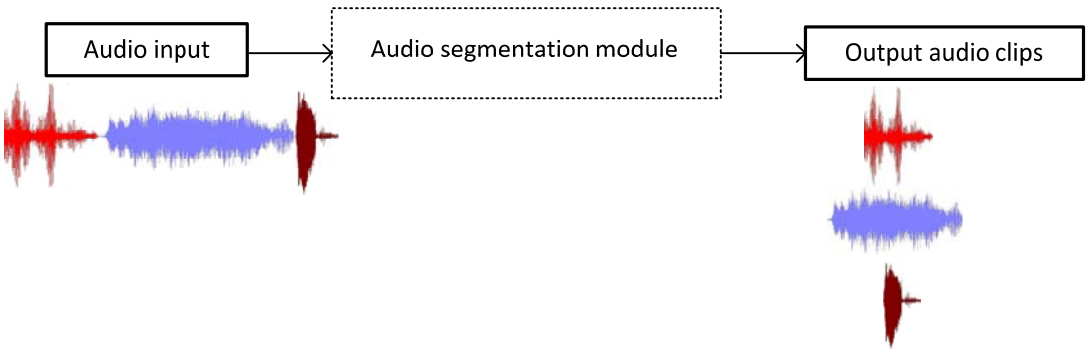

Figure 3-1: An overview of the segmentation process.

The boundary detection of two sounds that overlap is purpose-dependant and challenging. A discussion of audio segmentation with overlap can be found in a number of references [54] [55]. Boakye segmented speech acquired from interpersonal meetings into three categories: single-speaker speech, overlapping speech, and non-speech [54]. Overlapping alters the audio features even if the original sounds are of the same type (belonging to the same class). In audio segmentation, the overlapping period between two sounds is ambiguous because whether a sound clip is the overlapping of two sounds or a homogeneous single sound is determined by the task specific semantics. For example, noisy speech can be considered as a kind of overlapping if the segmentation aims to separate speech, noise, and silence. Another example is the processing of musical audio to identify and separate musical instruments. Samples containing multiple instruments are of the overlapping type though they are treated as a type of music in many applications. For sound visualization, the signal of two overlapping sounds is different from either of the original two sounds and should be treated as a new sound snippet. The proposed segmentation module acts as a pre-process for further classification and visualization of a single sound. Wherever overlapping regions exist the goal is to identify the overlapping areas between the neighbouring sounds and to separate the overlapping areas from the surrounding sounds.

The requirements of the segmentation method in the audio visualization system are:

Robustness: The database contains unrestricted and general classes of sounds so it is desirable to be able to segment most types of audio and not be limited to just the broadly defined classes such as speech/music/sounds/silence.

Accuracy: As the first module of the audio visualization system, all further analysis of the audio depends on the output of this module. Therefore the accuracy of the segmentation is fundamental and crucial to the following modules in the audio visualization system. Inaccurately detected segments would lead to incorrect classification results and give spurious visualization results.

Flexibility and practicality: The audio visualization system uses an existing database to visualize any new input audio. The database cannot cover all possible types of input audio therefore, as the first step to process the input audio, this module should be able to function unsupervised and to automatically accommodate new audio types.

The novel two-phase segmentation method introduced herein not only meets the requirements of the audio visualization system, but is competitively accurate for general audio segmentation. It is also worth noting that this two-phase method, as an accurate segmentation technique, can be used in other applications as well. Potential applications include speech and music segmentation, speaker change detection, speaker clustering, musical structure extraction and music summarization.

3.2 Literature Review

Several methods have been developed to segment audio streams in different applications. Some techniques have been designed for, and limited to, audio samples that contain only certain types of sounds, while others have been used for audio samples containing heterogeneous sound types.

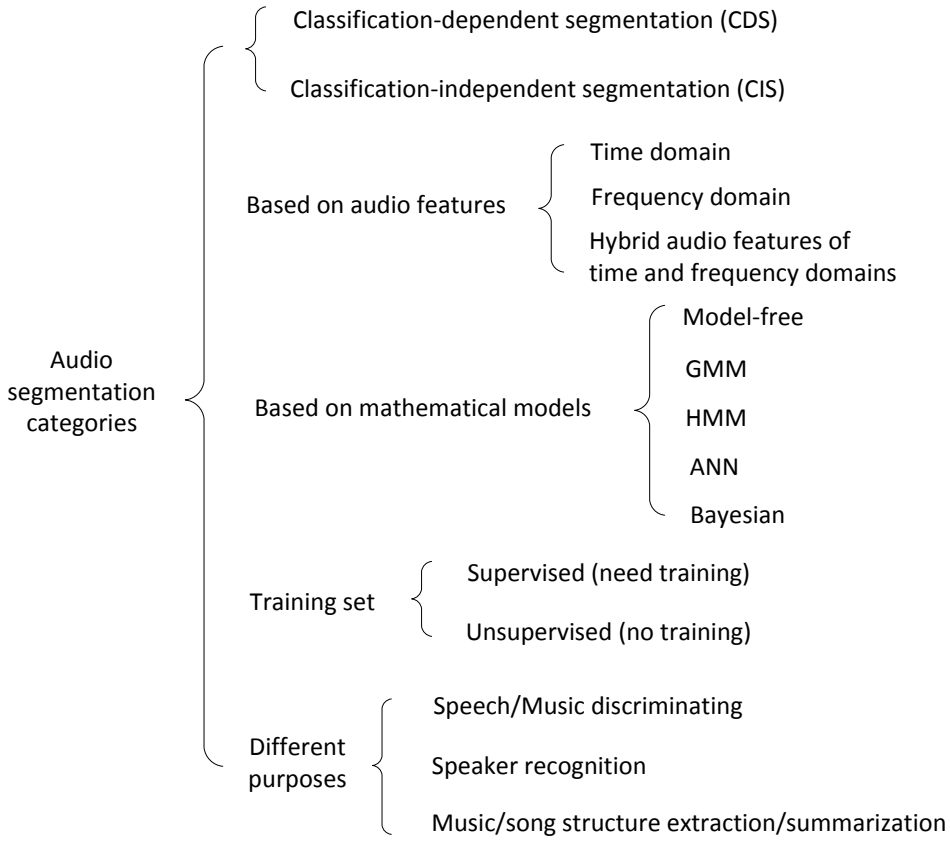

Figure 3-2: Categories for existing segmentation methods.

Figure 3-2 illustrates some possible ways to group reported segmentation methods. Segmentation approaches belong to either classification-dependent segmentation (CDS) or classification-independent segmentation (CIS). They can also be grouped depending

on which audio features they use to view and interact with the input stream, such as time-domain, frequency-domain or a hybrid of both. Yet another criterion is whether the approach requires a training set to learn from prior audio segmentation results (supervised approach) or not (unsupervised approach). Segmentation methods may also be defined as model-based or model-free. These categories will guide our discussion of prior work.

Chen et al. [56] identify two types of segmentation approaches: classification-dependent segmentation (CDS) and classification-independent segmentation (CIS). CDS methods consider that the property-vector of each frame (a short duration signal) in an audio clip independently belongs to a specific class. If all frames of a sound piece belong to the same class, the sound piece belongs to the class. So in CDS methods such as in Lu et al. [57] and Hain et al. [58], classification is carried out before segmentation. Segmentation can therefore be regarded as a post-analysis (classification) step. Hence the accuracy of the partitioning obtained is highly dependent on the accuracy of classification. Whichever algorithm is employed, misclassification cannot be totally eliminated because the spread of the acoustic properties across different classes contained in the audio stream makes it difficult to find distinct boundaries between audio classes. Frame-based classification increases the risk of misclassification because different types of sounds have different structures. Even if a homogeneous sound belongs to a class, its individual frames do not all necessarily belong to the same class. Huang and Hansen [59] argued that pre-classification tends to cause the insertion of short duration segments, resulting in a "toggling" action between potential classes. Lu et al. [57] claimed the post-smoothing process can improve the segmentation accuracy for audio segmentation, but it is based on the continuity of an audio stream in a video program, which does not suit all general purpose audio segmentation methods. CDS methods can be problematic because it is difficult to control their performance [56]. Unlike methods which rely on a prior classification, CIS approaches detect changing points in input audio by self-similarity. For a general audio visualization system, the CIS method is more suitable than CDS if classification is undertaken prior to segmentation. This is due to the misclassification of individual frames observed when the CDS is used. Consequently CDS is more likely to result in incorrect template image selection and therefore incorrect visualization.

For audio segmentation, there is no absolute measurement for determining which audio feature set is better than others. Audio feature set selection depends on the purpose of the segmentation process and the type of audio file being segmented. Panagiotakis and Tziritas [60] employed a typical time-domain approach that used the root-mean-square (RMS) and zero crossing rate (ZCR) to discriminate speech from music. Tzanetakis and Cook [61] presented a general methodology for temporal segmentation based on multiple features. Saunders [62] used the energy contour and the zero-crossing rate features to discriminate between the speech and music segments in an audio stream. Scheirer and Slaney [63] used a set of 13 audio features to discriminate between speech and music segments. Eight of the 13 audio features were based on power such as 4 Hz modulation energy and spectral centroid. However, these power-related features are not very suitable for short-duration sounds. Cepstral coefficients were used by Moreno and Rifkin [64] and Seck et al. [65] to classify and segment speech and music. The reported accuracies of these audio features provide some inspiration for general segmentation in developing the present audio visualization system.

Different mathematical models have been employed in model-based segmentation approaches, such as Gaussian Mixture Models (GMM) [63] [66] [67], Hidden Markov Models (HMM) [68] [69] [70], Bayesian Methods [71] [72] [73] and Artificial Neural Networks (ANN) [74]. Some classifiers have also been used for segmentation such as support vector machines (SVMs) [75] and K-nearest neighbour clustering schemes [76]. For instance, a two level process using a K-Means classifier combined with Hidden Markov Models, which has been applied to football audio broadcast tracks with three result classes whistle/crowd/speaker, has been proposed in Lefèvre et al. [76].

In contrast to the model-based methods, there are also model-free or metric-based audio segmentation methods [73]. The model-free methods always depend on the self-similarity of the input audio [77] [78] [19] [79] [80] [81]. Tzanetakis and Cook [61] proposed a general framework for audio segmentation and classification. The segmentation was based on change-detection of audio textures which was generated from the Mahalonobis distance. From their experiments on a large data set (2 hours of audio data including speech, music and mixed audio), the accuracy achieved was approximately 90%. Foote [17] processed the self-similarity matrix with a checker-board kernel, which calculates the cross-similarity values between regions. Subsequent work by Foote [82] detected the heterogeneous regions with a checker-board kernel

with Gaussian taper. Audio segmentation methods based on a similarity matrix have been employed to process news-broadcasting [83], which is relatively acoustically dissimilar, and to extract music structures or music summarizations. The accuracy evaluation of these methods was undertaken with specific input audio types and has not been reported for use with audio files in a non-music/non-speech database. Moreover, all these methods have the common drawback that there is no kernel that suits all audio files.

The main difference between a supervised and an unsupervised segmentation method is that the supervised method needs a training set, whereas the unsupervised approach derives its model from the input signal itself. The unsupervised approach tests the self-likelihood ratio within the input audio itself to detect and observe possible changes. Examples of unsupervised audio segmentation approaches can be found in several papers [70] [71] [84]. On the other hand, the systems developed by Spina and Zue [66] and Ramabhadran et al. [68] must be trained before segmentation. It is difficult to compare the performance of supervised vs. unsupervised methods in the literature as no common bench-mark dataset has been used however both methods have reported relatively high accuracies.

As audio segmentation can be used in various fields, the existing segmentation methods can be classified according to their purpose such as "speech/music discriminating" [85] [86], "speaker identification" (speaker change detection) [59] [72] [87] and "music/song structure extraction/summarization" [88]. Kimber and Wilcox [89] classified audio recordings into speech, silence, laughter, and non-speech sounds, in order to segment discussion recordings from meetings. Peeters [90] solved speech/music segmentation and music genre recognition with a system for audio indexing. The system includes feature extraction, feature selection, feature space transformations, and statistical modelling. Its purpose is radio stream segmentation (speech/music) but its performance deteriorates when its audio input is mixed with other categories of sounds. Homayoon et al. presented a distance measurement for evaluating the closeness of two sets of distributions in speaker recognition [91]. A method named DISTBIC has been proposed to segment audio based on speakers [92]. The method detects the most likely changes of speaker, then validates or discards them.

A significant amount of work has also been conducted in the music field. Some systems concentrate on finding repeating patterns, ignoring sections that occur only once, and analyzing structures in music and songs [81]. Some try to attach semantically meaningful labels to the sections found, whereas others concentrate on locating correct boundaries. Ong [93] separated the music structure discovery into two processes: pre-analysis segmentation and post-analysis segmentation. Pre-analysis segmentation (sometimes called frame segmentation) takes place before the content analysis process and performs crucial preparations for the content analysis description. Bartsch and Wakefield [77] presented a method that produced short representative samples ("audio thumbnails") from selections of popular music by identifying the chorus or refrain of the song. Ganapathiraju, et al. [94] pointed to the importance of segmentation boundaries for summarization.

Although there are many approaches to audio segmentation, they are focused on a narrow type of audio such as speech, or music, or noise, or silence separation. For any input audio, the final outcome from different applications may be different owing to these distinct purposes. The selection of a segmentation method always depends on the specific requirements of the application. Existing segmentation methods work well for specific applications but are not general enough for databases containing broad types of sounds from different fields. Because of their limitations, none of the existing methods can give satisfactory segmentation results for general audio. Finding a method to segment general audio files containing different types of sounds remains an open and significant problem.

An accurate automatic segmentation method that is capable of accommodating a variety of sounds is needed for audio visualization. The next section introduces a two-phase unsupervised model-free segmentation method that works for general audio files. This novel two-phase segmentation method is flexible enough to handle much broader classes through self-learning and self-adjustment.

3.3 A Novel Two-Phase Audio Segmentation Method for General Audio Files

In order to study an audio file, it must be split into a sequence of snippets (frames) from which feature vectors are extracted using acoustic features. An audio feature vector characterizes its corresponding audio frame. Any further processing is based on these audio feature vectors. The segmentation module looks for the difference between the audio vectors of frames within a given input stream.

This section presents a two-phase CIS method for general audio files. It can detect audio clip boundaries from the information contained within the audio file itself. Its implementation and audio feature selection abilities are discussed in each of the corresponding procedures within the method.

3.3.1 A Framework for the Two-Phase Audio Segmentation Method

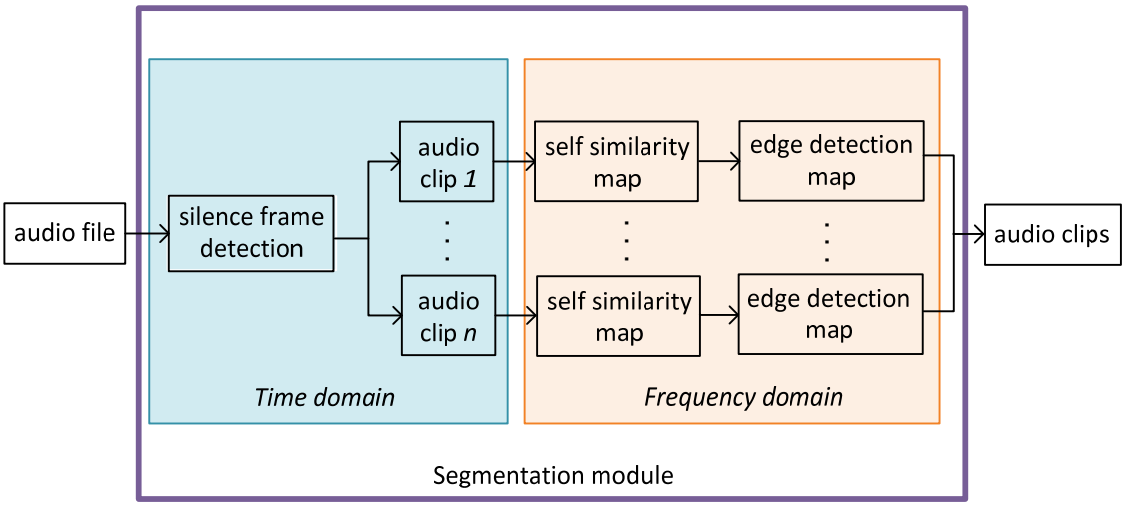

Figure 3-3: Framework for the novel two-phase audio segmentation method.

Figure 3-3 is a structural view of the two-phase audio segmentation method. The first phase exploits time-domain features to identify silence periods and roughly separates the audio. After the initial segmentation, further segmentation in the frequency domain, phase two, is performed. This second phase of the segmentation process takes advantage of the fact that sounds tend to be homogeneous in terms of their audio features. Therefore it can be assumed that any abrupt change in the audio features indicates the start of a new audio clip. This more sophisticated approach is performed on segments

from the initial segmentation; it detects subtle changes in more complex acoustic features until each audio clip contains a single sound. These stages, depicted in Figure 3-3, will each be discussed in further detail.

3.3.2 Phase One – Silence Detection

General audio files may have a short period of silence between sounds where the sound clips do not overlap. Therefore silence detection can often be used to separate sounds. The first phase of the two-phase segmentation finds the start and end of audio clips based on silence periods. Root Mean Square (RMS) [60] and Spectral Power (SP) [95] are widely used audio features for silence detection. If the selected audio feature of a frame is less than a pre-determined threshold, it is regarded as a silence frame [95]. But neither RMS nor SP is suitable when the sound's duration is very short, say less than 10ms, e.g. hand claps as shown in Figure 3-4.

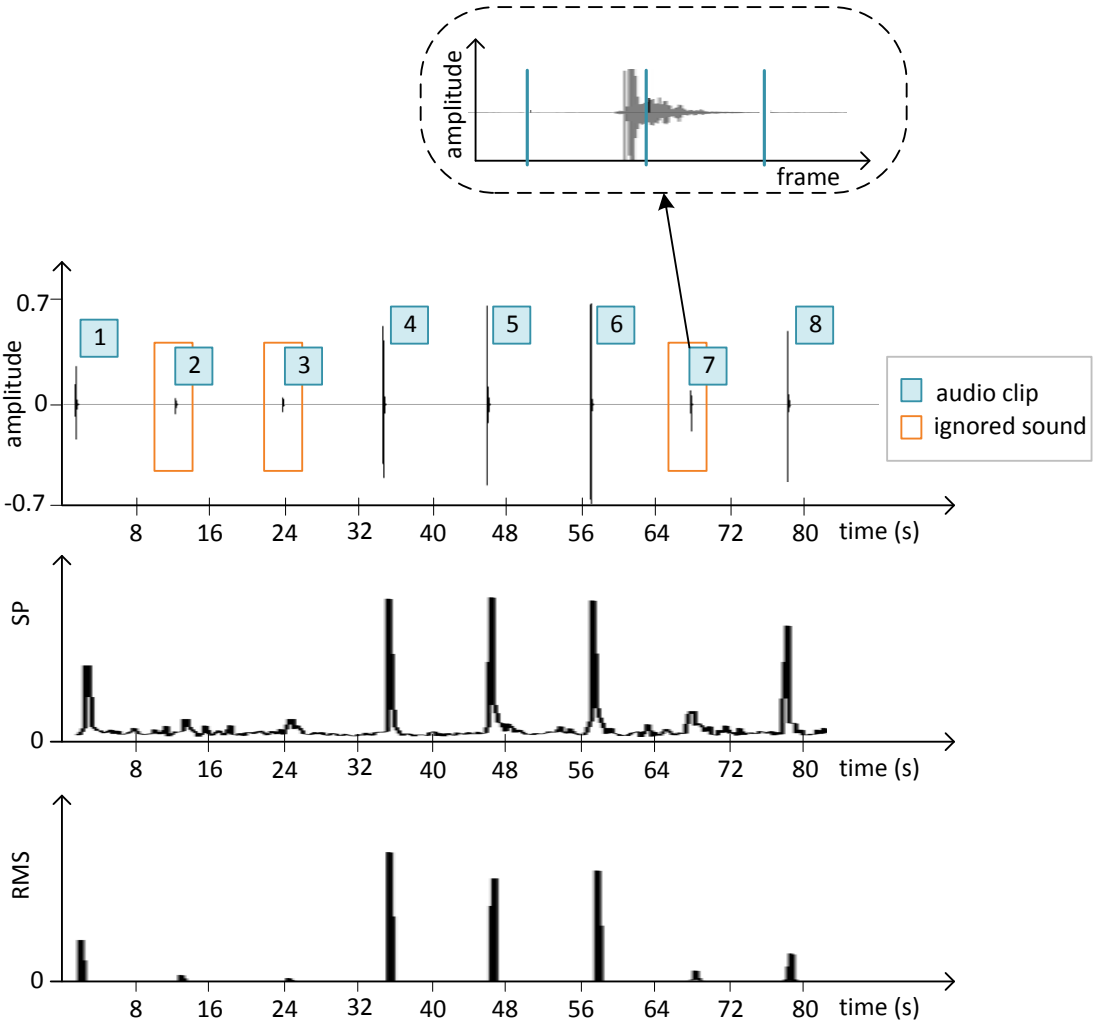

Figure 3-4: Wave shape of audio file "Hands clapping" and its SP, RMS; (Inset) Enlarged wave shape of sound 7 in "Hands clapping".

The signal in Figure 3-4 contains eight hand claps, each lasting ~11ms. Using both RMS and SP, only five are detected. The remaining three clearly audible sounds, visually identifiable in the time-amplitude signal, are not detected. When sound one and sound seven are compared (the top graph in Figure 3-4) the SP values of the two sounds are similar if marking their boundaries manually. However sound seven is not identified as a signal in the input audio. This is because when the input audio is split into frames sound seven is split into two frames (Figure 3-4 (Inset)) which contain short limited sample points, resulting in two frames being detected as silence. Neither RMS nor SP work for short duration signals such as this hand-clapping example.

In order to cope with very short sounds, a new approach is proposed. The maximum amplitude value of each frame is used as a parameter to label silence and signal frames with an empirical threshold T. The threshold T adapts to each audio file thus:

$$T = \begin{cases} 0.015 \times \max(A), \; if \; mean(A) > 0.075 \times \max(A) \\ 0.05 \times \max(A), \; if \; mean(A) \leq 0.075 \times \max(A) \end{cases} \quad \text{(E 3-1)}$$

where A is the absolute amplitude of the signal. Adapting the threshold using the mean value of the absolute amplitudes of an input sound is necessary to minimize the effects of noise within an audio file with variable SP. This results in higher thresholds which eliminate most of the noise in a signal but may also result in the removal of the edges of a fading signal. This tradeoff between loss of signal edges and elimination of noise is necessary to achieve good segmentation.

The given audio is separated into frames, each lasting 2.5ms. Such a short frame size is not appropriate for accurate audio feature extraction. But in this two-phase segmentation method audio features are not required in this phase, and the 2.5ms frame leads to accurate silence detection for signal boundaries using the maximum amplitude.

Once all the frames have been marked, the start and end points for each audio clip are detected. Given frame f_i if f_{i-1} and f_{i-2} are silent then f_i is a clip start frame f_s and if f_{i+1} and f_{i+2} are silent then f_i is a clip end frame f_e. The clip is then defined as $C \in \{f_s, f_{s+1},...f_e\}$. If the silence between two audio clips C_i and C_{i+1} is short (less than 0.01 × *input_audio_duration* for general sound) these two audio clips are combined. If an

audio clip C_i is very short (less than 5ms in duration), it is barely audible and is therefore discarded as noise and does not need to be considered in further analysis. If a resulting sound clip lasts more than 5ms but less than 16ms, the frame in front of it and the frame after it are added to it to form a single clip. The result of the first phase is a set of sound clips (signal frames).

Different systems may have different requirements even when using the same segmentation method. The considerations discussed here are for a segmentation method that is used as a precursor to the classification and eventually to the audio visualization of general audio files. It is important to consider the number of different sounds present in an audio file. For the audio visualization system, it has been limited to less than one hundred. This is because the space on a computer screen is limited and as a result the visualization output cannot contain too many components. This restriction can be fine-tuned to fit other applications. For instance, for long-lasting input audio files which may have hundreds of short sounds, the duration restriction for combining two neighboring audio clips should be decreased correspondingly. It is of little concern if two different audio clips are combined incorrectly, because they are likely to be separated again in the second phase of this segmentation method.

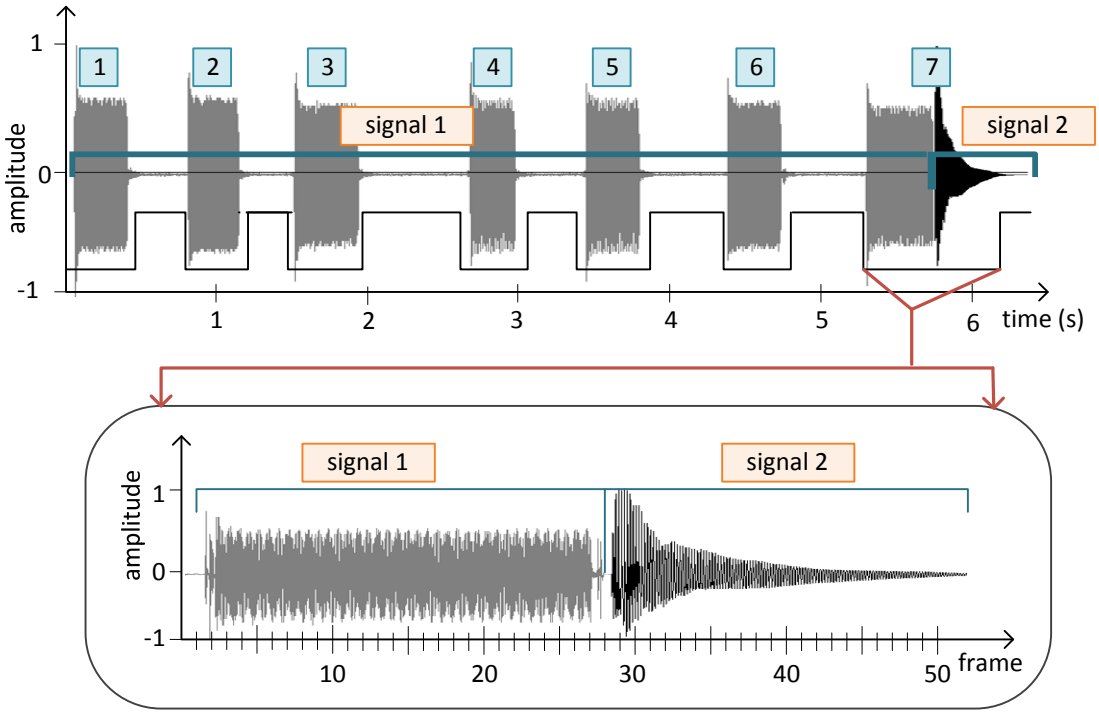

Figure 3-5: Correct audio segmentation result for a mixed sound audio file.

Figure 3-5 gives an example of segmentation results based on silence detection. There are two signals in the input audio. The first is a telephone's touch-tones, (marked in grey) and the second is a violin (marked in black). Using this adaptive threshold based on amplitude the silence periods between the telephone touch tones in signal 1 are correctly detected. However the silence period between signal 1 and signal 2 is not of a sufficient duration to be detected using silence detection.

3.3.3 Phase Two – Edge Detection within Self-similarity Maps

For any audio clips detected in the first phase, if two sounds are seamlessly connected with no or very short silence between them, such as audio clip seven in the telephone/violin audio example, silence detection does not work. When the presence of silence cannot be totally relied on, a subsequent, more sophisticated refinement phase is required for the separation of contiguous clips. Because a sound's audio features tend to be homogeneous, any abrupt changes in audio features may indicate the start of a new sound clip. Foote [17] was the first to use a two-dimensional self-similarity matrix (autocorrelation matrix) where a song's frames are matched against themselves. In a similar vein, Cooper and Foote [19] used a two-dimensional similarity matrix for music summarization. These methods establish that the self-similarity map can indicate the changes within audio clips. Foote's similarity matrix [17] is adapted as the second phase of the two-phase audio segmentation procedure. When similarities between each pair of frames are extracted, a self-similarity matrix is generated for the input audio segment.

This self-similarity matrix can be represented by a greyscale image (See Figure 3-6 Top (L)). Each element in the self-similarity matrix is mapped to a pixel in the image and its value is linearly calculated. Because the matrix is scaled to a grey image, the maximum of the self-similarity matrix is set to one and the minimum value is zero. In this thesis, the term "self-similarity map" refers to the greyscale image of the self-similarity matrix. If there is only a single sound in the audio clip, the self-similarity map is homogeneous; otherwise, a self-similarity map contains some homogeneous blocks which correspond to sounds in the input audio. Once the self-similarity map is generated, the next step is to accurately detect abrupt changes of pixels between blocks which represent changes of sounds in the audio.

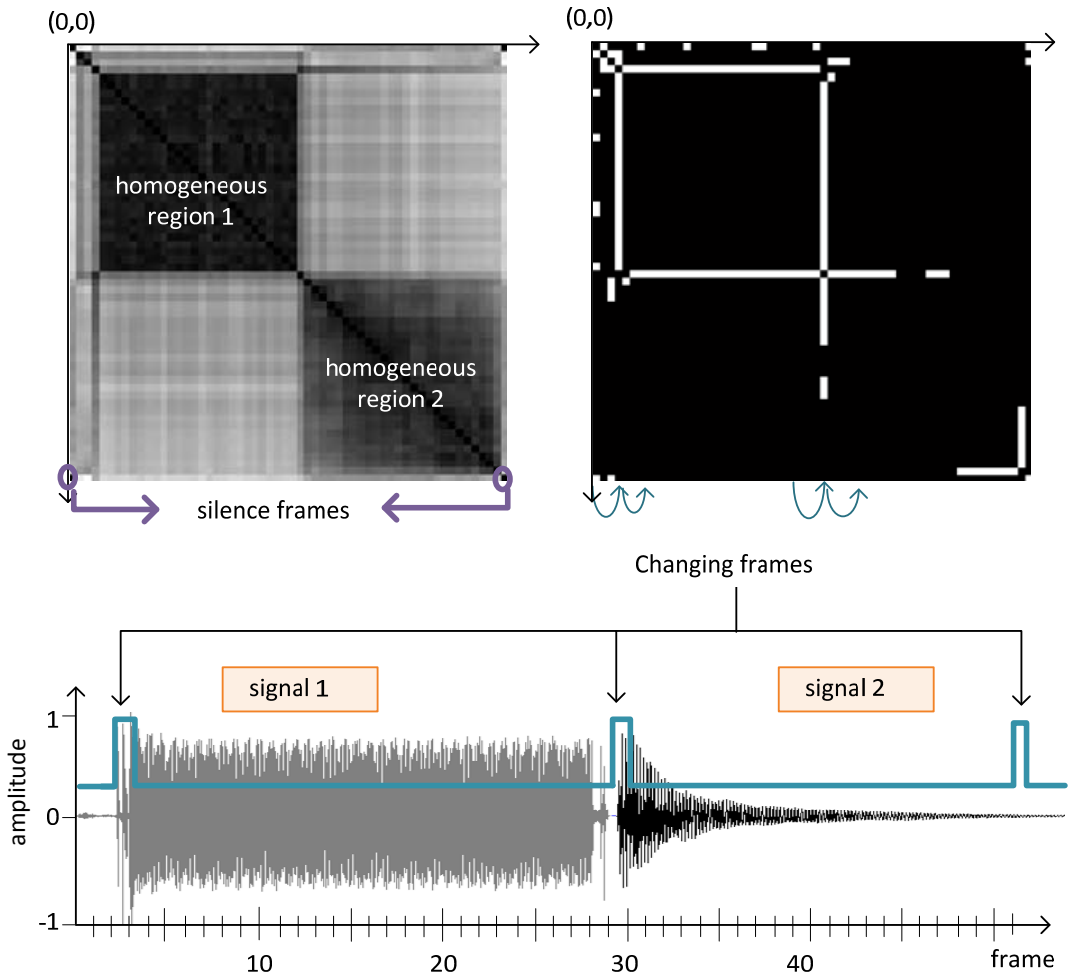

Figure 3-6: Top: (L) Similarity map image; (R) Edge detection of similarity map image; Bottom: Segmentation result based on similarity map image.

In Foote's segmentation method a checkerboard kernel is used to calculate cross-similarity values between different regions along the diagonal of the self-similarity map. Kernel size affects the accuracy. A small kernel tends to be sensitive and on a short time scale and is capable of finding detailed changes. On the other hand, a large kernel is suitable for those audio files that have few segments but each segment contains several sounds. Because it is impossible to find a kernel size that suits all kinds of sounds, an edge detection method in image processing is employed in our segmentation method, instead of a kernel comparison on a similarity map, to locate changes in audio properties. Edge detection in self similarity maps has not been previously used to assist audio file segmentation.

Sobel edge detection [96] is commonly used in image processing to detect the edges of an image. Here we employ Sobel edge detection, for the first time in audio segmentation, to identify any abrupt changes in the audio self-similarity map. When applying Sobel edge detection, a heterogeneous image can be separated into

homogeneous regions. The edge pixels of the regions represent the boundaries in the audio clip. In this way, the audio clip is then separated into homogeneous sound regions.

The audio clip in Figure 3-5 (lower image) is used as an example: its similarity map is the left image of the top row in Figure 3-6. The similarity map has zeros along its diagonal because the distance is zero from each frame to itself. There are two homogeneous blocks in the similarity map that indicate two homogeneous sounds in the audio clip. Figure 3-6 Top (R) gives the resultant image from performing Sobel edge detection on the self-similarity map (Figure 3-6 Top (L)). The changing frames are clearly evident in the resultant edge detection image. The final segmentation result for the phone and violin mixed audio clip is illustrated in Figure 3-6 (bottom image). It can be seen that the edges of the similarity map accurately illustrate the boundaries of the sounds.

This second phase of segmentation is performed only on audio clips that are longer than ten frames, since there is little possibility that audio clips shorter than ten frames contain multiple distinguishable sounds.

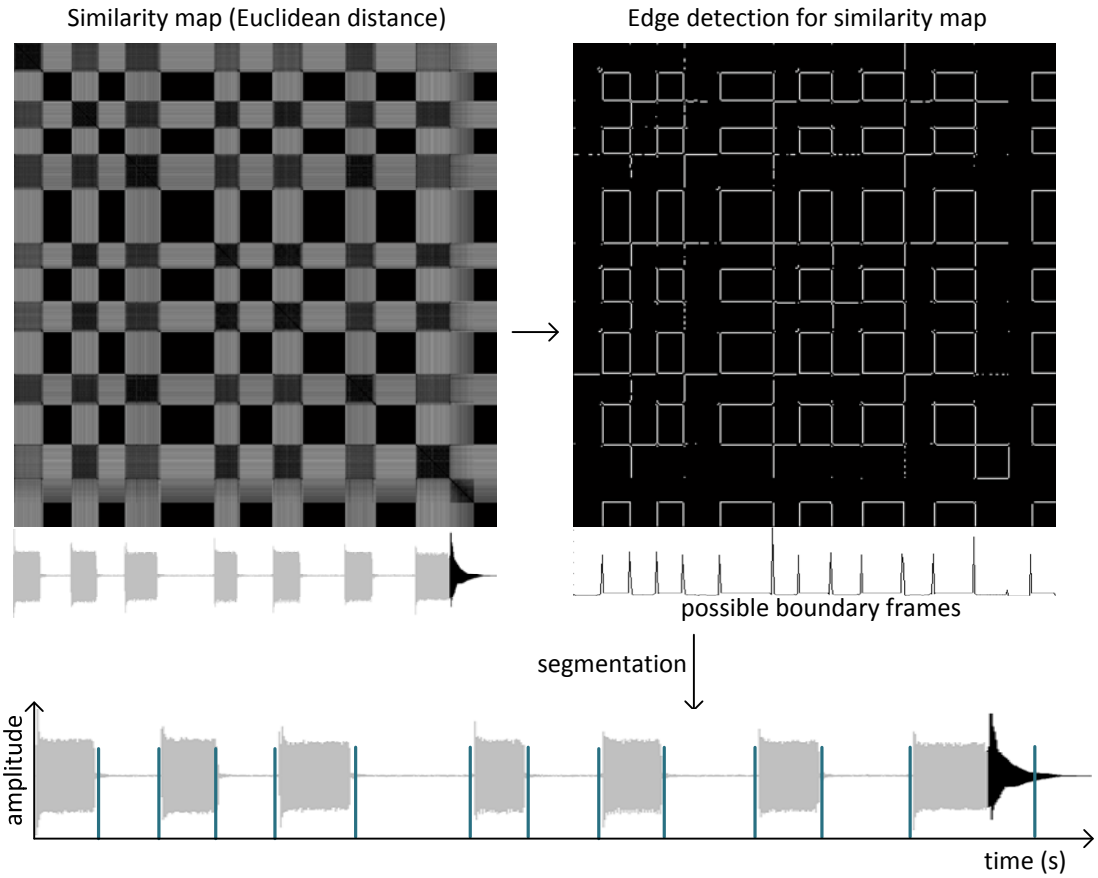

Figure 3-7: Result of segmenting the audio file (Figure 3-5) using Euclidean-d similarity segmentation method.

An alternative method used in detecting self similarity that has been employed previously is the use of a Euclidean distance (Euclidean-d or $d_{Euclidean}$) similarity measure. Figure 3-7 shows the result of applying one phase Euclidean-d as a similarity segmentation method using the same audio file as in Figure 3-5. The audio clip seven in the upper image of Figure 3-5 is not separated when a one-phase Euclidean-d measure is used. This example clearly illustrates that the two-phase segmentation method (Figure 3-6) is better than the silence detection method (where clip seven is not segmented) or the similarity segmentation methods (Figure 3-7) because it correctly separates the two sounds in audio clip seven.

The novel two-phase segmentation developed here is a method that was designed for both long and short duration audio clips. Figure 3-8 shows results obtained using this segmentation method with clips of varying lengths in a single audio file. In this example, there are five audio clips. The first and fifth have a relatively long duration and the rest are relatively short. The resulting segmentation is demarcated by blue lines superimposed on the original waveform. Segmentation generates a series of sound clips in which each sound clip is a segment of audio, containing a homogeneous set of features for each frame.

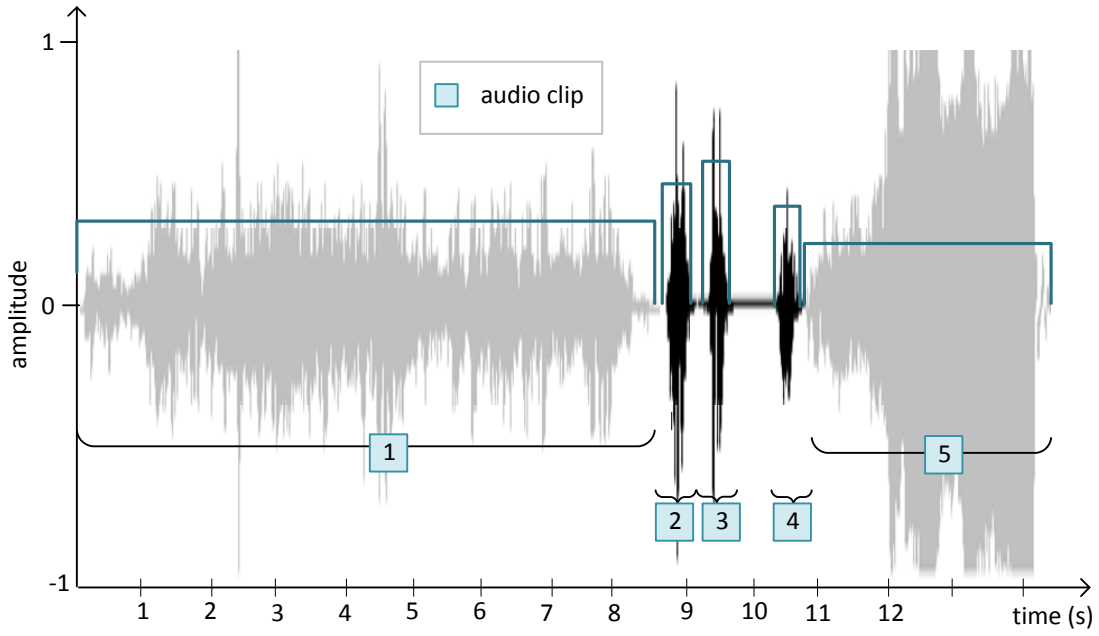

Figure 3-8: Segmentation result for long and short signals.

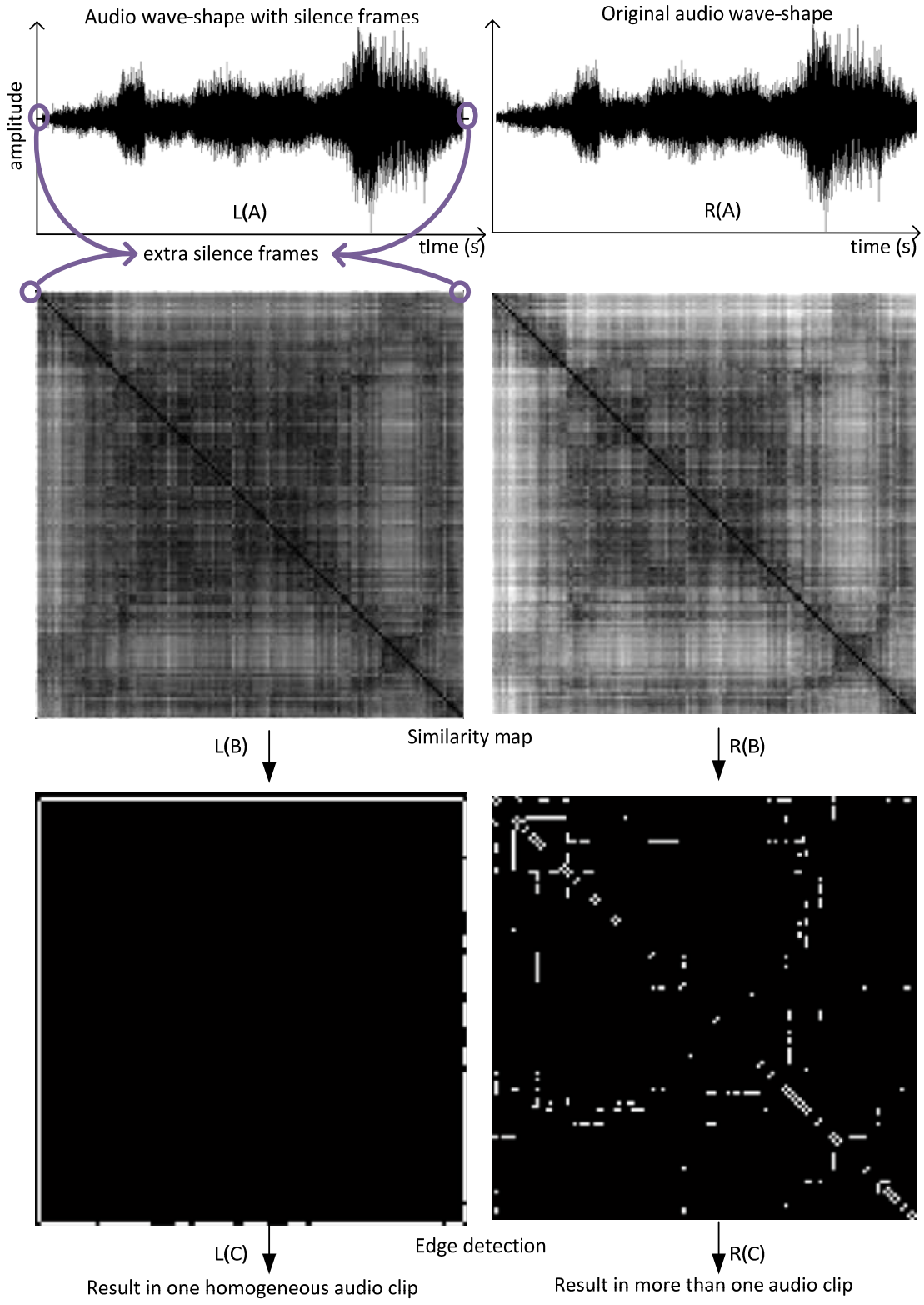

Figure 3-9: The comparison of the similarity map with and without adding silence frames. Images in the left column are a homogenous audio with 2 extra silence frames, its similarity map and edge detection result. Images in the right column are the same audio without silence frames and the corresponding results.

In general, the similarity matrix has minimum values of zero on the diagonal because each frame has a maximum similarity to itself by definition. For a homogeneous audio

clip, the real values of similarity distance are small, but are extended in the similarity map after normalization because no matter what the real value of the maximum similarity distance, it is scaled to one. To represent the audio clip correctly, two silence frames are added at the front and the end of the audio clip. After adding a silence frame, the maximum similarity distance in the audio clip is a signal frame compared with silence frames. Similarity distances of homogeneous audio clips will not be extended. The images in Figure 3-9 illustrate the importance of adding silence frames to a homogeneous audio clip. Image L(A) in Figure 3-9 is the wave-shape of the audio shown in image R(A) with two extra silence frames at the front and at the end. Their corresponding normalized self-similarity maps are shown in images L(B) and R(B). Compared with image L(B), the differences are accentuated in image R(B), which is the normalized self-similarity map of an audio clip without adding extra silence frames.

Further edge detection based on R(B) gives image R(C). The self-similarity map is separated into many small regions which correspond to improper small audio clips. However, image L(B) keeps homogeneity because, in comparison with silence, frames within this homogenous audio signal have similar properties. The edge detection result shows there is no obvious change within the audio clip. There are two silence frames added in the example of Figure 3-6 too. It shows that for heterogeneous input audio, adding silence frames does not affect the segmentation result.

The pixel S_{ij} in the self-similarity map of an audio file represents the distance between the feature vectors of frame *i* and frame *j* in the audio. There are many methods that can be used to compare and derive the differences between two vectors. Some of them can be adapted for self-similarity matrix generation. The similarity map can be generated from different distance extraction methods, such as the Euclidean-d mentioned in [17] [81], the Cosine of the angle between frame vectors [82] [93], the Kullback Leibler distance (KL2) [87], and the Hotelling T2-Statistic distance [97]. Ong [93] claimed that compared with Euclidean-d, the cosine of the angle between two vectors (named cosine-d or d_{cosine} in this chapter) yields a large similarity score even if the vectors have small magnitudes. Besides these distance equations, Chen et al. [56] presented a similarity equation for two vectors, which measures the differences in the magnitude and angle between two vectors. Based on these existing similarity extraction equations, four more vector similarity extraction equations are presented here. The performance of each of

these possible equations, in the two-phase audio segmentation method, was evaluated. A detailed discussion of the calculation of these similarity extraction methods follows.

Given two audio frames with *n*-dimension audio vectors *a* and *b*, then we define θ as the angle between *a* and *b*. The existing methods are defined in Table 3-1.

Table 3-1: Existing equations for similarity between two vectors.

	Equation	
Euclidean-d	$d_{Euclidean}(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$	(E 3-2)
Cosine-d (Dot product)	$d_{Cosine}(a,b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$	(E 3-3)
Chen-d	$d_{chen}(a,b) = \frac{\alpha}{2} * (1 - d_{cosine}(a,b)) + (1-\alpha) * (1 - d_{log}(a,b))$	(E 3-4)

Viewed geometrically, vectors have length (magnitude) and direction. Because the length of each vector is a value between 0 and 1, the distance between two vectors $d_{Euclidean}(a,b)$ is between 0 and $\sqrt{n}$ and gives the magnitude of vector ||*a*-*b*||. $d_{Cosine}(a,b)$ also equals a value between 0 and 1, and gives the cosine of angle θ. Chen's vector distance equation (also named Chen-d or d_{Chen} in this chapter) consists of two parts. The first part d_{log} estimates the difference between the magnitudes of two vectors. In Chen-d, when $\alpha = 0.5$, the magnitude (d_{log}) and direction ($d_{Cosine}(a,b)$) of the vectors have the same weights. In Chen's paper and this thesis, α =0.5 when extracting d_{Chen}. The same value for α is employed in the equations $d_{Angle}(a,b)$, $d_{Sine-Chen}(a,b)$, $d_{Linear-Chen1}(a,b)$ and $d_{Linear-Chen2}(a,b)$. The variable of the function d_{log} is $\|a\| / \|a\| + \|b\|$ and the result of length differences is:

$$d_{log}(a,b) = -\frac{\|a\|}{\|a\|+\|b\|} log_2\left(\frac{\|a\|}{\|a\|+\|b\|}\right) - \frac{\|b\|}{\|a\|+\|b\|} log_2\left(\frac{\|b\|}{\|a\|+\|b\|}\right) \quad \text{(E 3-5)}$$

When the coefficient of Chen-d equation α is set to 0.5, length and direction have the same weight in computing the distance between the two vectors. Chen-d was the first method to use both magnitude and direction when calculating the distance between two vectors. Chen-d values are between 0 and 1 according to the following:

$$d_{chen}(a,b)=\begin{cases} 0 & \|a\|=\|b\| \quad and \quad \theta=0 \\ 0.5 & \|a\|=\|b\| \quad \theta=\frac{\pi}{2} \quad or \quad \|a\|=0 \quad or \quad \|b\|=0 \quad \theta=0 \\ 1 & \|a\|=0 \quad or \quad \|b\|=0 \quad and \quad \theta=\frac{\pi}{2} \end{cases}$$

Based on these existing methods, four new equations for comparing the similarity between two vectors have been developed and are presented in Table 3-2.

Table 3-2: New methods for calculating the similarity between two vectors.

	Equation	
Angle distance	$d_{Angle}(a,b)=\frac{\theta}{\pi/2}$ or $d_{Angle}(a,b)=\frac{arccos(\frac{a\cdot b}{\|a\|\cdot\|b\|})}{\pi/2}$	(E 3-6)
Sine-Chen distance	$d_{Sine-chen}(a,b)=\frac{\alpha}{2}*(1-d_{cosine}(a,b))+(1-\alpha)*(1-d_{sine}(a,b))$	(E 3-7)
Linear-Chen1 distance	$d_{Linear-chen1}(a,b)=\frac{\alpha}{2}*(1-d_{cosine}(a,b))+(1-\alpha)*(1-d_{linear}(a,b))$	(E 3-8)
Linear-Chen2 distance	$d_{Linear-chen2}(a,b)=\frac{\alpha}{2}*d_{Angle}+(1-\alpha)*(1-d_{linear}(a,b))$	(E 3-9)

$d_{Angle}(a,b)$ is the normalized angle between two vectors and represents the directional difference between vectors *a* and *b*. Whereas $d_{Cosine}(a,b)$ produces a curvilinear representation for the angle between vectors *a* and *b*, $d_{Angle}(a,b)$ produces a linear curve. Figure 3-10 (L) shows the difference between the two methods of calculating distances.

In Chen-d, the curve of d_{log} is a logarithmic curve for $\|a\| / \|a\|+\|b\|$ (Figure 3-10 (R)), which ranges from [0, 1]. In an *n*-dimensional feature space, vectors, $\|a\|$ and $\|b\|$ range from [0, $\sqrt{n}$].

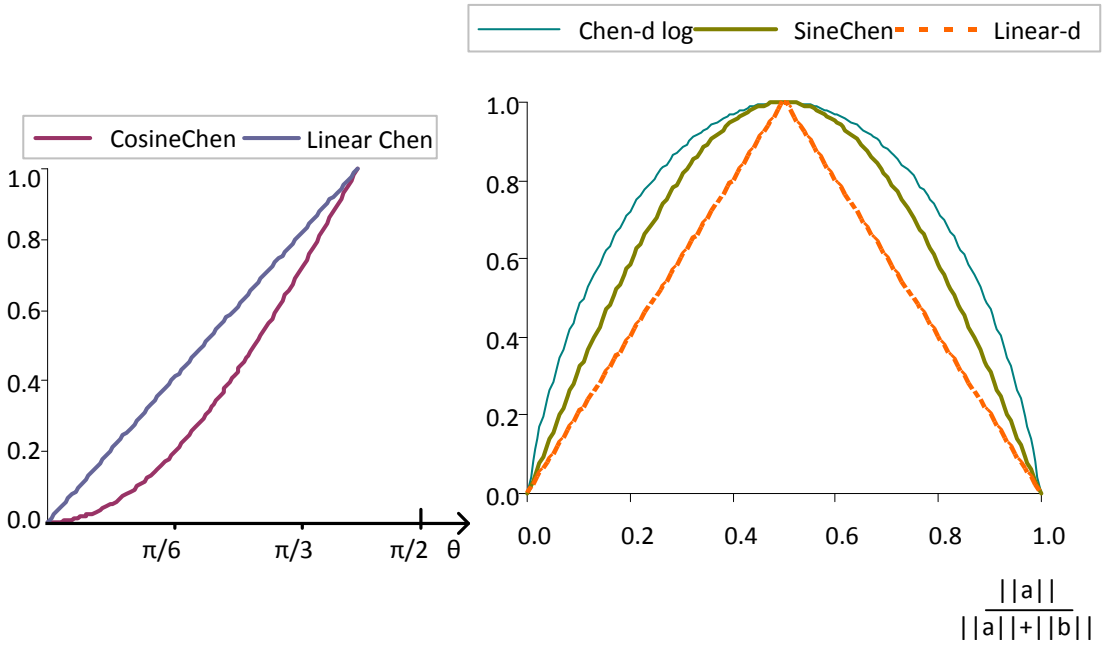

Figure 3-10: (L) The comparison of $d_{cosine}(a,b)$ and $d_{Angle}(a,b)$.
(R) The comparison for distances d_{log}, d_{Sine}, and d_{Linear}.

Figure 3-10 (R) visually compares the Chen-d measure with two new variants, Sine-Chen and Linear-Chen. $d_{Sine-Chen}(a,b)$ uses a sine curve instead of the logarithmic curve in Chen-d:

$$d_{sine}(a,b) = sin(\frac{\|a\|}{\|a\|+\|b\|} \times \pi) \qquad \text{(E 3-10)}$$

Figure 3-10 (R) shows that Sine-Chen distance ranges from [0,1]. As in Chen-d, the magnitude and direction of the vectors have the same weight when $\alpha = 0.5$.

The graph of $d_{Linear}(a, b)$ is the polyline in Figure 3-10 (R). The polyline changes linearly with variable $||a|| / ||a|| + ||b||$.

$$d_{Linear}(a,b) = \begin{cases} \dfrac{\|a\|}{\|a\|+\|b\|} & if \quad \dfrac{\|a\|}{\|a\|+\|b\|} < 0.5 \\ 1 - \dfrac{\|a\|}{\|a\|+\|b\|} & others \end{cases} \qquad \text{(E 3-11)}$$

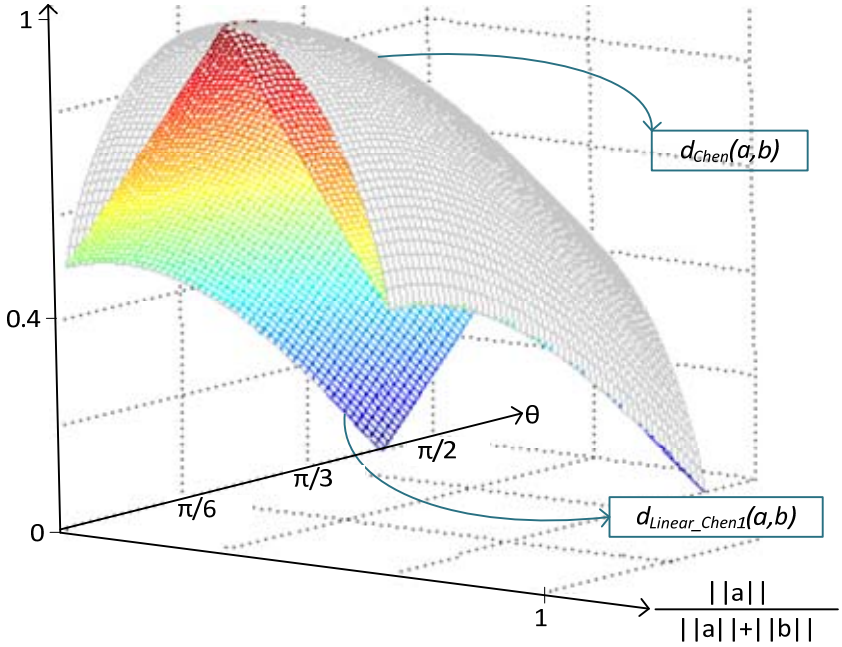

Figure 3-11: Visualization of distance $d_{Chen}(a,b)$ and $d_{Linear\text{-}Chen1}(a,b)$.

The d_{Linear} line length is also clamped between zero and one. The distances $d_{Linear\text{-}Chen1}(a, b)$ and $d_{Chen}(a, b)$ are visualized in Figure 3-11 as surfaces. The $d_{Linear\text{-}Chen1}(a, b)$ distance is the upper one because its $d_{Log}(a, b)$ is bigger than $d_{Linear}(a, b)$ when the value of ($||a|| / ||a|| + ||b||$) is neither 0, 0.5, or 1.

Table 3-1 and Table 3-2 contain seven equations that can be used to extract the similarity (distance) between two vectors in *n*-dimension audio feature space. Table 3-2 shows four novel equations, discussed in detail earlier in this section, that were derived in an attempt to improve on the results of the existing approaches (Table 3-1). The tests undertaken to evaluate and compare the performance of all seven measures in the two-phase audio segmentation method are reported in the following sections.

3.4 Evaluation Method

A widely used standard non-musical sound database, called *MuscleFish* [1] [98], is employed to fully evaluate this two-phase segmentation method. *MuscleFish* contains 16 classes and 410 audio files as described in Table 3-3, with N_c being the number of sounds in each class. The duration of the sounds varies from less than 1 second to approximately 15 seconds. When compared to databases in previously reported

segmentation evaluations, this database introduces some significant segmentation issues. For example, classes "Alto trombone", "Cello(bowed)", "Oboe", "Percussion", "Tubular Bells", "Violin (bowed)" and "Violin (pizz)" belong to the category of music so these audio files are acoustically similar. Moreover, the database has a broad range of classes, including musical instrument samples, animals, machines, speech, and everyday sounds (e.g. thunderstorm and hand clapping).

Table 3-3: The 410 audio files and 16 classes in the *MuscleFish* audio database.

MuscleFish Classes	N_c	*MuscleFish* Classes	N_c
Alto trombone	13	Male Voice	17
Animals	9	Oboe	32
Bells	7	Percussion	99
Cello (bowed)	47	Telephone	17
Crowds	4	Tubular Bells	20
Female Voice	35	Violin (bowed)	45
Laughter	7	Violin (pizz)	40
Machines	11	Water	7

For segmentation evaluation, 1000 audio files were generated. Each file was built by conjoining two randomly selected files from the *MuscleFish* database. Each file in the *MuscleFish* database contains from one to seven sounds. Thus the generated test files may contain anywhere from two to fourteen sounds. These 1000 audio files were separated into 10 sets of 100 files and these sets were used for testing the accuracy of segmentation methods. For example, the generated test audio file illustrated in Figure 3-5 has seven sounds where the first six sounds belong to the first audio file and the last sound belongs to the second audio file. Because the number of sounds in a test audio file is unknown, experiments based on these audio files are general and comprehensive.

The accuracy of the segmentation method is compared with the manually marked audio file to ensure that the correct segments or sounds in the file have been identified. This is the generally accepted way of verifying the performance of segmentation methods in the literature reviewed.

There is no consistent evaluation method that can be applied for all possible audio applications. Martin, Scheirer and Vercoe [99] also note that there is no ground truth for

music. The segmentation results are therefore context dependent. The correctness of segmentation depends on the application and should be perceptually meaningful. As the primary module of an audio visualization system, segmentation should partition sounds by using acoustical differences. Although the physical end of the first sound and the physical beginning of the second sound is the actual boundary of the two sounds, more tolerant rules for a "correct segmentation result" are adopted according to the specific purpose of the audio visualization system thus:

- In this application, if two signals belong to the same class, any segmentation result is regarded as "correct" because the two signals can be visualized by the same kind of video/image.
- One tolerance frame before and after the actual boundary is integrated for evaluation. This small tolerance is introduced because, for the actual boundary frame, it is possible that some samples in the front belong to the sound in front of it and the remaining samples belong to the sound after the frame. So the actual boundary frame is acoustically different from its two neighbouring frames. Any one of these three frames can be regarded as a boundary frame. In other words, suppose the real boundary frame is numbered *i*, any one of frame *i-1*, frame *i* and frame *i+1* is correct for separating the two neighbouring sounds.
- If an audio file is separated into smaller pieces because the silence periods between are long enough to cause partitioning, the results with smaller pieces are accepted as correct.

To evaluate the two-phase segmentation method its accuracy is compared with existing self-similarity methods (Euclidean-d and Cosine-d) and an existing silence detection method. In the two-phase segmentation method, the performance may be affected by other factors, such as the equations used to generate the similarity map in the 2nd phase, the audio feature extraction algorithms and audio feature set selections. The different combinations of these factors, as well as different similarity calculation methods, are tested and compared in the following section.

3.5 Experiments and Analysis

The various different factors or parameters that affect the accuracy of audio segmentation results were discussed in 3.4 above. A corresponding experiment has been designed for each factor.

- Generally speaking, accurate audio feature extraction requires sounds of adequate duration such as the standard frame size of 16ms found in Li [98]. However, some experiments are required to ensure a suitable frame size is employed.
- After extracting the audio features for all the frames, the values of all audio frames are normalized to [0, 1] for each feature. This ensures that each feature has the same weighting in the distance calculations. After normalization, the feature vector of a silence frame may be non-zero. Some experiments are also necessary to test whether or not the feature vectors for silence frames need to be reset to zero. In the experiments described below the audio feature vectors for silence frames after normalization are either retained (general normalization) or clamped to zero (modified normalization).
- A suitable equation for vector similarity distance extraction is required for the second phase of this segmentation method. The audio feature set is another important factor for the segmentation accuracy. Thus experiments with different audio feature sets are essential.

The following experiments were undertaken using the 10 groups of audio files (100 audio file each) the generation of which was discussed in Section 3.4. Because each parameter in the segmentation process (i.e. frame size, normalization method and feature set) is interdependent an initial assumption must be made about one of these three parameters. In the first two experiments, to enable the selection of a suitable frame size and audio feature vector normalization method, the PercCeps8 feature set was selected. The following audio features form the PercCeps8 feature set:

- Total Spectrum Power: the spectral density of the signal wave (the power carried by the signal wave per unit frequency).
- Brightness: the frequency centroid of given audio signal.
- Bandwidth: the square root of the power weighted average of the squared difference between the spectral components and the frequency centroid.
- Pitch: the mean fundamental frequency (F_0).
- Eight order Mel Frequency Cepstral Coefficients (MFCCs): a compact representation of an audio spectrum that takes into account the nonlinear human perception of pitch, as described by the Mel scale.

These features were selected because their combination has been shown to give the most accurate classification results for the *MuscleFish* database [98] and have been employed in post-classification segmentation [75]. It is reasonable to assume that this feature set therefore provides the best possible representation of general audio files and would also therefore be appropriate for the segmentation of such audio files. Details of the computation of these features can be found in Li [98]. Additionally, although the feature set PercCeps8 was normalized according to the training set [98], which is not the same as an unsupervised audio processing, it provides a good indication of possible audio features and feature combinations that are worth trying in the experiments for establishing parameters for the two phase segmentation method.

Experiment One: Frame size determination

The performance of the two-phase method using two different frame sizes, 16ms and 32ms were tested. These two frame sizes were selected because they are commonly used in audio processing. The feature vectors were normalized and the silence frames retained using Li's method [98]. For silence detection methods normalization is not required and is therefore not used in the silence experiment here.

Table 3-4: % average accuracy of segmentation methods using different frame sizes.

Frame Size	One-phase			Two-phase	
	Silence	Cosine-d	Euclidean-d	Cosine-d	Euclidean-d
32ms	81.0	87.5	89.6	88.8	93.9
16ms	82.7	91.4	94.6	92.7	96.6

Table 3-4 and Figure 3-12 show the percentage average accuracy for each approach. It is clear that in all five methods a 16ms frame size gave better accuracy. The results when using the best performing Euclidean-d two phase segmentation method with a 16ms frame size was significantly more accurate than those achieved using a 32ms frame size; $\chi^2 = 8.06$, $df = 1$, $p < 0.01$.

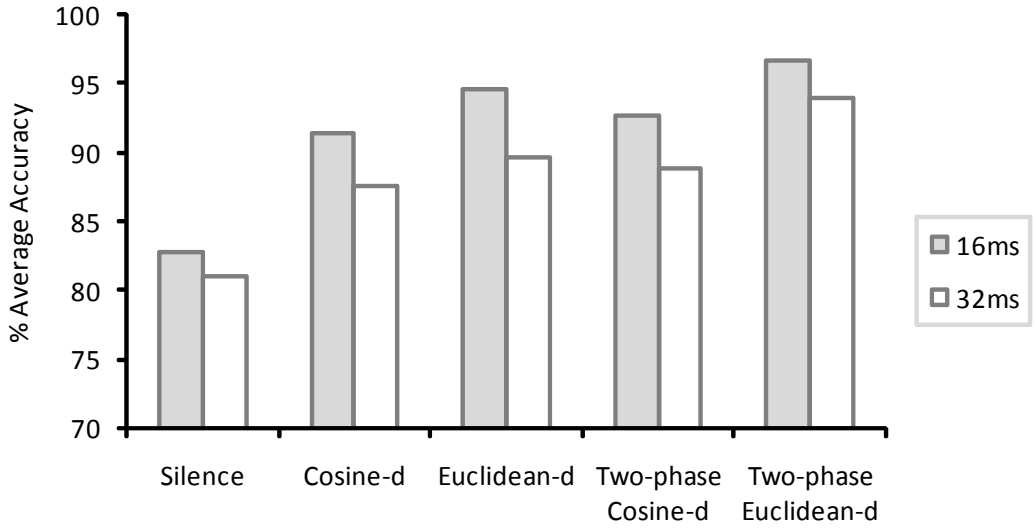

Figure 3-12: Comparison of different frame sizes (16ms and 32ms).

Experiment Two: Normalization

In an attempt to improve the accuracy of segmentation we explored an alternative approach to normalize the audio feature vectors. Typically, during normalization, audio features for a silence frame may be scaled to a non-zero vector because some audio features can be negative (MFCC features). In Li [98] and our experiment one the audio feature vectors for silence frames after normalization were retained (general normalization). We proposed a new approach, modified normalization, in which the feature vectors for silence frames were clamped to zero. This modified normalization approach was compared with the standard reported general normalization approach. We examined the accuracy of various segmentation methods using a 16ms frame size and the PercCeps8 feature set (Table 3-5).

Table 3-5: Average accuracy of segmentation methods using different normalization approaches.

Normalization Method	One-phase		Two-phase	
	Cosine-d	Euclidean-d	Cosine-d	Euclidean-d
modified	85.1	86	86.7	92.8
general	91.4	94.6	92.7	96.6

Unfortunately the modified normalization method in all cases reduced the accuracy of the segmentation.

The results from experiment one indicated that our proposed two phase method using Euclidean-d should provide the best result when segmenting audio files derived from the *MuscleFish* database. Therefore the next experiment was designed to explore the hypothesis that the two-phase method is better than the one-phase method for general audio segmentation.

Experiment Three: Comparison of segmentation methods

In this experiment the accuracy of our two-phase segmentation method was compared with existing methods. Again we used the experimentally determined best frame size of 16ms and the best reported feature set PercCeps8. Experiment two determined that the best normalization method was the general approach and the audio feature vectors calculated using the PercCeps8 feature set were therefore normalized using this method.

Table 3-6 shows the segmentation accuracy using the previously described methods. When examining the accuracy by group it can be seen that the silence method is the least reliable. The best silence edge detection result accurately segments 91 of the 100 audio files in group 1 and the worst accurately segments 76 of the 100 files in group 2.

Table 3-6: Accuracy of the different segmentation methods by audio file group.

Group	One-phase			Two-phase	
	Silence	**Cosine-d**	**Euclidean-d**	**Cosine-d**	**Euclidean-d**
1	91	96	98	96	98
2	76	86	90	89	95
3	84	94	98	95	99
4	82	90	94	93	96
5	83	90	92	91	94
6	84	91	92	92	96
7	89	97	96	97	98
8	79	86	94	88	95
9	79	94	98	95	99
10	80	90	94	91	96
Average	82.7	91.4	94.6	92.7	96.6

Variations in the audio file features appear to have a large impact on the accuracy of this method. The two phase Euclidean method has the least variation between groups and is also the most accurate method. This two-phase method with Euclidean-d in a similarity map provided the most accurate segmentation result with an average of 96.6%. The method using only silence detection gave the worst average result (82.7%). The poor performance of the segmentation based on silence detection is not unexpected because some audio files in the *MuscleFish* database do not contain short silence periods so there is no obvious boundary between two sounds in the wave-shape of the generated audio files.

The Euclidean-d gave better accuracy, regardless of the approach (two-phase method or the similarity map alone), than the cosine-d. The two-phase method gave greater accuracy than the use of the similarity map alone. Moreover, the Euclidean-d two-phase method was significantly more accurate that the second most accurate method, Euclidean-d; $\chi^2 = 4.75$, $df = 1$, $p < 0.05$. Cosine-d is mainly used for calculating the dissimilarities between vectors of small magnitude. In contrast the magnitudes of the audio feature vectors for audio in the *MuscleFish* database are relatively long. Where the vectors have a higher magnitude Euclidean-d has been shown to perform better [82] so it is not surprising that for the *MuscleFish* database Euclidean-d proved to perform better than Cosine-d.

Because the two-phase segmentation approach looked promising further experiments were undertaken focusing solely on the two phase method and exploring a number of potential feature sets.

Experiment Four: Evaluation of feature sets in the two-phase method

For segmentation there is always a tradeoff between the accuracy and the efficiency of the feature calculations. More features give a more accurate result but can significantly affect the computing efficiency. This experiment was designed to determine the degree of contribution of the features (see Table 3-7) used in the PercCeps8 feature set (Section 3.5). If a set with fewer features could be employed the usability of the audio visualization may be improved. A frame size of 16ms and a general normalization method were used. Each feature set was tested in the two-phase segmentation method using seven vector distance extractions.

Table 3-7: The three new feature sets tested.

Feature set	features				
	Bandwidth	**Brightness**	**Pitch**	**MFCC8**	**Spectral Power**
PercCeps8	✓	✓	✓	✓	✓
1	✓	✓	✓	✓	
2	✓	✓	✓		
3				✓	

Table 3-8: Percentage average accuracy using four different feature sets and seven different vector distance calculations in the two-phase method.

Feature Set	vector distance calculation method						
	Euclidean-d	**Cosine-d**	**Angle**	**Chen-d**	**SineChen**	**Linear-Chen1**	**Linear-Chen2**
PercCeps8	96.6	92.7	94.3	92.7	92.6	93.1	94.6
1	92.6	87	90.5	84.7	84.9	89.5	95.6
2	90.8	86.3	87.4	82.2	82.8	87.6	83.5
3	92.2	86.7	90.1	84.8	85.3	89.0	93.0

The results of the experiments are given in Table 3-8 and the average accuracies for different audio feature sets are shown in Figure 3-13. The PercCeps8 audio feature set gave the most accurate results in 6 of the 7 segmentation methods and audio feature set 2 (Bandwidth, Brightness and Pitch) was consistently the worst result.

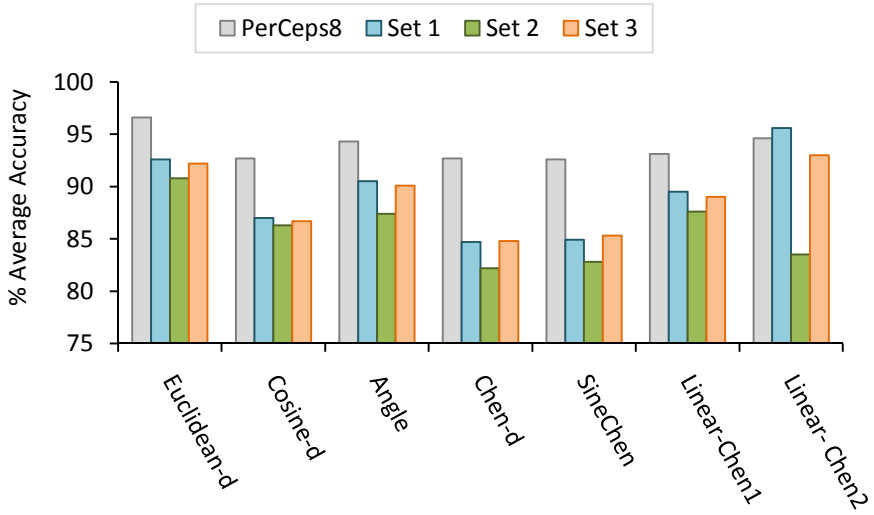

Figure 3-13: Comparison of different audio feature sets.

The results show that the MFCC feature contributes significantly to the accuracy of the segmentation. Because the overall accuracy is relatively high, the improvement gained by choosing any particular audio feature set or method is not large but some combinations seem comparatively more accurate than others. In our case accuracy of the segmentation is critical to the subsequent audio classification step and it is clear from this experiment that PercCeps8 is the most appropriate feature set for our application. For an application where speed is more important than accuracy any of the subsets may be appropriate.

Experiment Five: Using a larger database

The hypothesis we have posed is that the two-phase segmentation method is more accurate than the one-phase method and that it should be applicable to other audio data stores.

Because the segmentation is the pre-classification step we must also consider the potential affect of a larger database on the classification accuracy. Solely more audio files do not necessarily lead to less accurate classification results. More data may lead to more concrete clusters if there are a small number of classes. Additionally, it may lead to more overlapping regions between the two clusters. The segmentation's performance may be affected because the feature set and distance calculations that we have experimentally established may be too specific to the audio files in the *MuscleFish* database.

In this experiment we examined the accuracy of the two-phase segmentation using a larger database (*VisualData*; see Section 4.6 for a full discussion of the *VisualData* database and ontology). Not only is this database larger (contains more audio files) but it also has a broader and more hierarchical ontology. A frame size of 16ms is used along with general normalization and the PercCeps8 feature set.

Using the larger database the two phase segmentation method gives the best accuracy regardless of the vector distance calculation method (Table 3-9). Although the accuracy for Euclidean-d two-phase method with the *VisualData* database (92.8%) is slightly

lower than for the *MuscleFish* database (96.6%) the gap between two-phase and one-phase accuracy increased. For the *MuscleFish* database the difference between the accuracy of the one-phase method and that of the two-phase method is two percent whereas with the *VisualData* database the two-phase method is 10.3% more accurate.

Table 3-9: The percentage average accuracy of segmentation using the *VisualData* database.

Segmentation Method	vector distance calculation method						
	Euclidean-d	**Cosine-d**	**Angle**	**Chen-d**	**SineChen**	**Linear-Chen1**	**Linear-Chen2**
Two-Phase	92.8	91.4	89.7	89.0	89.4	92.1	92.0
One-Phase	82.5	81.2	81.0	79.1	78.3	80.1	80.1

On the basis of these experiments it can be concluded that:

- The two-phase audio segmentation method performs better than existing approaches such as segmentation based on silence detection and self-similarity segmentations.
- Frame size affects the segmentation accuracy.
- Different audio feature sets lead to different segmentation results.
- The standard Euclidean-d calculation (Section 3.3, Table 3-1) performs better than any of the novel similarity equations discussed in Section 3.3 (Table 3-2).

To fully illustrate how the two-phase segmentation method performs, the five segmentation methods in experiment 1 were undertaken using the audio file illustrated in Figure 3-14. This audio file was generated by conjoining the audio files "female3.cosponsor.au" and "teltt7.au" from the *MuscleFish* database. The upper image in Figure 3-14 gives the shape of the audio file. The first audio file is marked in grey and the second is marked in black in the image. Each frame lasts 16ms and is marked in the image. The segmentation results are illustrated in Figure 3-14. In this instance the Euclidean-d gave the correct segmentation result using the two-phase segmentation method. The results from the other methods are incorrect.

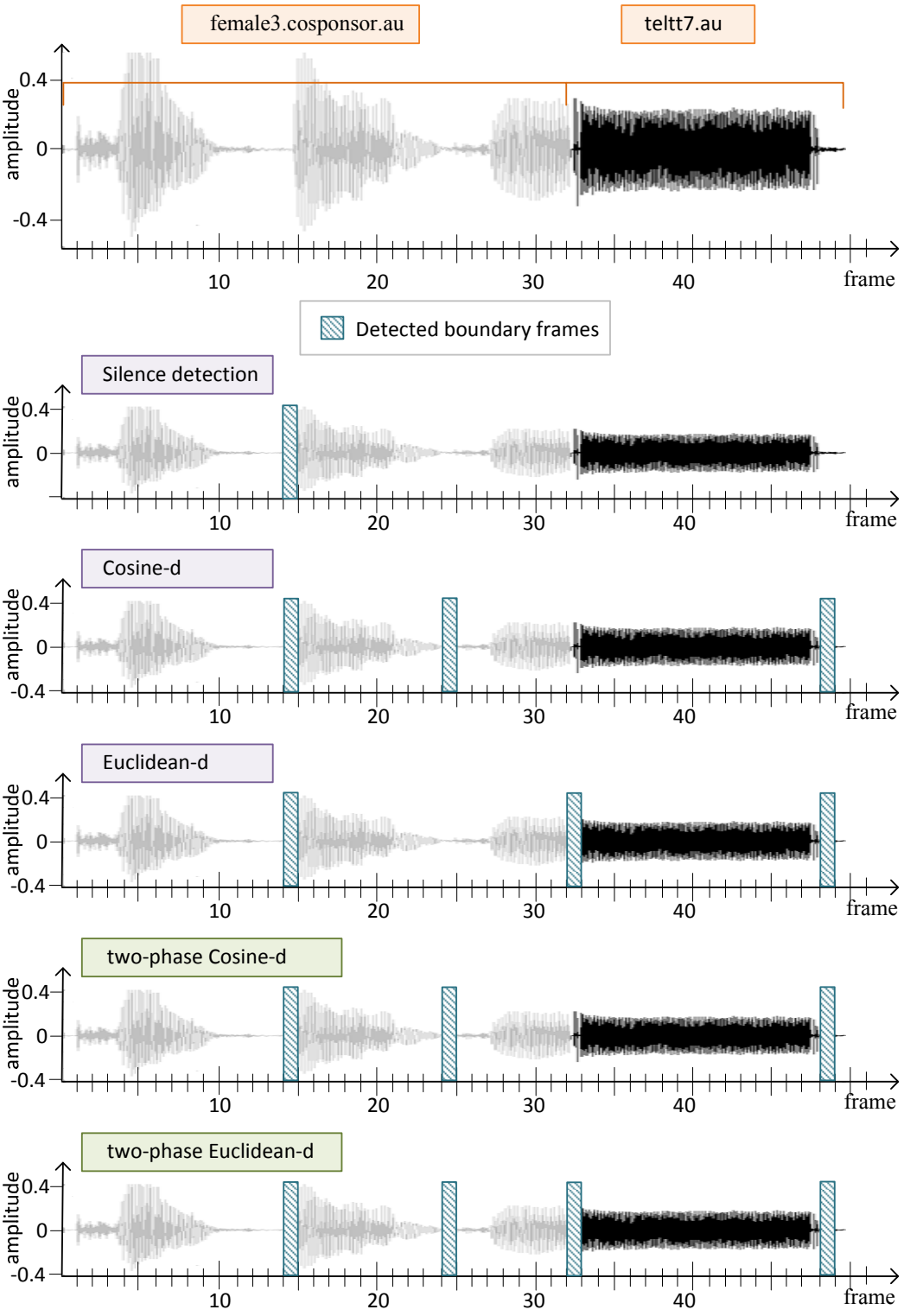

Figure 3-14: Segmentation experiment results using 5 different methods.

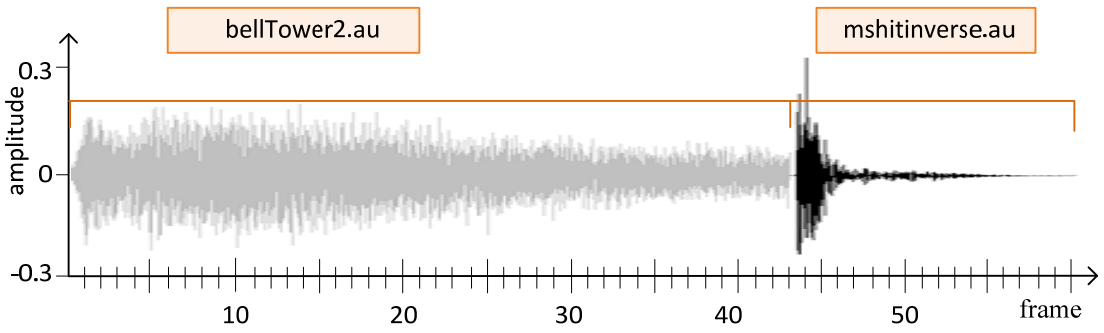

Figure 3-15: The starting audio file amplitude vs. frames (16ms).

Figure 3-15 shows a repeat of the previous experiment using another audio file. In this case the input audio file is generated from the audio files "bellTower2.au" and "mshitinverse.au", marked in grey and black in the image respectively.

Their segmentation results are illustrated in Figure 3-16. The Cosine-d and Sine-Chen distance give incorrect segmentation results. The results from the other distance measures are correct.

The two examples discussed above are just two of the 1000 audio files tested and have been chosen to illustrate that two-phase segmentation method works well for heterogeneous audio files even if there is no obvious silence (the silence period is shorter than one frame) between different sounds.

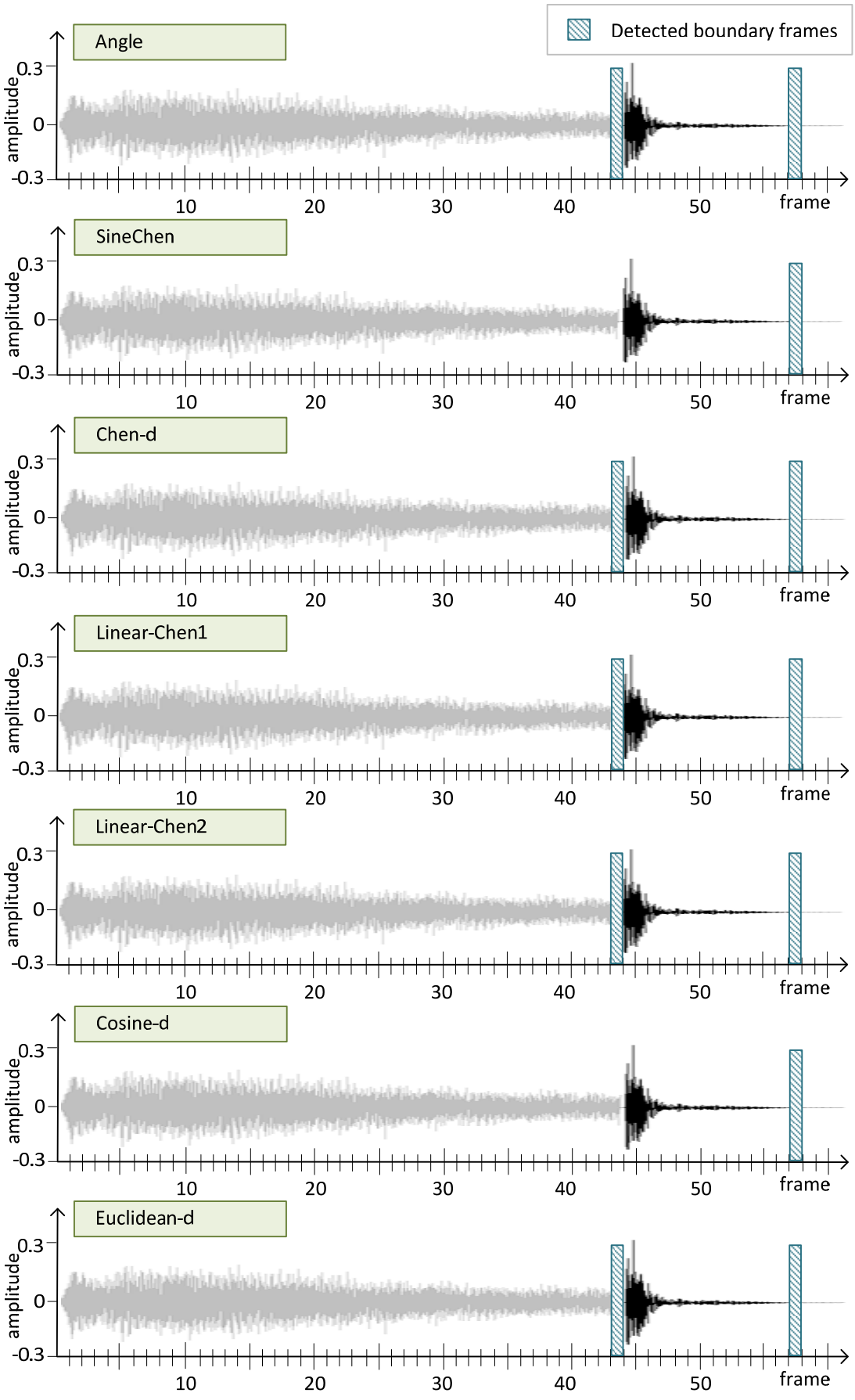

Figure 3-16: Segmentation results for audio shape in Figure 3-15 using our two-phase method.

3.6 Chapter Summary

This chapter describes a model-free and training-free two-phase method for general audio segmentation. The experiments detailed in Section 3.5 prove that this two-phase segmentation method performs well with a wide variety of classes in the *MuscleFish* database. Therefore we propose that this novel method is potentially a good, generic solution to general audio segmentation problems. As a segmentation tool, this method outperforms typical CDS methods or model-based approaches reported in prior work. Moreover, by a judicious design of the feature set, this method can be tailored for other applications such as music structure analysis and summarization, speaker detection in speech analysis, as well as audio retrieval and classification for general sounds.

At the onset of the development of our segmentation method we outline three core requirements for an effective segmentation process namely, robustness, accuracy, and flexibility and practicality. Our novel two phase method meets these requirements.

Our experiments have demonstrated that the two phase method performs well with a wide variety of classes. The segmentation results are good for both the *MuscleFish* and the larger *VisualData* audio databases despite the fact that each database has a different ontology and different audio files. This suggests that the two phase method is not only robust but also flexible. Our two phase segmentation method is fully automated and functions unsupervised so that this method is a practical first step in an audio classification process.

By employing the maximum of amplitude, the first segmentation phase avoids the errors commonly occurring for short duration sounds when using existing segmentation methods. The second phase distinguishes segments where there is no silence between them. As a result our two phase method can accurately separate silence from sounds and detect abrupt changes in acoustic features. Indeed the experiments in Section 3.5 found that our novel two phase method is more accurate than existing methods.

Chapter 4
Audio Classification

In the audio visualization system, after segmenting the input audio file into homogeneous pieces (discussed in Chapter 3), each of the resultant audio clips is fed into a classification module to find the class it belongs to in the given database. The module reads an audio clip out of the segmentation module and performs a classification process to determine the most plausible matching class of audio in the database. At the same time, it detects whether or not the input audio clip is a new class that is not in the database and gives a reminder to users if this is the case.

The first two sections in this chapter introduce the key concepts of audio classification and review existing classification methods. Full details regarding the classification module and a new class detection method are presented in Section 4.3 and Section 4.4. Experiments that evaluate the performance of the classification method are reported. The integration of the classification method and the new-class-detection method is described. Methods for training the module to find suitable parameters and thresholds for the given audio data set are discussed in Section 4.5.

4.1 Introduction to the Classification Module

Classification can be understood as a process in which a previously unknown input signal is assigned to a class $C \in \{C_1, C_2, ..., C_n\}$ in an audio database. Such an assignment is made by establishing and applying a decision rule in a designed feature space to find a class (or an audio file in a class) that is deemed closest to the new audio [100].

In the proposed audio visualization system the classification module is used to process the resultant audio clips from the segmentation module and the content of each input audio clip is categorized. This is achieved by employing a pre-selected audio feature set

to enable the audio clip to be compared with the audio files in the database. The schema for the module is shown in Figure 4-1. The input for this module is an audio clip and the output is the class in the database that the input audio clip belongs to. The proposed audio visualization system assumes that when the class an audio file belongs to is known it can be visually represented using a template image or video for that class.

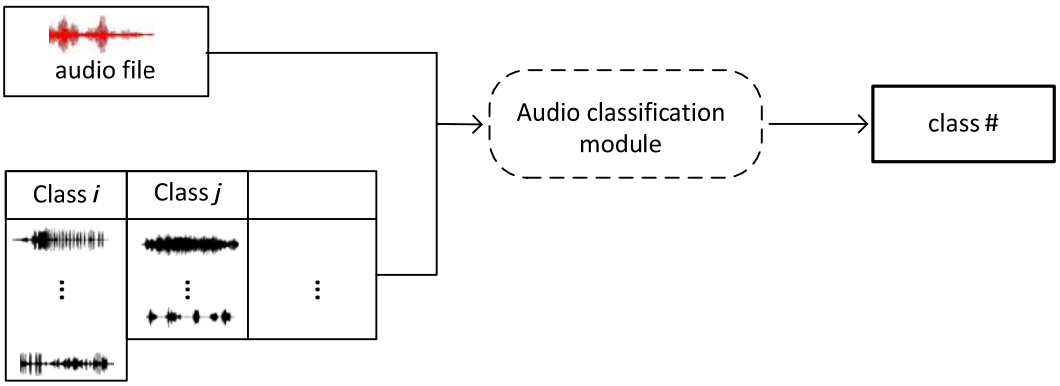

Figure 4-1: Scheme of classification module.

The correctness of the audio visualization system is determined by the template selection. The template image/video is correct when the classification is accurate. If the given audio clip is classified into an incorrect class, the template image for this class cannot represent the audio clip correctly and the result would be an incorrect audio visualization. The visualization aspect is discussed in more detail in Chapter 5 and Chapter 6.

The previously discussed segmentation module (Chapter 3) is an audio pre-processing step designed to enhance the performance of further analysis. Segmentation provides an input audio clip to the classification module that is homogeneous. The input audio clip is therefore a single sound and the input audio file does not contain overlapping sounds as these have been eliminated during the segmentation process.

Although many approaches have been developed for audio classification, existing research assumes the use of rich training datasets where it is unlikely for a query candidate to fall outside the ontology of the database. Dealing with a situation where the input sound is very different from those in the training dataset remains an unsolved problem.

4.2 Literature Review

Classification plays an important role in audio processing because it offers ways to efficiently navigate and provide control for the search of, and retrieval from, audio databases. Effectively managing numerous audio files is challenging and relies on the accurate classification and retrieval of audio files. A traditional text-based method is not feasible for audio management because an audio file is usually treated as an opaque collection of bytes with primitive fields attached [1]. Furthermore, because of the subjective nature of assigned keywords and the tedium of manual annotation, text-based methods are inefficient. Audio classification is an efficient tool for managing audio collections by clustering audio files automatically. Audio classification extracts the physical and perceptual features from an audio input and then uses these features to find the class in which it is most likely to fit. Audio classification aims to categorise the audio files automatically and accurately.

Audio classification has been studied using many approaches and can be used in different domains. For example, clustering the audio/visual data into events such as passing through doors and crossing the street [101]. In the music domain, there have been studies of artist classification [102] [103] and musical instrument detection and classification [104] [105] [106] [107] [108] [109] [110] [111]. Some approaches have concentrated on classifying specific audio files, such as classifying speech into local speech/crosstalk and local speech/ cross talk alone/ silence [112]. Kokoras and Pasquet evaluated classification methods for flute sounds [113]. In the music field there has also been some work to establish classification methods for the music genre [49] [114] [115] [116]. Various classifiers have been investigated as well as the performance of different audio feature sets. Certain audio features such as timbre and rhythm, used in the classification of music, are only meaningful for music and therefore are not applicable to the classification required of audio files in our system.

Some researchers have developed sophisticated approaches for the classification of more diverse sounds. For example, Burred and Lerch extended the classification results to speech, background noise and 13 musical genres [117]. All of these approaches demonstrate the importance of audio classification accuracy in an audio processing system.

Although the audio segmentation and audio query/retrieval methods used in audio processing may be capable of categorising audio inputs, they are not suitable for this proposed audio visualization system. Such techniques focus on finding the most similar audio files to the query/retrieval candidate and as such can be regarded as a kind of Nearest Neighbour classification method. We are looking for an exact classification that will allow us to select an appropriate template image in order to visually represent a sound.

Studies of audio classification and segmentation show that they are closely related to, and dependent on, each other. Segmentation can be a pre-process, as in the new two phase segmentation reported in Chapter 3, or a post-process of classification, such as the smoothing process proposed by Lu et al. [57]. A general framework for integrating, experimenting and evaluating different techniques of audio segmentation and classification has been proposed by Tzanetakis and Cook [61]. In addition, they also proposed a segmentation method based on feature change detection. The accuracy of classification achieved in their experiments on a large data set was reported to be about 90%. Some segmentation approaches can even be regarded as classification because audio segmentation also results in classes. Speaker recognition can be a segmentation issue but it can also be a classification problem if users group the collections by speakers. Another example can be found in Kimber and Wilcox [89], who separated meeting records into speech, silence, laughter and non-speech sounds. Although they were "segmenting audio recordings", the process was similar to classification as the resultant classes were defined according to their specific aims. But segmentation can only take the place of classification when the classes are limited in number and broad such as in the examples mentioned above. In our proposed audio visualization system, the segmentation process cannot take the place of the classification module because the classes in our system are not as broad and there are many more classes. Additionally, we wish to have an adaptive ontology that allows for the classification and visualization of files that belong to classes that do not exist in the database. The segmentation module in the system aims to separate the audio input into homogeneous clips but does not classify. Rather, it acts as a pre-process to the classification of the input audio file. The results of the segmentation need further analysis (classification) before it becomes possible to categorize the audio file contents.

Audio query and retrieval are two processes that are closely related to classification because they can be regarded as maximum likelihood classification. They are based on a similarity measurement that is similar to the Nearest Neighbour (NN) classifier. For the query candidate in our audio visualization system, there is not always an exactly matching audio file, but one or more relatively similar audio files to the query candidate. In an audio visualization system, audio retrieval and query are not employed because they cannot categorize the input audio file or find suitable image/video to represent it. Moreover, the audio/image or audio/video pairs in the same class of the audio database are generated by using the same template image/video and the specific audio features of the audio files. The visual features may be different but their content is the same if two images or videos belong to the same class. The resulting image or video of a query and retrieval cannot truly visually represent the input audio file unless its audio features are exactly the same. We propose that the most efficient way for image/video to represent an input audio file is to generate a unique image/video by employing image processing filters driven by the audio features on the template of the class.

A number of previous attempts have been made to classify audio files [118] [119]. In one sense the result of audio classification is subjective because the definition of the different classes is dependent on specific requirements and the subjective judgements of the person who interprets these requirements and constructs the definitions. Like segmentation, audio classification results can be quite different even for the same audio database. For example, music collections have been classified by genre [120] [121] [122], by mood or emotion [118] [123] [124] [125], or by instrumentation [126]. Experiments reported in Homburg et al. [127] also supported the subjectivity of audio classification. In the experiments, 24 different classification schemes were created by the users from various different personal viewpoints. From these experiments, even for the same audio files in a database, there were different database structures to group the audio files. To fit the specific structure of the database and its audio files, any classification system needs to be trained. When the audio files in a database are divided into different classes, their audio features are extracted and used as a training data set for the classification system. Training consists of a series of experiments in which each experiment tests a feature set and model using the training data set and gives a classification result. The model induced from the training set is tested for accuracy on

an independent old-out test dataset. The classification result from an experiment is compared with the training data set and results in a measure of the accuracy of this feature set and model combination. The best feature set and model offers the maximum accuracy. In other words, the classification system is trained by a process of finding the best feature set and model so that the classification result from the system can best fit the class definitions.

The sound to be classified may be not only from an audio database but other sources of sound. Some researchers have used news broadcasts as sound sources. Meinedo and Neto separated broadcast news into speech and non-speech, then further analyzed the speech and non-speech sounds separately [83]. The speech sounds were further divided by gender and the non-speech sounds were separated into noise and music. Then they further presented a system for speech/non-speech classification, speaker segmentation, speaker clustering, gender and background conditions classification for broadcast news [74]. The resultant classes were limited in part by the limited scope of the audio inputs. These approaches address the most well studied topic: speech/music classification. This type of classification is relatively straight forward and can achieve a high level of accuracy because the two classes are so broadly classified. There are many different approaches for speech/music classification [63] [128] [129]. Classification of speech, music and noise has also been reported [130] [131]. Similar approaches have been used to separate audio files into broad classes such as speech, music, environmental sound and silence [57] [132]. Especially when segmentation is performed beforehand, speech/music classification can be achieved with a high level of accuracy [132]. As the classes in these approaches are broadly defined the methods are limited in application and they are not suitable for databases containing a large number of narrow classes such as the *MuscleFish* database.

Overlapping, where multiple sounds occur in a single audio file simultaneously[54], brings difficulties for not only segmentation (see Section 3.1) but also classification. Many approaches have been proposed for pure audio classification using different feature sets based on different models. When an audio file is comprised of more than one kind of sound, the classification is more difficult and the result is not as accurate as for homogenous audio files. To achieve higher accuracy, some new classes other than pure acoustic classes have been defined. For example, Kiranyaz et al. [132] classified audio files into speech, music, fuzzy and silent. The fuzzy class can be regarded as a

class of audio files with overlap. The introduction of fuzzy regions was made to minimize misclassification and thus increase accuracy. Srinivasan et al. [133] detected and classified audio files that consisted of more than one sound, such as combinations of speech and music together with background sound. The audio files were categorized into mixed class types for example music with speech, noisy speech, etc. They achieved a classification accuracy of over 80%.

The accuracy of classification is affected by many different factors, such as the noise ratio of the input audio, whether the classes are broadly defined, the resources of the audio and classification pattern, etc. Appropriate feature set selection is important for accurate classification results. Pfeiffer et al. described basic features used in audio analysis [85]. One hundred and forty three features were studied by Li et al. [134] to determine their discrimination capability. Wold et al. [1] analyzed and compared statistical values (means, variances and autocorrelations) of four audio features for content-based audio indexing purposes. They found that not only audio features but also their statistical values can be used in audio processing. The audio features and their statistical values can form a large number of acoustic parameters in audio processing.

The selected feature set is always specific to each classifier and class type. Different audio features are selected according to specific requirements and patterns. For certain type of sounds some audio features are more important than others, for example ZCR is critical for speech determination [135]. Biatov demonstrated that features extracted from ZCR can achieve 76.6% accuracy for speech recognition [135]. The importance of ZCR in speech/music discrimination was also reported by Saunders [62] who presented a real-time classification for radio broadcasts using ZCR and short-time energy, which gave an accuracy of 95% to 96% [62]. Other audio features have been compared in different implementations. For example, Ravindran et al. proved that the feature Noise-Robust Auditory Feature (NRAF) is better than MFCC in classifying noisy sounds [136].

The influence of the number of features and the need for feature selection were discussed by Bocchieri and Wilpon [137]. A feature set that contains a high number of dimensions does not always lead to more accurate classification for unstructured sounds [138]. Burred and Lerch pointed out that more features do not necessarily result in more accurate classification especially when the training samples for each class are limited

[117]. Audio feature set selection is dependent on the experimental results. In other words, any classification system needs to be trained to find the most suitable feature set. Li performed experiments to compare various classification methods and feature sets and reported that the accuracy of classification depends on both audio feature set selection, and the classifier [98]. So the selection of classifier is also an important factor in improving the classification accuracy.

Many methods have been used in audio classification such as Hidden Markov Models (HMM) [139] [140] [141] [142], Nearest Feature Line (NFL) [98], k-Nearest Neighbour (k- NN) [57] [143], Gaussian Mixture Model (GMM) [102] [110] [112] [144], Bayes decision rules [145] [146], AdaBoost-based classifier [131] [147], Artificial Neural Networks [74] and Support Vector Machines (SVMs) [103] [105] [108] [114] [118] [119] [148] [149].

Some of the commonly used classifiers have been modified for specific purposes. Zhang and Kuo used a three-component GMM as classifier [130]. Joder, Essid and Richard used SVM with alignment kernels in audio classification [111]. Han, Gao, and Ji provided selective ensemble SVMs to improve the accuracy of speech/music/silence/speech with environment music (S-M class) discrimination [150]. In audio signal processing, a Bayesian hierarchical structure has been used for classifying music with pre-selected audio features [151] [152]. Bayesian models have been proposed for the classification of polyphonic music [153] [154]. A Gaussian Radial Basis function based Artificial Neural Network (ANN) [155] has also been applied to music classification. Bugatti et al. compared the performances of Bayesian Multi-layer Perceptron (MLP) for speech and music [156]. Comparative studies of the accuracy of various classification methods have also been undertaken [119]. Many studies have been undertaken that compare the accuracy between these classification patterns [98] [119] [157]. Homburg et al. compared the averaged accuracies from four different methods: "Random classifier, C4.5 decision trees, k-Nearest Neighbour and Naïve Bayes" [127]. Support vector machines (SVMs), together with a binary tree recognition strategy, a distance-from-boundary metric, and suitable feature combination selection can control the error rate within 10% [119]. All these studies provide information about potential classification processes for our proposed audio visualization system.

For the *MuscleFish* database the NFL method has been reported to achieve the best accuracy (90.22%) if a suitable feature combination is employed [98]. Accordingly, NFL has been selected and adapted as the classifier for our classification module.

There are other possibilities for the improvement of classification accuracy, such as combining two or more classifiers or using a hierarchically structured database instead of a direct structured database. Some models have been combined in an attempt to improve the accuracy of classification. A hybrid of SVM/HMM has been employed to obtain accurate classification [158]. Lopes, Lin and Singh used a scheme of indoor and outdoor audio classification [159] and found that a multi-stage classifier performs better than a single stage one. However, multi-stage classifiers require users to select different features for each stage and this increases the complexity of the system. The constitution and structure of the database can also affect the classification accuracy. In a number of studies the use of hierarchically structured databases has been reported to result in accurate classification [49] [117] [130] [142] [160]. Hierarchically structured databases have been found to be an improvement on direct structure databases. Burred and Lerch claimed that a hierarchical classification scheme is preferable because some features are more suitable than others for classification [117]. A generic scheme based on a hierarchical (and recursive in some places) structure was used by Herrera and Serra [160] to describe sounds at multiple levels of details. Although the hierarchically structured database may improve accuracy, it also shares the same drawback as using more than one classifier: it increases the difficulty of implementation because of the requirement for feature set selection for each level. As the accuracy of using NFL alone is satisfactory for our audio visualization purpose these other possibilities were not considered in our classification module.

As a result of the prior research presented above it can be concluded that a carefully selected classifier and a suitable feature set can be used to achieve satisfactory classification accuracy. But the existing applications for audio classification can only determine the most *likely* class in the database the query candidate may belong to. They are limited because they can not process audio files that do not belong in an existing class. Instead they classify these files incorrectly into the most similar class. This problem is addressed in our module by introducing a novel new-class-detection function. New class detection is discussed in the following sections together with the training and learning processes and the classifier employed in this module.

4.3 Training and Learning for the Classification Module

The training set is a basic concept in the area of artificial intelligence. A training set consists of an input vector and an output vector. Training is used to induce a model from a set of representative samples from the underlying data distribution. A model is induced that captures the separation between classes. This model is later used to classify new samples as they arrive.

In general, training is a function that takes one or more arguments and results in an output vector. The learning method's task is to run the system once with the input vectors as the arguments, calculating the output vector, comparing it with the answer vector and then changing the feature weights or feature selections in order to produce an output vector that is more likely to produce the answer vector the next time the system is simulated. The main characteristics required of a training set are that it:

1. defines the categories and the thresholds between different categories,
2. selects the features and their weights to get satisfactory classification results, and
3. defines the relationships between audio features and visual features so that the output (video/image) can be generated by comparing the differences and similarities between a new input and its most similar audio files.

This process of training is needed in the classification module. Figure 4-2 shows the framework for the classification module in our audio visualization system.

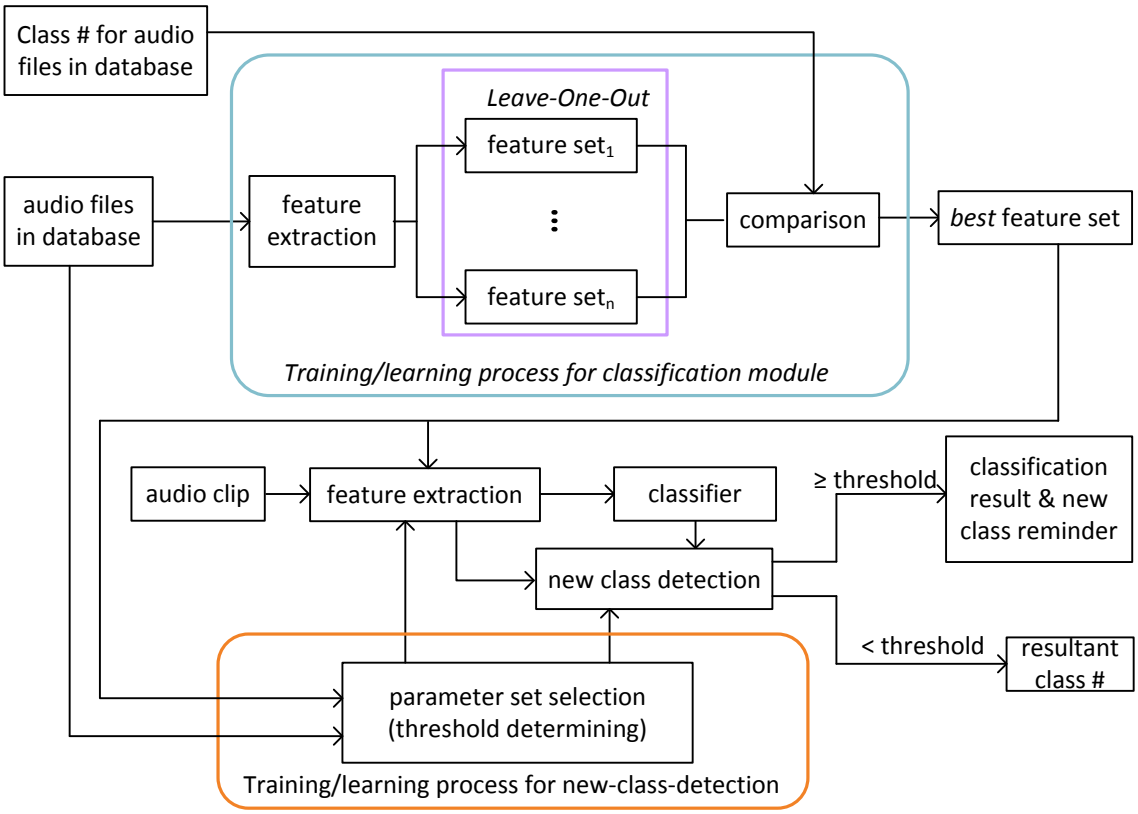

Figure 4-2: Framework for the classification module.

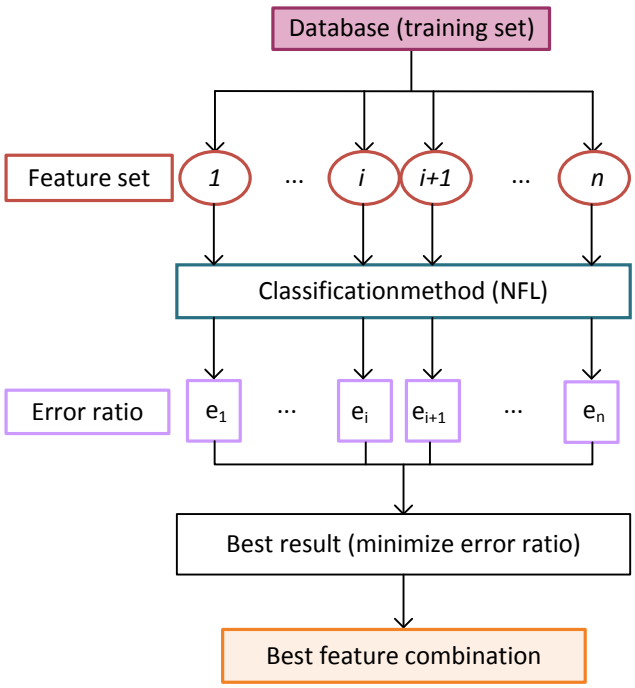

Figure 4-3: Training/Learning process for feature set selection.

Figure 4-3 shows the training/learning process for feature set selection in classification. All possible audio features for classification are extracted from each audio clip. With

the pre-defined feature combinations (which will be discussed in Section 4.4) and classifier (NFL method), the classification accuracies are calculated. These results are compared and the most accurate feature set is saved for the module.

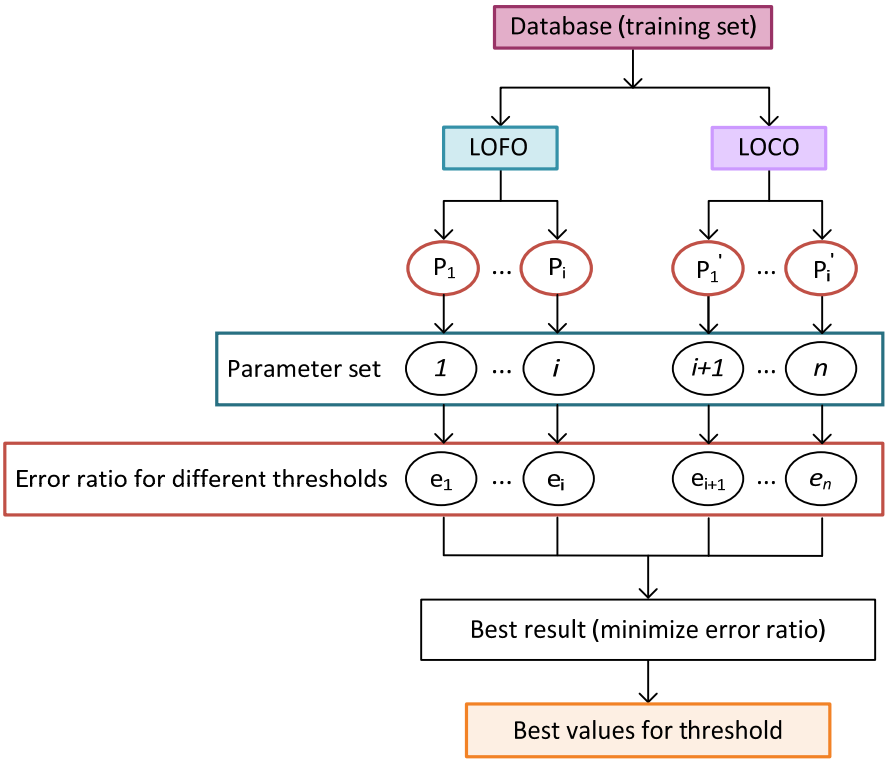

Figure 4-4: Training/Learning process for parameter selection and automatic threshold determination in new class detection.

Figure 4-4 shows the steps taken to train the new class detection function in the classification module. The steps use the selected feature set that resulted from the processes shown in Figure 4-3. The possible parameters (which will be discussed in Section 4.5) for the new-class-detection are extracted first. Then for all the possible parameter combinations, thresholds are determined by comparing their error ratios. The corresponding parameter set and thresholds of the minimum of the errors are selected for the new class detection function in the classification module.

When the database is initialized the above processes are run based on the training set (the original database). The selected audio feature set, parameter set and corresponding thresholds are stored and used for classification until the learning process is run again. The learning process is re-run when new audio files are added to the database.

As new audio files are added to the database it is possible that the initial settings for the module may not be suitable for the expanded dataset. The training process can be run again as learning processes to adjust the settings of the module. Thus the audio feature set for classification, parameter set and thresholds for new-class-detection in the classification module are adjusted automatically when new audio files are introduced to the database.

4.4 General Classification Method

The review of the literature on classification methods identified the Nearest Feature Line (NFL) method used by Li to be the one that produced the best classification result on the *MuscleFish* database [98]. Therefore, the classifier NFL method is adapted in the classification module and is describe in this section. The details of suitable feature set selection, based on a given training set, are provided after a description of the NFL method. The process developed and described in this chapter is general and can be used on any given training set. The *MuscleFish* database used for our segmentation testing is employed as a training set for the classification experiments. Details of the *MuscleFish* database can be found in Chapter 3.

4.4.1 NFL Method in Audio Classification

There are many different pattern classification applications that use the NFL method in domains outside of audio processing, such as face recognition [161], image retrieval [162], and prediction of protein locations [163]. When compared with four other classifiers (SVM, NN, 5-NN and NC) the NFL method produced the lowest error ratio when applied to the *MuscleFish* database [98] [119].

The NFL method can be seen as an extended classifier derived from the NN method. The feature line in the NFL algorithm is a linear interpolation and extrapolation of any two prototype points within a class. NFL requires more than one prototype sample per class. With suitable samples, a class can be represented in the feature space by the feature lines in it.

For a given query sample, the Euclidean distances from it to any feature line are calculated. From all these distances, the feature line of smallest value is called the

nearest feature line of the query sample. The NFL method defines the rule to classify the query sample: the query sample belongs to the class which contains its nearest feature line.

In the NFL method, the topological shape of the distribution for a class is more important than its sample numbers. When there are not many prototype samples in a class, the NFL is theoretically able to expand the representational capacity of the available points in the feature space, accounting for new conditions not represented by the original samples [164]. The NFL method has been reported to effectively improve the classification accuracy when the number of prototype samples per class is small [165].

Figure 4-5 illustrates the calculation of the distance from a query sample x to the NFL $\overline{x_1 x_2}$. The NFL distance can be calculated using two steps:

Step 1: Compute the projection point p on the line $\overline{x_1 x_2}$:

$$p = x_1 + \mu(x_2 - x_1) \qquad \text{where} \qquad \mu = \frac{(x - x_1)\cdot(x_2 - x_1)}{(x_2 - x_1)\cdot(x_2 - x_1)} \qquad \text{(E 4-1)}$$

Step 2: Calculate the Euclidean distance $\|xp\|$ in n-dimensional feature space. See Chapter 3, E3-2 for details of the calculation.

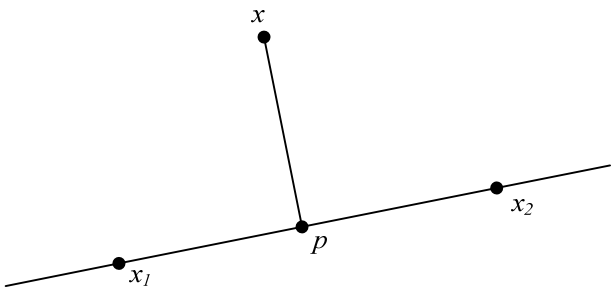

Figure 4-5: Generalization of two feature points x_1 and x_2 to the feature line $\overline{x_1 x_2}$. The feature point x of a query sample is projected onto the line as point p.

4.4.2 Cross-Validation for the Classifier and Feature Set Selection

The accuracy of any classification method is determined by feature set selection. As mentioned above, different audio features suit different audio categories. An improper

feature set will lead to inferior classification accuracies. To find a suitable feature set for a classification, based on a given training set, an experiment dependent process is used. In the classification module, a standard Leave-One-Out cross validation is used to both test the accuracy of the classification method and to find the best feature set for the training set.

Leave-One-Out cross validation uses a single sample from the original database as the validation data, and the remaining samples in the database are used as training data. A comparison of the predicted property value and the real value of the validation data comprises the assessment of the predictive result. This process is repeated until each sample in the database has been used once as validation data. The classifier and the feature set are evaluated by comparing all the results.

The computational cost of Leave-One-Out cross-validation is high if there are a large number of samples in the database. There are two reasons that Leave-One-Out cross-validation was utilized firstly it was only employed as a bench-mark and secondly it was used because it gives an almost unbiased estimator. An alternative to this method would be to employ a 10-fold cross validation method which is less computationally expensive. However to ensure the result is reliable the process must be repeated several times with different foldings thus raising the computational cost. For bench-marking purposes, the Leave-One-Out cross-validation method is the general accepted method.

4.4.3 Feature Set and Accuracy Experiments

The feature set that is used can have an important effect on classification accuracy. As can be seen from the experiments reported by Li [98], even for the same classification method, the results for different audio feature sets may be quite different. Therefore the selection of a suitable feature set is important for classification. The subsequent feature extraction process selects the audio feature set that will result in the most accurate classification based on the training set.

For certain types of sounds, some audio features are more important than others, for example ZCR for speech determination [135]. The best feature set also varies according to the specific training set. Different types of sounds require different combinations and weightings of the feature set in order to best match them.

The problem of audio feature selection is simpler if the types of sounds in the database are quite different and the number of classes is limited, for example having only a speech class and a music class in the database. Speech audio files all have the same deterministic audio features, which are quite different from those of music files, making the selection of features for the purpose of audio classification simpler. The training set *MuscleFish* presents unique difficulties for audio feature selection owing to the mixed classes and the disparate nature of the sound files.

The tested feature sets were adopted from Li's experiments [98]. He used Perc, CepsX as well as their combinations for feature sets. But, for better classification results, ZCR has been uniquely employed in combination with Perc in the experiments reported in this thesis.

Table 4-1: Number of correct classifications for 32 audio feature sets using *MuscleFish* database (n = 410).

Modified-Perc		**ZCR**	**CepsX**			
Pitch	**Pitch/Silence ratio**		**X=5**	**X=8**	**X=10**	**X=15**
✓	✓		356	363	364	357
✓	✓	✓	361	363	366	357
	✓		360	366	368	362
	✓	✓	360	368	367	366
✓			363	367	364	359
✓		✓	364	**370**	364	356
			363	364	366	362
		✓	362	366	365	361

In order to investigate both the efficiency and the accuracy of the classifications, variations on the Perc features set with and without ZCR and CepsX (where X=5, 8, 10 and 15) were tested in the experiments. The Modified-Perc feature set consists of a *core set* of features Spectral Power, Subband Power, Brightness and Bandwidth. This core is extended to form various Modified-Perc feature sets as indicated in the first two columns of Table 4-1. This forms the training process for the classification module based on the Leave-One-Out cross-validation method. The normalization of features was carried out as detailed by Li [98]. It should be noted that in the following general classification experiments the segmentation process is not used as a pre-classification step. This is because we wish to validate our system against the results reported by Li

using the standard evaluation database *MuscleFish*. Moreover, the segmentation module test files were generated by randomly combining *MuscleFish* files and therefore if segmentation was 100% correct we would have segments to classify that are the same as the audio files in *MuscleFish* with the silence periods removed.

Table 4-1 shows the results for 32 experiments using a Leave-One-Out test, with different feature sets, using the *MuscleFish* database as a training set.

The most accurate feature set was a combination of pitch + ZCR + Ceps8, and has been selected for the audio visualization system. It achieved a best accuracy at 90.2% (370/410). Our classification system always automatically selects the *best* feature set regardless of the size of the margin between it and the next best feature set.

In order to ensure that the classification method was generalisable to a larger database we evaluated its performance on a larger database *VisualData* which was designed as a dataset for the generation of audio file visualizations (Table 4-2).

For the *VisualData* audio files the experiments indicate that using pitch in the feature set does not increase the accuracy of classification. This is not entirely unexpected because there are many methods for extracting pitch and none of them are particularly accurate.

Table 4-2: Number of correct classifications for 32 audio feature sets using *VisualData* database (n = 611).

Modified-Perc		**ZCR**	**CepsX**			
Pitch	**Pitch/Silence ratio**		**X=5**	**X=8**	**X=10**	**X=15**
✓	✓		531	545	545	544
✓	✓	✓	529	542	543	545
	✓		532	546	**547**	**547**
	✓	✓	530	**547**	**547**	546
✓			528	541	544	538
✓		✓	528	538	544	541
			528	542	546	544
		✓	530	**547**	546	544

Three feature sets gave the same classification accuracy (86%) for the *VisualData* database. This accuracy is similar to that observed for *MuscleFish*. Of these the one most suitable for our classification method consists of the Modified-Perc *core feature set*, ZCR and Ceps8. There are two reasons for this. Firstly in our visualization system classification is preceded by the segmentation of the audio files. In the segmentation process all silence periods are removed therefore the pitch/silence ratio is not a contributing feature. Moreover because we wish the system to be computationally efficient we wish to select the best feature set with the least features in order to minimise the effort in calculating the feature vectors and to reduce the dimensions of the feature space.

4.5 Introducing a New Class Detection Method

Although the NFL method can classify audio files with satisfactory accuracy, it assumes the use of rich training datasets where it is unlikely for a query candidate to fall outside the ontology of the database. Dealing with a situation where the input sound is very different from those in the training dataset (a process known as novelty detection) remains an unsolved problem for audio.

In the literature, novelty detection is defined as the identification of new or unknown data that a machine learning system is not aware of during training. Novelty detection is considered to be one of the fundamental requirements of a good classification system. Most novelty detection methods employ statistical approaches that are driven by data distribution [166]. Novelty detection is most generally used in unsupervised learning systems [167] and in audio classification, models are usually supervised therefore to date there has been no reported attempts to incorporate novelty detection in an audio classification system.

This section presents an additional phase in our classification module in order to detect new class files. The next section explores a novel approach to the detection of new classes during classification. The parameter set selection and threshold determination for new-class-detection is also discussed and an evaluation established.

4.5.1 Overview

Existing audio classification methods assume high quality training sets that map cleanly to the target audio provided by the user. However, for more general use, the application should be able to process input sounds that are very different from those in the training set.

The new class detection step addresses this limitation, by flagging a particular sound input as potentially belonging to a new class. In new class detection, suitable parameters are extracted for the audio file first, then used to determine if the audio file can be classified into the existing ontology or whether it should form a new class in the database. The output of the classification module, that is comprised of the classification and new-class-detection processes, is the classification result which identifies the class in the database that the clip may belong to. If the file cannot be classified, a new class is required in the database so the classification module automatically builds a new class for the audio input and allows the user to define the new class name.

To be more flexible, the module should be able to not only predict whether a single query sample belongs to an existing class, or a new class, but also to give a reminder when the database has new audio files which may be able to be clustered together for a new class. This is achieved by introducing an *uncertainty* label besides the two results of "belonging to an existing class" and "belonging to a new class (not in the database)".

When the classification module identifies an *uncertain* result, it leaves the classification decision to the user. For an audio file, if all of its parameters are consistent it is easy to make a definitive classification. However, for those audio candidates that have conflicting parameter values, for example one parameter may indicate that the file belongs to an existing class but another may indicate a new class is required the *uncertain* criterion is necessary in order to ensure a more accurate classification result.

If the user classifies an uncertain file to an existing class the file is still marked as *uncertain*. When there are more audio files marked as *uncertain*, users can re-evaluate them and make further decisions about whether new classes should be built for these *uncertain* audio files. In general, by incorporating new-class-detection, this

classification module addresses limitations of current classification methods, and offers greater flexibility and robustness in general audio classification.

4.5.2 Parameters Used in New Class Detection

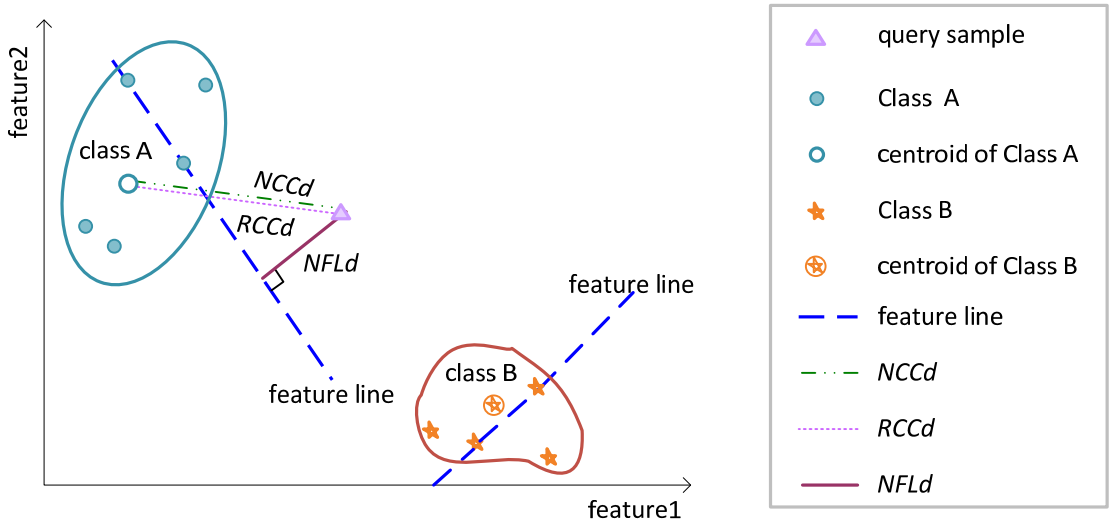

Figure 4-6: Parameters *NFLd*, *RCCd* and *NCCd* used in "new class detection" experiments.

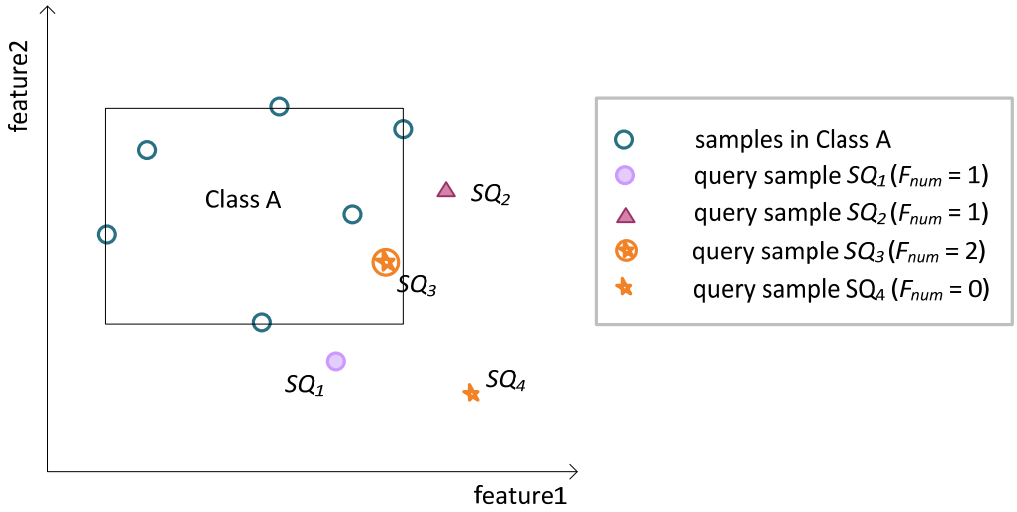

Figure 4-7: Parameter *Fnum* used in "new class detection" experiments.

Four novel parameters are introduced for new-class-detection in this section. Because the PercCeps8+ZCR feature set offers the best classification results, this feature set is used for the calculation of these four parameters (Figure 4-6 and Figure 4-7):

- NFL distance (*NFLd*), Euclidean distance from the query sample to its NFL.
- Real Center Class distance (*RCCd*), Euclidean distance between the query sample and the class centroid.

- Normalized Centroid Class distance (*NCCd*), with normalization carried out in the same manner as for PercCepsX [98].
- Class Feature Range Number (*Fnum*). When we test Audio *a* and determine its class A, regardless of the accuracy of the result, we find the maximum values and minimum values of each audio feature in the class. *Fnum* is defined as the number of features of *a* that are in the feature ranges of that class.

If an audio file $a \in A$ but is incorrectly classified as $a \in B$ then the parameters for new class detection are extracted from *B*. These parameters can be used individually or as a parameter set, depending on the training set.

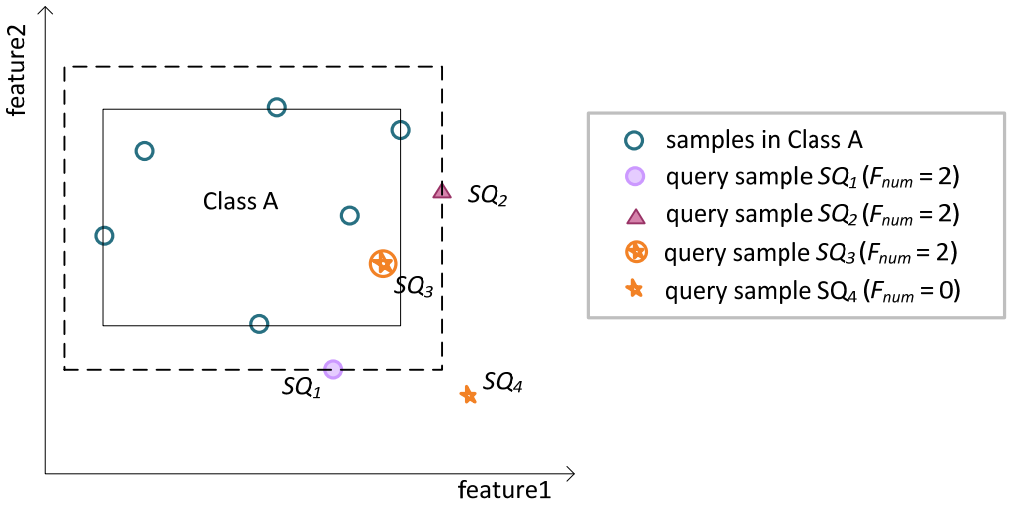

Figure 4-8: Expanded *Fnum*.

For any given database, the classification parameters may be adapted or fine-tuned to that dataset to give more accurate classification results. There are many ways to fine-tune these parameters. For example, when extracting the parameter *Fnum*, the feature range can be expanded to a bigger area, for example see Figure 4-8 which illustrates a 20% enlargement of the area. The corresponding values for *Fnum* are shown on the right of Figure 4-8. The extraction of the parameters is flexible and can be fine-tuned according to the training set. In order to illustrate the point Figure 4-6, Figure 4-7 and Figure 4-8 show a two-dimensional feature space formed using only two audio features. However, in new class detection the parameters are extracted in a high dimensional space. The number of dimensions is determined by the number of features used in the pre-defined feature set.

4.5.3 Evaluation of the "New Class Detection" Method

In order to evaluate the accuracy of new class detection and select the suitable parameter set for the system, two Leave-One-Out experiments were developed. The two experiments Leave-One-File-Out (LOFO) and Leave-One-Class-Out (LOCO) are detailed in Figure 4-9.

In LOFO, when a sound is used as the query, it is not used as a prototype so the prototype set consists of the entire database minus the query. 370 audio files were tested in the LOFO experiment. These files are those that can be correctly classified into an existing class, when classifying the file without new class detection (see Section 4.3). If one of these audio files is classified as a "new class", it is incorrect because we know that it actually belongs to an existing class in the database. In LOCO, the prototype set only includes the files that do not belong to the class of the query candidate. In this case if the query candidate is classified as belonging to an existing class in the database, the classification result is incorrect. All 410 audio files in the *MuscleFish* database were tested in the LOCO experiment.

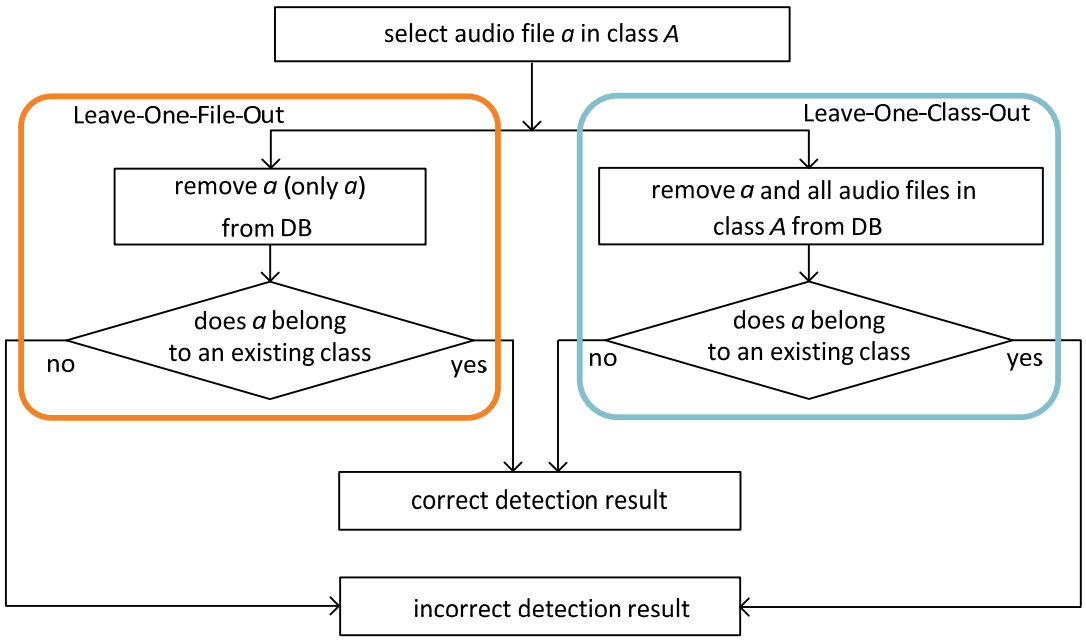

Figure 4-9: LOFO and LOCO experiments.

The parameter set and threshold for each parameter were also determined during the LOFO and LOCO experiments. The values for the four parameters (*NFLd*, *RCCd*, *NCCd*, and *Fnum*) were calculated for each query candidate. For *NFLd*, *RCCd* and

NCCd if the resulting values were less than a threshold, the query candidate had a higher likelihood of belonging to the calculated class. And if the value of *Fnum* was greater than its threshold, the query candidate was close to the result class. So we define that for a query candidate *a*, if any of the parameters in a parameter set do not pass the threshold test, *a* is classified as a candidate for a "new class". Because the correct classification and new class definition results are known for these files, the best parameter set and thresholds are the combination where it achieves the most accurate new class detection result. The 3 steps for threshold determination, employing the database as a training set, are as follows:

1. Calculate parameter vectors,
2. Determine the threshold range for each parameter,
3. Calculate the best threshold value.

For the LOFO and LOCO experiments four parameters are extracted (*NFLd*, *RCCd*, *NCCd*, and *Fnum*). For each parameter *i* a vector v_i is calculated. The threshold for parameter *i* must be greater than the minimum value and less than the maximum value (the range of valid threshold values) for the vector v_i. This range is divided equally into 100 possible threshold values. Classification (with new class detection) is then undertaken using LOFO and LOCO using this set of 100 thresholds. The threshold that resulted in the lowest error ratio is then used as the threshold for parameter *i* for the database.

The results of a number of experiments using different combinations of parameters are given in Table 4-3. The ticks indicate which parameter or parameters have been used in the experiment. From these experiments, the corresponding parameter set for the most accurate result is found. The most accurate result is considered to be the best parameter set for the classification module. As has been found for the feature set for classification, different training sets will give different parameter sets.

The results in Table 4-3 show that for the classification of audio files in the *MuscleFish* database, *NFLd* is the most effective parameter of the four parameters tested. *NFLd* used on its own has an accuracy that is comparable to the most accurate parameter set *NFLd* and *NCCd* (see Exp. 1 and Exp. 5, Table 4-3).

Table 4-3: Number of files incorrectly classified for the LOFO and LOCO experiments and the *MuscleFish* database using various parameter sets.

Exp #	Parameter set				LOFO n = 370	LOCO n = 410	Total n = 780
	NFLd	***NCCd***	***RCCd***	***Fnum***			
1	✓				132	63	**195**
2		✓			223	69	292
3			✓		283	34	317
4				✓	111	101	212
5	✓	✓			131	63	**194**
6	✓		✓		131	67	198
7	✓			✓	132	63	**195**
8		✓	✓		228	64	292
9		✓		✓	111	101	212
10			✓	✓	111	101	212
11	✓	✓	✓		132	63	**195**
12	✓	✓		✓	132	63	**195**
13	✓		✓	✓	132	63	**195**
14		✓	✓	✓	100	101	212
15	✓	✓	✓	✓	132	63	**195**

The new class detection experiments were duplicated using the *VisualData* database to ensure the flexibility and robustness of the approach. In the LOFO experiment, 547 audio files were tested. These are the files which were correctly classified using the general classification method. In LOCO, all the 611 audio files were tested. The results are given in Table 4-4. Again *NFLd* proved to be the best parameter for classification with total percentage accuracy, for the LOCO + LOFO, experiments of 79%. It is worth noting that there is a minor improvement (4%) in the *best* classification accuracy for the *VisualData* database when compared to the *MuscleFish* database classification results.

The parameter F*num* does not contribute to the accuracy of the classification for either database. When it is used individually it performs substantially worse than *NFLd* (see Exp. 4 and Exp. 1 respectively in Table 4-3 and Table 4-4). The accuracy is not improved when it is combined with parameters or parameter combinations (see for example, Exp. 1 and Exp. 7 in Table 4-3 and Table 4-4).

Table 4-4: Number of files incorrectly classified using various parameter sets with the *VisualData* database.

Exp #	Parameter set				LOFO n = 547	LOCO n = 611	Total n = 1158
	NFLd	***NCCd***	***RCCd***	***Fnum***			
1	✓				123	124	**247**
2		✓			260	129	389
3			✓		197	251	448
4				✓	161	166	327
5	✓	✓			123	124	247
6	✓		✓		123	122	**245**
7	✓			✓	123	124	247
8		✓	✓		266	125	389
9		✓		✓	161	166	327
10			✓	✓	161	166	327
11	✓	✓	✓		124	122	246
12	✓	✓		✓	123	124	247
13	✓		✓	✓	123	122	245
14		✓	✓	✓	161	166	327
15	✓	✓	✓	✓	124	122	246

NCCd and *RCCd* do not produce accurate classification results in the LOFO experiments however they make a minor contribution to accuracy when combined with *NFLd* (see Exp. 11 in Table 4-3 and Table 4-4).

4.5.4 The *Uncertain* Criterion: "New Class Detection" Experiments

In previous experiments, the audio files were separated into 2 categories: the files were either classified into an existing class in the database or labeled as a new class. For an audio file, if all of its parameters are consistent it is easy to make a definitive classification. However, some audio files have conflicting parameter values. One parameter may indicate that the file belongs to an existing class, but another may indicate a new class is required. These audio files are difficult to classify and so an *uncertain* criterion was introduced. Figure 4-10 shows the results of testing audio classification using the *uncertain* criterion and Table 4-4 provides the results of testing the accuracy of audio classification using the *uncertain* criterion.

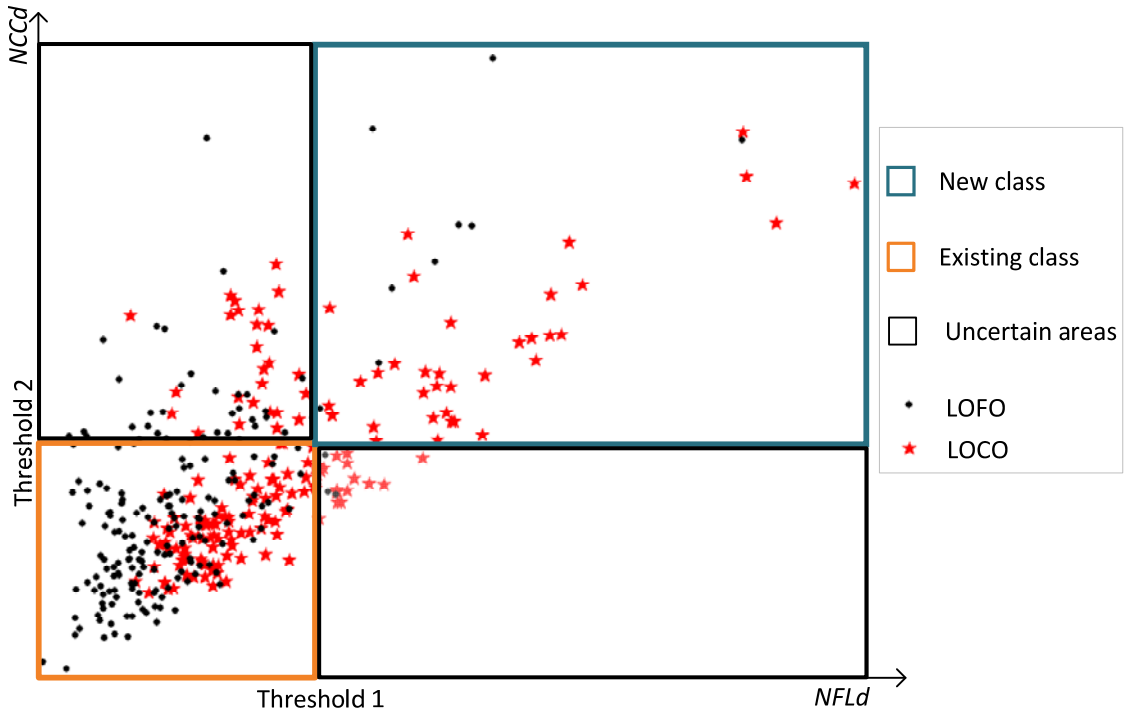

Figure 4-10: Audio file classification using the *uncertain* criterion.

From the experimental results provided in Table 4-3 and Table 4-4, it can be seen that the most effective combination of parameters is *NFLd* + *NCCd*, these form the parameter set that is employed in the following work. Figure 4-10 shows how the parameter space of *NFLd* and *NCCd* is separated into four regions, *existing class*, *new class* and *uncertain* areas. For a query candidate, when both parameters are less than their corresponding thresholds, it is classified to an existing class (the lower left quadrant area). When both the parameters are greater than their threshold (the upper right quadrant), the query candidate belongs to a new class. The rest of the query candidates, which have one parameter greater than threshold and another less than threshold, are marked as *uncertain*.

Reduction of the number of incorrect files can lead to an increase in the number of files classified as *uncertain*. The threshold is therefore optimized to minimize the number of incorrect files while simultaneously maximizing the number of correct files. With the optimized thresholds, 370 audio files were tested (which are correctly classified without new class detection) in LOFO and all the 410 audio files in LOCO (Table 4-5).

Table 4-5: Results for LOFO and LOCO experiments using the *uncertain* criterion and the *MuscleFish* database (*NFLd + NCCd*).

Incorrectly classified (40 audio files)	**LOFO**	**Existing**		8
		New		22
		Uncertain		10
Correctly classified (370 audio files)	**LOFO**	**Existing**	**Correct**	131
		New	**Incorrect**	117
		Uncertain		122
410 audio files	**LOCO**	**Existing**	**Incorrect**	40
		New	**Correct**	318
		Uncertain		52

In the two experiments, the number of incorrectly classified audio files is 157 (117 in LOFO and 40 in LOCO) of the 780 query candidates. This shows that our system can deal with audio files that do not belong to existing classes of the training set, and the correct classification ratio with new class detection is ≈ 80%. This is a significant result as it is comparable with the results of classification accuracy *without* new class detection using other classifiers such as 5-NN [98]. Furthermore, the system allows users to fine-tune any threshold to obtain desired results. For example, by increasing the thresholds to maximize the existing class region.

The 40 incorrectly classified audio files were also tested (top row, Table 4-5). Only 8 files of the 40 are classified into an existing class directly. The system prompts the users to consider classifying the remaining 32 audio files manually because they are marked as *uncertain*. Our system largely reduces the risk of misclassification. In a system without new class detection, for example that used by Li [98], a LOCO experiment would result in all 410 files being incorrectly classified. In our system only (40/410) 10% are incorrectly classified.

This experiment was repeated using the *VisualData* database (Table 4-6). The number of incorrectly classified audio files for both the LOFO and the LOCO experiments was 185 files. The number for audio files that gave an uncertain result was 276. In the context of the experiment an uncertain classification is considered a correct classification therefore the total error is 16%. This means that the accuracy of the

classification using new class detection and the uncertain criterion is 84% for the *VisualData* database and 80% for the *MuscleFish* database suggesting that this method is suitable for general sound classification.

Table 4-6: Results for LOFO and LOCO experiments using the *uncertain* criterion and the *VisualData* database (*NFLd + NCCd*).

Incorrectly classified (64 audio files)	**LOFO**	**Existing**		12
		New		41
		Uncertain		11
Correctly classified (547 audio files)	**LOFO**	**Existing**	**Correct**	290
		New	**Incorrect**	94
		Uncertain		163
611 audio files	**LOCO**	**Existing**	**Incorrect**	91
		New	**Correct**	407
		Uncertain		113

There are other possible parameters for new class detection such as the distance of the test sample to its nearest point in the resultant class. This is similar to finding a threshold of NN classifier. Other classifiers can also provide other potential parameters to detect whether or not the input belongs to a new class. As the classifier in this module is NFL, these possible parameters were not considered for use in the module. In other implementations, they could be worthy of investigation.

4.6 Hierarchical Classification

Some research has shown that the use of a hierarchically structured database leads to a more accurate classification [49] [117] [130] [142] [160]. The *VisualData* database has been designed with that in mind. Currently this database is somewhat limited as there are only two levels within the hierarchy of the database (Figure 4-11). However, it seemed reasonable to undertake some preliminary tests to evaluate the affect of a hierarchical structure on our audio classification method.

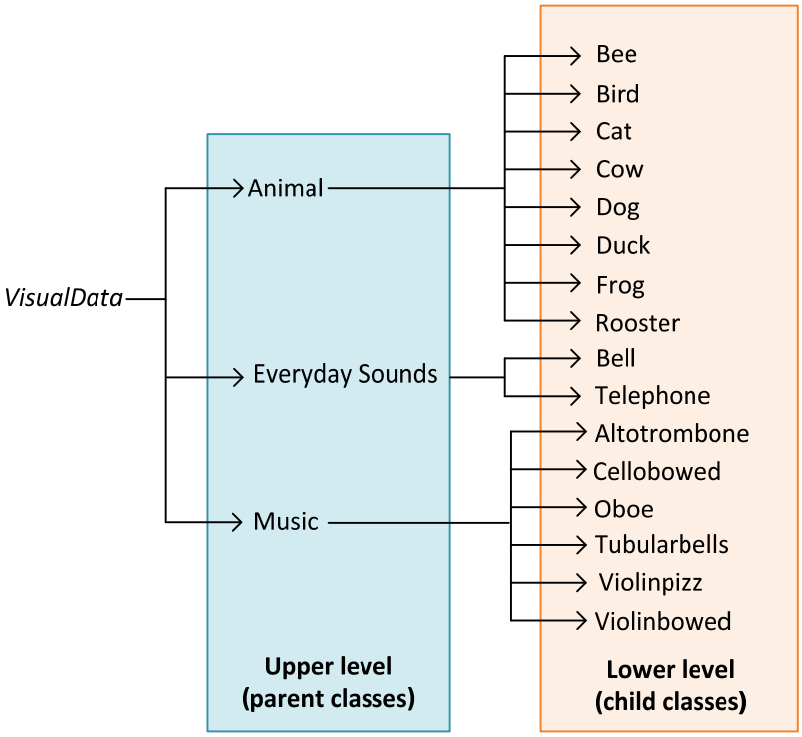

Figure 4-11: *VisualData* ontology overview.

In this experiment an audio file is first classified into a parent class in the upper level. Then it is reclassified within the child classes of the identified parent class to determine if the file actually should belong to an existing child class. The general classification method uses the NFL method and various feature sets. Because silence is removed in the segmentation method, the pitch-to-silence ratio feature would not contribute to the accuracy of the feature sets as the pitch-to-silence ratio would be a constant value. Therefore the pitch-to-silence feature is always omitted from the classification feature sets used.

The Leave-One-Out cross-validation method is again employed to test the classification accuracy. If an audio file is incorrectly classified at the parent level, it can never be correctly classified at a lower level. For this experiment, only the audio files correctly classified at the parent level are processed in the second, subclass classification step. Therefore, in this experiment, the number of incorrectly classified audio files is considered to be the number of incorrectly classified audio files at the parent level (Table 4-7) and at the child level (Table 4-8).

Table 4-7: Number of incorrectly classified files for LOFO, using parent level classes (n = 611).

Modified-Perc	ZCR	CepsX			
Pitch		X=5	X=8	X=10	X=15
✓		19	14	13	11
✓	✓	21	14	13	12
		18	11	11	10
	✓	19	14	11	11

Table 4-8: Number of incorrectly classified files for LOFO, using child level classes.

Modified-Perc	ZCR	CepsX			
Pitch		X=5	X=8	X=10	X=15
✓		66	60	54	53
✓	✓	60	58	57	55
		68	62	55	55
	✓	61	59	55	52

Table 4-7 shows the inaccuracy for the first pass classification at the parent level and Table 4-8 shows the inaccuracy for the second pass classification at the child level.

Using a hierarchical database, in this case, only improved the classification by one file. Five hundred and forty eight files were correctly classified, using the *best* feature set and the hierarchical classification method (Table 4-9). With standard classification, using the *best* feature set, 547 files were classified correctly (Table 4-2). However, a non-hierarchical classification is clearly more computationally efficient than a hierarchical approach. Firstly because the non-hierarchical classification occurs in a single pass. Secondly because in a hierarchical classification method the parent class is broad and contains many files and the feature set calculations increase exponentially with the number of files.

Table 4-9: Total number of correctly classified files after two passes.

Modified-Perc	ZCR	CepsX			
Pitch		X=5	X=8	X=10	X=15
✓		526	537	544	547
✓	✓	530	539	541	544
		525	538	545	546
	✓	531	538	545	**548**

4.7 Classification of Mixed Sound Audio Files

All the experiments detailed previously in this chapter classify the files in the *MuscleFish* and *VisualData* (Section 5.3) databases. These audio files consist of sounds from a single class. However when audio files are to be visualized the input audio file may consist of any number of sounds from various classes as was the case with the files that were used to test the two phase segmentation module in Chapter 3. The experiments described in this section use mixed sound audio files which must undergo segmentation prior to classification. Not only is this segmentation step essential to the audio visualization process but it should also lead to more accurate classification of mixed sounds audio files. In a standard classification approach the sounds in an input file are treated as if they are one sound belonging to one class and are classified using the mean values of their combined audio features. Naturally this standard approach is most likely to lead to a misclassification of the file.

In our system the mixed sounds in an audio input are separated into audio clips using the two-phase segmentation process (Chapter 3). Each resultant audio clip contains a single sound or group of similar sounds. Then these audio clips are independently classified.

For this set of experiments, if an input audio file is a mixed sound file, the audio file is correctly classified when and only when all the sounds in it are correctly classified. Misclassification of any one of the sounds leads to an incorrect result for the audio file as a whole. The accuracy of this test depends on both the accuracy of the segmentation and the classification phases. The audio files used for testing in the following experiments are the audio files that were generated using the *VisualData* files (see Chapter 3, Section 3.4 for test file generation discussion) for evaluating the two phase segmentation method.

The first experiment used two phase segmentation followed by general classification of the audio segments. For classification the feature set used was Modified-Perceps8 with ZCR with the *NFL* method. Of the 1000 mixed audio files 63.4% of audio files were classified correctly (Table 4-10). It is important to note that an incorrect classification may mean that only one of the segments is incorrectly classified for that audio and that

the majority of the segments extracted from the input audio file were correctly classified.

Table 4-10: Classification accuracy for mixed sound audio files.

	Correct classification	Incorrect classification	Total
Mixed audio files	634 (63.4%)	366 (36.6%)	1000
Number of audio segments	2936 (83.8%)	569 (16.2%)	3505

Figure 4-12 shows a mixed audio file and its segments. In total there are 20 segments extracted using the two-phase segmentation method. Of those 20 segments all but one are classified correctly. While this is a good result for classification, it is not so good for visualization purposes. When an audio file is visualized a template image will be selected from the database that matches each audio segment's class. Therefore even one incorrect classification will result in a visualization which has an anomalous image (see Chapter 5 for a more detailed discussion of the selection of template images and subsequent visualization).

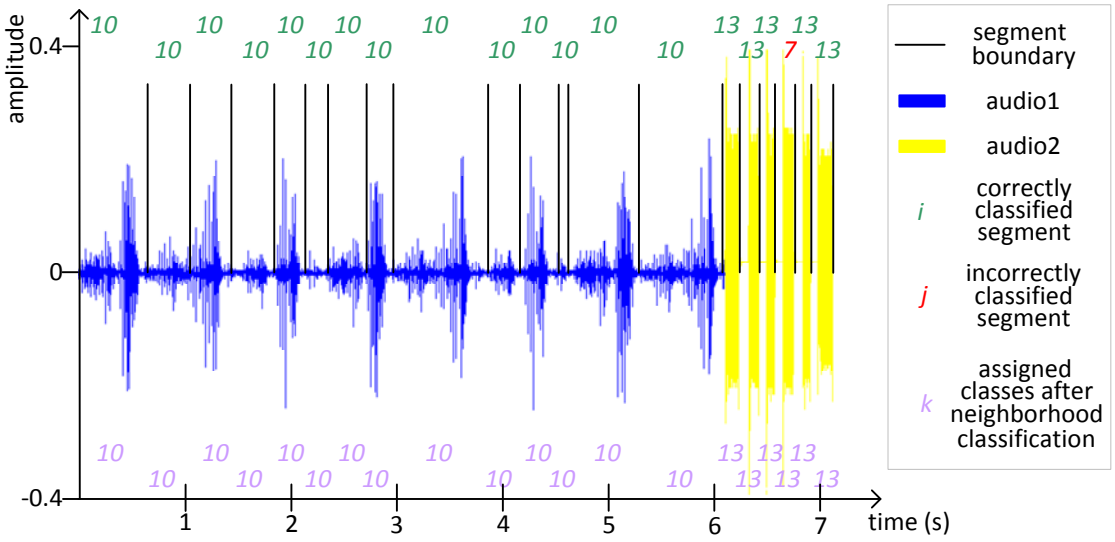

Figure 4-12: An example of an audio file that is incorrectly classified.

It maybe that if new class detection was used for classification of the segments the incorrectly classified segment would be marked as *uncertain*. Therefore the 1000 audio files were again segmented and classified but this time using the new class detection method with the *uncertain* criterion (Subsection 4.5.4). If this method is used 93% of the incorrectly classified segments are now correctly classified as *uncertain*. However a new problem was discovered in using this new class detection approach. Some

segments that were correctly classified and assigned a definitive class were now assigned to that class but also marked as *uncertain.* This resulted in 40% of the previously correctly classified audio files, with all segments correctly classified, now posing problems for the visualization phase because a segment was marked as *uncertain*. User input is required to indicate whether the audio segment belongs to a class or is a new class of audio file and also to indicate how to handle that segment in the visualization. Another alternative is to remove *uncertain* segments from the visualization of the audio by default or to increase the transparency of visualization elements to indicate the uncertainty of each element. Neither of these options is entirely satisfactory. However, as the database evolves the *uncertain* audio files will eventually form a cluster and therefore a class and the visualization will become more representative of the mixed audio files.

There are a number of ways in which this process could be improved. These are discussed below.

One possibility is to alter the *NFLd* new class threshold. Experiments were undertaken with the automatically determined threshold increased by 25%. This threshold is based on the pre-segmented mixed audio file feature set. Of the incorrectly classified audio files 83% of them were marked as *uncertain*. For the correctly classified audio files only 16% now have segments marked as *uncertain*. This would make user intervention to indicate the classification of an *uncertain* file more reasonable because the number of segments marked as *uncertain* is significantly lowered. Ideally an approach that minimizes user input should be employed if possible.

Another option would be to perform a general classification first without new class detection. A nearest neighbour check may then be performed to check for anomalous classification results. For example in Figure 4-12 the third to last segment is classified to class number seven but this appears to be inconsistent when its nearest neighbours are considered. If the surrounding segments are all classified to 13 and the anomalous segment duration is short, then it seems likely that the segment has been misclassified. This segment could be automatically detected and then classified using the new class detection method. If the segment is marked as *uncertain* it is removed from the visualization of the audio files. An alternative to removing the segment would be to perform a neighbourhood classification. In neighbourhood classification a single

segment on either side of the anomalous segment and the anomalous segment would be combined. The classification could then be performed on the mean audio features for the combined segments. If the segment is a terminal segment in the audio file the neighbourhood might be considered to be the nearest segment.

The third option would be to perform general classification, then check the entire audio segment sequence using a nearest neighbour check to identify anomalous classification results. If an anomalous segment is found, neighbourhood classification is performed to determine whether or not the classification is correct.

All of these possible solutions should mitigate the problem of having a large number of segments being classified as *uncertain* and should reduce the level of user intervention required.

To test the neighbourhood classification strategy three typical mixed sound audio files were selected from the incorrectly classified audio files. First the audio files were segmented using the standard two phase segmentation method. Then these segments were classified using the general classification method. The consistency of each segment was checked by comparing it with the class assigned to each of its nearest neighbours. When a potentially incorrectly classified segment was detected it was classified using the neighbourhood classification method. For all three files a correct classification was achieved. In the case of the mixed audio file shown in Figure 4-13 the second segment was identified as an anomalous classification. This segment was combined with the first and third segments in the files and the mean features used to reclassify the second segment. Neighbourhood classification resulted in the second segment being correctly classified. The nearest neighbour check followed by a neighbourhood classification seemed a promising approach therefore the remaining incorrectly classified mixed sound audio files were processed using this approach.

A segment is defined as anomalous if its duration is less than 20% of the total duration of the input audio file and the segments on both sides belong to the same class. For a terminal segment (the last segment in the file or the first), if its nearest two neighbour segments belong to the same class, or if the terminal segment is shorter than 1/3 of its nearest neighbour segment, it is also identified as an anomalous segment. When an anomalous segment is found it is reclassified using the neighbourhood classification

method. We found that only 29.3% of the incorrectly classified audio files are classified correctly. When examining the segments and the original audio file it became evident that the files that do classify well using the neighbourhood classification method have an incorrect terminal segment (either at the start or end of the audio file). The others which are positioned between two audio files are more difficult to identify as an anomalous segment if the previous segment and the next segment are from different classes. To identify an anomalous sound there needs to be a pattern of consistency in classification of the neighbouring files. For example (Figure 4-13), where the fifth to last segment in reality belongs to class 12, is classified to class 10 but is not identified as an anomalous segment using our method. If we examine the previous segment its class is 10 and the next segment after it is given as 12. So such a segment is not marked as anomalous because when scanning the segments the fifth to last segment appears to be a natural boundary between two sounds.

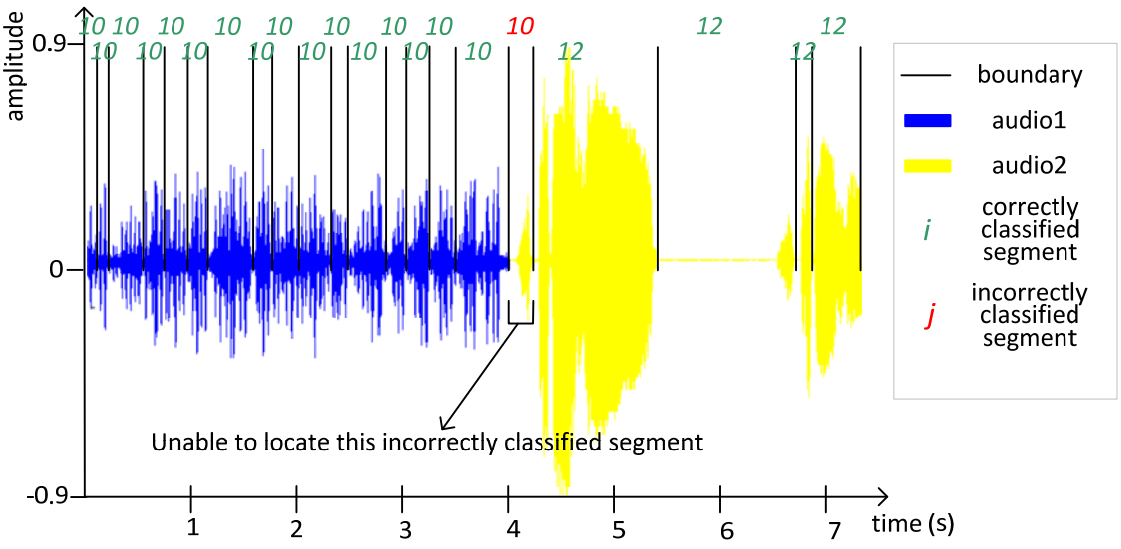

Figure 4-13: Incorrectly classified middle segment that is not identified as anomalous.

Of the 1000 mixed audio files tested we noted that 3% of the audio files have more than one segment incorrectly classified and 1% have only one segment incorrectly classified. We deducted that the reason that these segments are incorrectly classified is due to the segmentation of the audio file. Because the threshold of silence is determined by the mean of two sounds half of the noise period is segmented to the first sound, and the rest to the second sound. When classification takes place the noise may then end up being treated as "signal" if the signal-to-noise ratio (SNR) is low. Another reason may be that some of the incorrectly classified segments are of a relatively long duration and

although they are terminal segments they have comparable durations to their neighbours and are not recognized as anomalous segments.

In a further experiment the incorrectly classified audio files were checked for segment classification consistency and then any identified segments were processed using new class detection classification. In this case the automatically detected threshold for new class detection is used and 93% of these identified segments are reported as *uncertain*. The user may then choose how to handle these uncertain segments for the purposes of visualization. When the classification segments that are marked as uncertain in new class detection are considered to be correctly classified the overall classification accuracy achieved using this method is 98%. We argue that this method provides an accuracy of classification that will lead to accurate audio visualization.

4.8 Chapter Summary

This chapter describes an improved audio classification method for general sounds. It differs from the previously reported methods in two ways. Firstly a new feature set has been developed that has a unique combination of audio features. Secondly our method can handle the classification of audio files that belong to a new class.

Previous methods have calculated the feature set as an average of all the segments in an input audio file, and classified without new class detection so they are not suitable for heterogeneous audio files. For the purposes of audio visualization, a pre-classification step has been introduced that segments the audio files prior to classification. We have found that although the segmentation and the classification methods developed in this work outperform previously reported methods even when used in combination they are not sufficient for accurate audio visualizations. And indeed even segmentation combined with classification using new class detection was not sufficient to produce reliable audio visualizations. Therefore we added an additional processing step whereby the first classification results are examined automatically for inconsistencies and any anomalous segments are reclassified using the new-class-detection method with an *uncertain* criterion. This method proved to be very accurate and was sufficient for audio visualization. The next two chapters describe two possible methods for audio visualization namely, time mosaics and video textures respectively.

Chapter 5
Visualization Approach 1- Time Mosaics

This chapter presents a time mosaic approach to the visualization of audio files. The visualization is referred to as a time mosaic because it is comprised of several image tiles that match the time sequences of the audio clips in the audio files that they represent.

In the proposed audio visualization system, the generation of time mosaics takes place in the third module: the time mosaic generation module. Time mosaic generation occurs after the segmentation of the audio files (Chapter 3) and the subsequent classification of the segments or audio clips (Chapter 4).

The resultant time mosaic image provides a visual representation of the component sounds (audio clips) in the input audio file and simultaneously illustrates the corresponding audio properties which have been subjected to image processing filters driven by audio characteristics such as power, pitch and signal-to-noise ratio. Where the input audio file is comprised of a single sound it is represented by a single image that has been subjected to filtering. Heterogeneous audio files are represented as a seamless mosaic image along a time axis where each component image in the mosaic maps directly to a discovered component sound.

The first section of this chapter introduces the framework of the module. In the second section, the existing research on the representation of audio features by visual features and the techniques needed to generate time mosaics is reviewed. Then a specific audio database, called *VisualData*, is introduced to fully test the module. After this the processing of template images based on the audio properties of visual-audio feature relationships, which will be used to generate image tiles for the resultant final mosaic

image, are discussed. The method for combining the resulting image tiles is given in Section 5.6. Some resultant images are presented to demonstrate how this module and the audio visualization system work. It is then proposed that time mosaics can be adapted for other uses, such as audio database navigation, and this is discussed at the end of this chapter.

5.1 The Framework for Time Mosaic Generation

Figure 5-1 illustrates the framework of the time mosaic generation module. The input audio clips for this module are sounds (segments/clips) identified by the segmentation module. Their corresponding template images are selected from the pre-built audio/image database on the basis of the result of the classification for the segments. For each audio clip, the audio features are extracted and then used to select image filters that are applied to the corresponding template image. This results in an image tile for each sound in an audio file. Then all the image tiles are merged together to produce the resultant image or time mosaic.

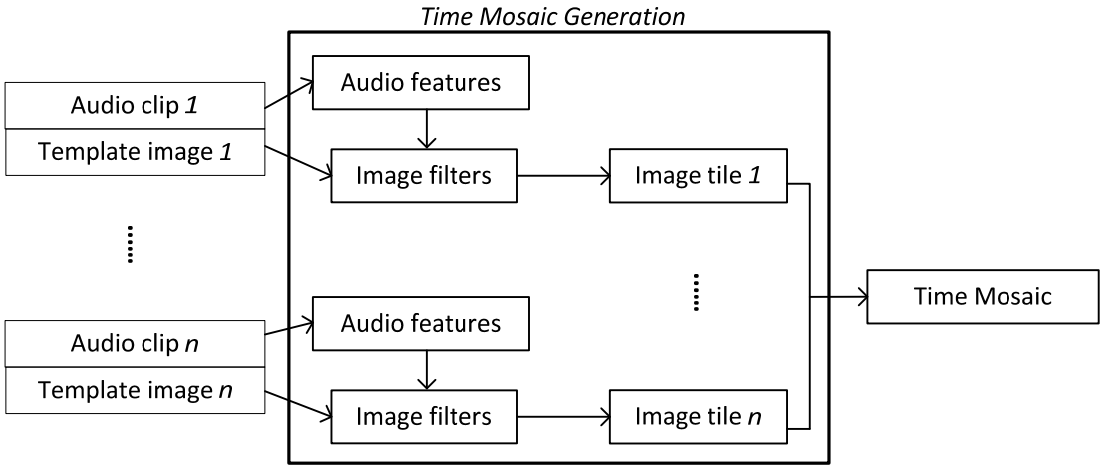

Figure 5-1: Framework for the time mosaic generation.

Figure 5-2 gives an example of a time mosaic that arises as a result of simply putting the template image tiles in the order of the corresponding audio clips in an input audio file. The input audio file is the example file presented in Chapter 3: Figure 3-8 which contains five audio clips. After classification, it is determined that the middle three audio clips (see the audio signal below the dog image in Figure 5-2) belong to the same class so they are combined automatically and represented by a single image tile. However, in this simple form of time mosaics the audio signal under each image tile

must be scaled to the width of the image tile and therefore does not illustrate the duration of the corresponding audio clip. Furthermore, the discontinuity between two template image tiles is obvious and the overall resultant image does not appear to be a single image.

Figure 5-2: Image tiles placed according to their corresponding time sequences.

Figure 5-3: Blended image mosaic generated with image tiles of the same size.

Figure 5-3 shows another visualization of the same input audio file as in Figure 5-2. This result is generated using the blended image mosaic method, reported in Section 5.5, which produces seamless image tiles. Compared with Figure 5-2, the image shown in Figure 5-3 is more suitable for presenting a visual representation of an audio file using a single image. The blending indicates that the visualization is of a single mixed sounds audio file rather than a series of individual distinct sound audio files. Moreover, because the seamless merging eliminates the edges between image tiles it is suitable for representing a silence period between two adjacent sounds, and accommodates images

of different sizes. The size of the images represents the different duration of the sounds (for example see Figure 5-7 and Figure 5-10).

Figure 5-3 depicts a typical result mosaic generated by this module. The intent is to use the mosaics as an aide in the navigation of a database of audio files. In addition, audio features are simultaneously represented through image features that have been filtered so that they can represent audio features (e.g. noise ratio and power). As a new form of audio representation, time mosaics offer a novel way to browse and edit audio files.

5.2 Literature Review: Image Processing

Audio visualization is a cross-disciplinary topic which is closely related to audio processing, visual perception, cognitive psychology and image processing. The related work reviewed here is arranged in the following order with commonly used visual and acoustic features introduced first. The association of colour and sound is discussed next as colour is the most widely used visual feature for acoustic representation. This is followed by a discussion of the literature on the audio-visual mapping relationships used in our image-tile generation process. Finally, research that investigates techniques used for merging images is reviewed.

For general sounds, there are several acoustic features such as amplitude, duration, rhythm, fundamental frequency, harmonic, modulation and noise to be considered. Some of them, such as loudness and pitch, can be more easily perceived by humans than others. On the other hand some audio features cannot be readily recognized by most people so it would be meaningless to try to represent them for non-professional users. There are, however, many visual features available for the visual representation of audio features, for example, colour, contrast, size, shape, texture, hue, transparency, blur, and diffuseness.

Associations between acoustic and visual characteristics have been extensively studied. The work reported support, the possibility of visualizing audio files using images. We propose that these well accepted acoustic and visual character associations can be adapted for our audio visualization method. Existing studies show that some visual and auditory features can inspire similar cognitive experiences [168]. The most widely studied topic is the relationship between colour and music and has been discussed by

many artists over the centuries. Newton [169] reported the first scientific attempt at colour and music association when he described the Colour Music Wheel. Wu and Li proposed several methods for generating music from image features [168]. Similar work has been done by Mao et al [170], who proposed mapping relationships between painting and music and who developed rules for the production of music from images. The mapping of audio/image features has been used for other applications, e.g. pitch was mapped to colour by Caivano [171], and Giannakis and Smith associated saturation with loudness [4]. Giannakis et al. [172] surveyed the auditory-visual associations used to generate music using visual features and Giannakis developed a method for evaluating auditory visual mappings [173].

Margounakis and Politis proposed a method for converting images to music [174]. In their method the relationship between perceptual aspects of images and sounds was illustrated, and this provided clues for the audio visualization system proposed in this thesis. They proposed a mapping matrix of chromatic synthesis which is shown in Figure 5-4.

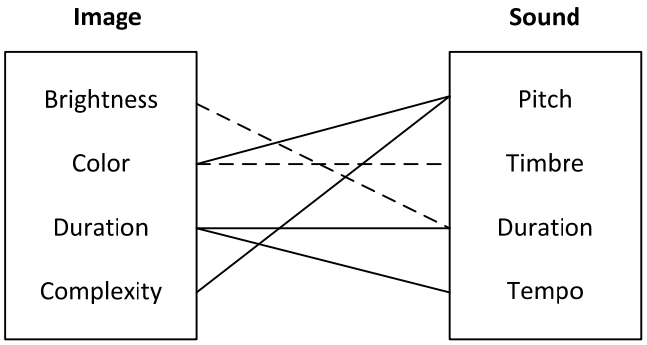

Figure 5-4: The mapping matrix of chromatic synthesis [174].

An approach called "chromatic bricks" was invented to describe how an image can be mapped to sound by Politis et al. [9]. "Chromatic bricks" mapped colour to melody by connecting musical entities and optical wavelengths (RGB values).

From studies of the association between colour and music it is now known that the feeling associated with a given colour may vary from one individual to another. Ghinea and Chen pointed that "*auditory-visual associations are primarily based on subjective judgements rather than on empirical evidence*" [175]. Kaya and Epps illustrated the relationship between colour and emotion using experiments performed with college

students [16]. They tested the responses of ninety-eight college students to five principle hues, five intermediate hues and three achromatic colours. They found that their subjects had different attitudes towards the same colour. For example, blue elicited relaxation, calmness, happiness, comfort, peace and hope for some people. But for others, it was associated with depression; it made them feel blue. Feelings associated with other visual features can also be person dependent. This makes sound visualization difficult. A possible solution to this problem is to use a legend in an audio visualization system, so that viewers have a guide to the meaning the image's features are intended to convey. Customization by users to adapt preferred mappings is another solution.

Besides the audio-visual feature relationship selection, there are other issues related to the representation of audio files by images. Because an audio file may contain more than one sound, image merging needs to be considered to allow the representation of audio files with mixed sounds. When the images for all the audio clips are processed, using their corresponding acoustic features, they need to be merged together to generate the final image result.

Research about merging one or more individual images and image segments together to form a new single image is directly related to image and video compositing [176]. There are a number of methods for carrying out image based synthesis. Patch-based approaches are commonly used in texture synthesis algorithms [177] [178] [179]. Patch-based approaches work by copying small regions of pixels from a source texture so that when these regions are stitched together they give the same visual impression as the original texture [176]. The output texture need not be the same shape or size as the source.

An image mosaic is another method used to merge different image tiles. Image/video mosaics and texture images are all variations of the image merging concept. There are various types of mosaics that could be adapted to the proposed visualization purpose, such as automatically generating video mosaics [180], jigsaw image mosaics [181] and Escherization [182]. The image tiles used in jigsaw image mosaics [181] can be of any shape, unlike traditional photo-mosaics that are limited to rectangular tiles. A process for making an Escher-like tiling of the plane from tiles that resemble a user-supplied goal shape has been reported by Kaplan and Salesin [182]. This has been extended to more complicated cases in dihedral Escherization [183]. The types of mosaics discussed

above could be useful for audio visualizations because they allow the seamless merging of image tiles that are either regular or irregular in shape. The seamless merging of image tiles in our time mosaics is relatively straightforward because the image tiles in the proposed audio database query system are always of regular, rectangular shape.

There are two fundamentally different techniques for creating mosaics. The first relies on detecting features in the scene and matching them across multiple frames. This approach works well on man-made objects that contain many straight edges and corners, but fails in natural scenes because the features are not so clearly defined. The second approach directly minimises the discrepancies in intensities between pairs of images. This approach is well suited to natural scenes because it does not require any easily identifiable features and is statistically optimal [184].

Poisson image editing [185] is an alternative algorithm that could be used for visualizing the mixed sound audio files in this project. It has been used for seamless editing of image regions by pasting a selected region from a source image onto a target image. It blends the colours of edges between the selected region and its target region to eliminate discontinuities and to generate a seamless image result. Similar results can be obtained using other algorithms such as multi-resolution splines [186]. But, by comparison with the multi-resolution spline method, Poisson image editing uses an elegant mathematical formulation and is easier to use [187]. Although Jia et al. [187] pointed out that the Poisson image editing method may generate unnatural blurring at the edges, this is not important in time mosaic generation because we believe that blurred edges should not lead to any misunderstanding of the visualization result. For computing efficiency and quality of blending, Poisson image editing was employed in the time mosaic generation module presented in this chapter.

There is another possible way in which an image tile can be placed into a bigger image and that is to regard an image tile as an element texture of a big image. Generally, image textures can be regarded as the subset of all images that exhibit spatial stationarity, meaning that the image is composed of repeating visual patterns. A considerable amount of research has been applied to the generation of texture images from real image exemplars using optimization [188], graph cuts [179], parallel processing [189], simplicial complexes [190] and simple Markov models [191]. De Bonet's approach [192] [193] was extended to multiple input samples to image-varying

textures by Bar-Joseph et al. [194]. A texture synthesis approach has not been used in this thesis, though it may be the subject of future work for audio visualization.

5.3 Audio/Image Database for Testing

A well-built audio-image database is a key component for testing this module and is also required for other components of the audio visualization system. The *MuscleFish* database [1] [98], which has been used as a standard non-musical sound database for the experiments reported in the segmentation and classification modules, is not entirely suitable for testing the time mosaic generation module. This is because some of the audio files in *MuscleFish* are too broadly classified. In the "animals" class, for example, there are nine sounds that belong to six different kinds of animal namely: cats (kittens), chickens, dogs, ducks (and geese), horses and pigs. Such a set cannot be meaningfully represented by a single archetypical template image. Therefore in order to fully test this time mosaic generation module and the whole audio visualization system, a new database named *VisualData* (Table 5-1) that is based on the *MuscleFish* database, has been built. The new database, *VisualData*, inherits some classes from *MuscleFish*. Those classes in *MuscleFish* that are not suitable for time mosaic testing have been replaced by new classes in *VisualData*. The audio files in the new-built classes have been downloaded from websites with freely available files [195] and [52]. For example, the audio files in the class "Bird" in *VisualData* are sounds from birds other than duck and rooster. These new audio files have been converted into the same format as the audio files in *MuscleFish*: 8-bit ISDN μ-law encoding with a sampling frequency of 8000Hz.

The *VisualData* training set has 611 audio files classified at a more suitable granularity for audio visualization (time mosaics). A comparison of the ontological structures of the databases *MuscleFish* and *VisualData*, with N_c being the number of sounds in each class, is given in Table 5-1. To fully test the visualization of audio files using *VisualData*, the accuracies of segmentation module and classification module in the proposed audio visualization system were also tested using the *VisualData* database (see Chapters 3 and 4). Table 5-2 summarizes findings about the accuracies of segmentation and classification of databases *VisualData* and *MuscleFish* from the segmentation module and classification module. The segmentation accuracy was tested using the 2-phase Euclidean method with feature set PercCeps8. The classification accuracy was

tested using the feature set PercCeps8+ZCR and the NFL method. The details of the feature set and experiments that measure the accuracy of segmentation and classification for *MuscleFish* and *VisualData* have been reported in Chapters 3 and 4.

Table 5-1: Comparison of *MuscleFish* and *VisualData* Ontological Structures.

MuscleFish Classes	N_c
Alto trombone	13
Bells	7
Cello (bowed)	47
Oboe	32
Telephone	17
Tubular Bells	20
Violin (bowed)	45
Violin (pizz)	40
Animals	9
Crowds	4
Female Voice	35
Laughter	7
Machines	11
Male Voice	17
Percussion	99
Water	7
Total	410

VisualData Classes	N_c
Alto trombone	13
Bells	14
Cello (bowed)	47
Oboe	32
Telephone	66
Tubular Bells	20
Violin (bowed)	45
Violin (pizz)	40
Bee	36
Bird	16
Cat	46
Cow	79
Dog	17
Duck	18
Frog	99
Rooster	23
Total	611

Table 5-2: Accuracies for the segmentation and classification for *MuscleFish* and *VisualData*.

	VisualData	*MuscleFish*
Segmentation accuracy	92.8%	96.6%
Classification accuracy	89.5%	90.2%

By comparison with *MuscleFish*, *VisualData* contains an additional 210 audio files (49% more audio files) and this increases the difficulty of segmentation and classification. In the case of both of these databases there is considerable overlap between classes, and therefore the more audio files there are in the database the less accurate the segmentation and classification processes are expected to be.

The experiments in this chapter test not only the time mosaic generation module but the whole audio visualization system. Any audio input is processed in the segmentation module, the classification module and the time mosaic generation module. The result for any audio input is therefore a time mosaic image.

Each class in the database has a template image which is manually selected once for the database and can represent the content of all the audio files in the class. When the audio features are extracted and the audio-visual feature relationships are defined, for any audio file, an image based on its template image is generated using image processing filters driven by the audio features of the file. The details of how this is done, and how the audio-visual feature relationships can be adjusted if required, are discussed in Section 5.4. Sets of audio-image pairs are stored in the database, together with their classification and training information. This comprises a library that associates sounds with archetypal images.

The experiments reported in the following sections were based on the *VisualData* database and were undertaken to test the time mosaic generation module and to investigate the contribution that the time mosaic approach could make to our audio visualization system.

5.4 Image Tile Generation

As discussed in Section 5.2, it is possible to connect audio features and visual features and this is fundamental for the generation of image tiles. The goal of image tile generation in time mosaics is to generate tiles for sounds and at the same time convey information about as many useful audio features as possible. This is achieved by using audio features to drive the image filters. The function described in this section is the generation of an image tile for a given audio clip using the template image for the audio clips class.

The audio features and their corresponding visual features need to be selected when the system is built. The mappings of the features to filters are set as defaults but can be altered by the user if required. As mentioned in the literature review, there are more than 50 commonly used audio features. Some of these features such as pitch and power are easily perceived by humans while others which have no metaphorical analogues in

human audio perception, such as bandwidth and MFCCs are strictly used for audio analysis. Therefore only subsets of these audio features are mapped to image filtering operations.

The image processing operations are employed to produce distinctive images for a new given audio clip for the purpose of visualizing the audio input. At the same time, the function of image tile generation in this module should generate images for individual audio files within the same class for navigation purposes as this is another potential usage for the audio visualization system.

5.5 Interpretable Audio Features

There are many audio features that determine the way humans perceive sound. In the proposed audio visualization system, the audio features Power, Pitch and Signal-to-Noise Ratio (SNR) are used for visualization so that viewers can grasp the character of the audio clip. A discussion of each of the audio features is as follows:

- **Power** is perhaps the simplest feature of an audio clip. It is related to the perceived loudness of a sound. The higher the volume (amplitude), the higher the power of the sound, and the louder a human will understand it to be.
- **Pitch** is another feature that listeners can readily comprehend. In the field of music analysis, pitch perception is often thought of in two dimensions, pitch height and pitch chroma [196]. But for casual users, it may be sufficient to know that an audio clip with a high pitch sounds high and shrill, while a low pitched sound is deep and soft [197]. Acoustics research indicates that pure sinusoids sound sharp and very low frequencies sound flat, compared to a purely logarithmic relationship [198]. As the aim is to let viewers grasp the difference between pitches at a glance, the system needs only to visualize differences within classes. The precise values of the pitches are not required.
- The **Signal-to-Noise Ratio (SNR)** can also be perceived by the human ear. It can be defined as the clarity of the signal. For example, poorly recorded audio, or audio with significant background noise, will have a low signal-to-noise ratio.

Correspondingly, there are visual features that may be mapped to the described audio features, such as image brightness and contrast, the depth of colour and the "noise" (or degree of blur) in an image.

5.6 Audio-Visual Feature Mapping

When considering an audio feature to visual feature mapping it is worth considering the dimensionality of those features [199]. It could be argued that an *n*-dimensional audio feature should be mapped to an *n*-dimensional visual feature and that this might improve user comprehension of the sound visualization. For example timbre can be a three dimensional audio feature it might make sense to map each dimension of timber to a colour channel. One abstract audio visual approach to timber, known as TimbreGrams [8] actually uses this specific feature mapping. The multi-dimensional properties of audio and visual features also make a multi-dimensional representation possible. However the use of combined multidimensional features in three dimensional space might result in visual cues that conflict with each other and confuse the user. In this work we elected to focus on two dimensional images and video for audio visualization. Moreover due to the novel and exploratory nature of this work we opted to use a simple set of one dimensional feature mappings. These default audio-visual feature mappings are as follows (Figure 5-5):

1. Brightness to Pitch
2. Colour Depth to Power
3. Size to Duration
4. Image Noise to SNR

It would be simple to extend the image and audio feature sets and to provide a means for the user to explicitly over-ride the default mapping with his or her own preferences. This customisation option seems sensible due to the subjective nature of human perception of audio and visual features.

The top row of Figure 5-5 illustrates how the audio SNR affects the visual noise ratio. The blue signal under the row of images denotes the pure audio signal for each sound clip. The red wrapping around the signal denotes noise added to the clear audio signals with varying degrees of white Gaussian noise added to the image. From left to right the

audio files generated have increasing background noise with the far left image representing the pure starting audio file. Accordingly, the images become fuzzy to represent this change in the signal-to-noise ratio of the input audio file.

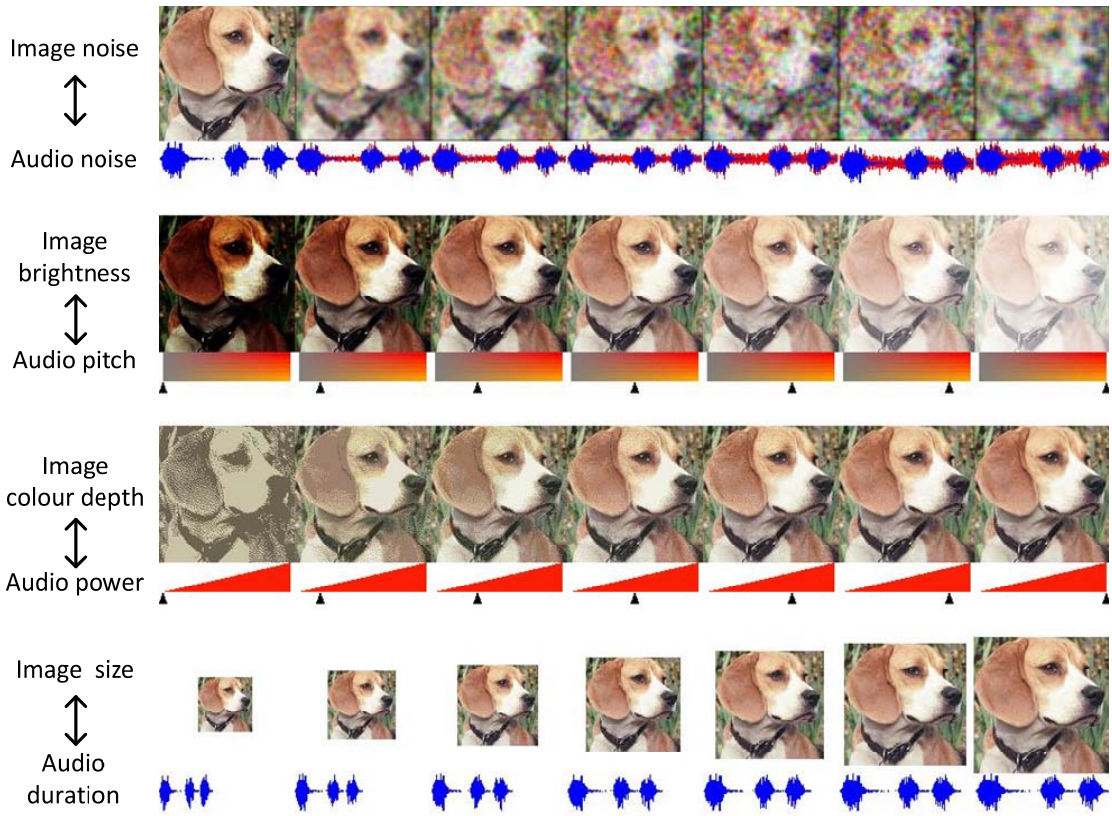

Figure 5-5: Legend for visualized features in the audio visualization system.

Brightness values for the template image are also linearly scaled to fit the pitch of an input audio as is shown in the second row of Figure 5-5. The template image represents an audio clip with the average pitch of the class. The audio sounds high and shrill when the pitch is high, so the image is set to be brighter for such sounds and darker for lower pitched ones. The pitch comparison is made strictly within classes. For example, the pitch of a bees' buzz is always higher than an oboe's sound, but it would not convey additional information to set the brightness of bees to always be very high and the brightness of oboes to be always very dark. Therefore, the visual representation of pitch is made to be relative within the same class so that, for example, the viewer can interpret a bright picture of a dog as being a relatively high pitched dog bark. Of course a legend is also required here to provide a sense of the level of brightness and thus the pitch of the audio.

Audio power refers to the degree of "loudness" or the volume of a sound. The higher the power value, the more easily it can be sensed. The colour depth of the image represents the power of the audio clip so that as a sound's power fades, so does the colour in the image that represents it. The third row of Figure 5-5 shows this relationship.

In addition to the above relationships, the size of an image is used to represent the duration of the audio clip as shown in the bottom row of Figure 5-5.

To generate an image tile, a template image is processed in four steps based on the four properties discussed above.

5.7 Image Tile Generation

When given an audio clip and its class in the database, the pre-selected audio features are extracted and then compared with the feature values of the files in the same class in the database. The generation of image tiles is processed on the template image according to the legend discussed above, which is the same for audio files in the pre-built database and for the new audio input.

Suppose the template image *I* represents class *A*, which audio clip *a* belongs to. Given an audio clip A_i in the class *A* its audio features are denoted as $\vec{v}_i^{\,A}$. The audio features for the given audio clip are $\vec{v}^{\,a}$. The steps taken to generate an image I_{output} for the given audio *a* are detailed as follows.

The audio features used in the proposed audio visualization system are $\vec{v}^{\,a}_{pitch}$, $\vec{v}^{\,a}_{power}$, $\vec{v}^{\,a}_{SNR}$ and its duration. Signal-to-noise ratio is a concept defined as the ratio of a signal's power to the noise power corrupting the signal. It compares the level of a desired signal (such as music) to the level of background noise. The higher the ratio, the less obtrusive the background noise is. Here its inverse (Noise-signal-ratio) $\vec{v}^{\,a}_{NSR}$ is used to represent the noise level of the background to audio.

Calculating the brightness scale parameter

The brightness of the template image is used as the standard to represent the median pitch of a class. When the pitch of the given audio clip is higher than the median pitch of the class, the resulting image is brighter than the template image. For the audio clip with pitch lower than the median pitch of the class, the resulting image is darker than the template. The brightness is calculated as follows:

$$brightness = \frac{0.5 * (\vec{v}^{\,a}_{pitch} - median\,\vec{v}^{\,A}_{pitch})}{max((max\,\vec{v}^{\,A}_{pitch} - median\,\vec{v}^{\,A}_{pitch}), (median\,\vec{v}^{\,A}_{pitch} - min\,\vec{v}^{\,A}_{pitch}))} \qquad \text{(E 5-1)}$$

Where $\vec{v}^{\,a}_{pitch}$ is the pitch value of audio clip *a;* and $median\,\vec{v}^{\,A}_{pitch}$, $max\,\vec{v}^{\,A}_{pitch}$, and $min\,\vec{v}^{\,A}_{pitch}$, are the median, maximum and minimum values of the pitch values in class A. The constant value 0.5 was determined experimentally and was found to give the best visual results. A value of greater than 0.5 results in a washed out image where the brightness level is too high and therefore the brightness may occlude the other visual audio property cues that are encoded in the image. A value of less than 0.5 results in a limited range of brightness, thus visualizations of similar audio in the same class are not distinguishable. The adjusted image is then calculated using a processing function available in MATLAB ***brighten(h, brightness)***, where *h* is the template image and brightness determines the intensities in the colour map of *h*.

Calculating the ColourDepth parameter

The ColourDepth scale is calculated in 3 stages:

1. The audio clip is normalized to set the maximum amplitude value equal to 1, producing *Normal*(a). Then the power value is calculated as $max\,\vec{v}^{\,a}_{power}$.

2. *Normal(a)* is scaled to *0.01×Normal(a)* to calculate the power value as $min\,\vec{v}^{\,A}_{power}$. This value is chosen empirically because low amplitude is not easily heard by the human ear. When the sound has an amplitude of *0.01×Normal(a)* or a lesser amplitude, we use minimum colour to represent it.

3. Colour depth is defined as the number of colours that are used in the colour map of an image. The colour depth ranges from a maximum of 2^8 to a minimum of 2^2. The colour depth scale is calculated as:

$$ColourDepth = 2^{2+6\times\frac{\vec{v}^{a}_{power}-min\vec{v}^{a}_{power}}{max\vec{v}^{a}_{power}-min\vec{v}^{a}_{power}}} \quad \text{(E 5-2)}$$

A new colour map, I_{output}, is generated by using minimum variance quantization on *ColourDepth* as described in Subsection 5.2.2.

Representing the audio noise-to-signal ratio

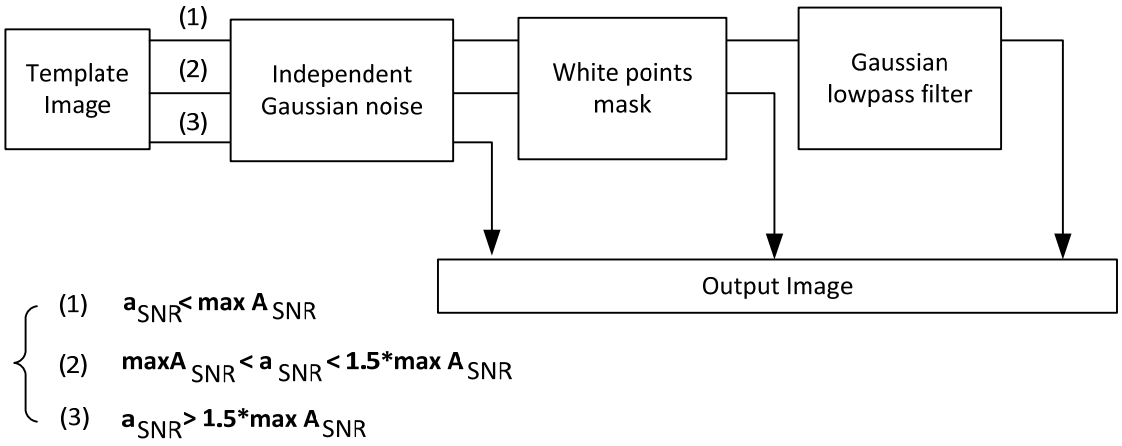

Figure 5-6: Algorithm to represent the audio noise-to-signal ratio.

Three algorithms were designed here for representing different levels of Noise-to-Signal Ratios (NSR) from very clean to very noisy, as shown in Figure 5-6. Note that $\vec{v}^{a}_{NSR}$ is the NSR for audio clip *a*. To represent the NSR of a given audio clip, not only is independent Gaussian noise added to the template image, but also white point noise for very noisy audio files. Furthermore, a degree of blurring is introduced for extremely noisy audio files. Each of the three filtering processes is calculated as follows:

1. Independent white Gaussian noise is added to I_{output}. If $\vec{v}^{a}_{NSR}$ is less than the maximum NSR $max\,\vec{v}^{A}_{NSR}$ in the class it belongs to, it is the only form of visual "noise" that is added. The white Gaussian noise has a mean of zero, with a variance of:

$$NoiseScale = 0.15 \times \frac{\vec{v}^{a}_{NSR}}{max\ \vec{v}^{a}_{NSR}} \quad \text{(E 5-3)}$$

2. After adding independent white Gaussian noise, white points are added when $\vec{v}^{a}_{NSR}$ is between *1* to *1.5* times $max\,\vec{v}^{A}_{NSR}$. The number of white points is defined by:

$$PointNumber = 0.1 \times \frac{\vec{v}^{a}_{SNR} - max\,\vec{v}^{A}_{NSR}}{0.5 \times max\,\vec{v}^{A}_{NSR}} \quad \text{(E 5-4)}$$

3. The image is blurred by using a *7 * 7* Gaussian low-pass filter if $\vec{v}^{a}_{NSR}$ is larger than *1.5 * max* $\vec{v}^{A}_{NSR}$.

Together, the three filters introduce a wide range of distortion in the final image.

Resizing the output image by audio duration

Image size is used to represent the duration of the audio clip. An image tile is resized according to its corresponding audio clip.

The longer the audio lasts, the larger its image size. In order to be able to display multiple timeline mosaics simultaneously, the maximum width of a component image is set to be 512 pixels and the smallest width is 128 pixels. The 128 pixel width corresponds to 0.125s and the maximum width (512 pixels) is for 8s. If an audio clip is less than 0.125s, it is set to 0.125s (128 pixels for width). When an audio clip is longer than 8s, it is set to be 8s (512 pixels) for width estimation. For an audio clip with $\vec{v}^{a}_{duration} \le min\, A\vec{v}^{A}_{duration}$, the image size is scaled linearly between the smallest and largest size depending on the audio's duration. By setting the maximum and minimum durations for width calculation it can be ensured that the image is always sufficiently clear. When two audio clips have similar durations, the widths of their image tiles are similar after linear scaling.

After these four stages, an image tile is generated to represent the given audio clip. All the image tiles are generated independently.

5.8 Mosaic Time-Line Generation

When the image tiles have been generated for all the audio clips in a given audio file, following the steps discussed in Section 5.4, the next process is to merge the image tiles together into one image or time mosaic.

It is not practical to assume that all audio clips have the same duration in a general audio file. The sizes of template images are not always the same either. Take the audio

file used in Figure 5-2 and Figure 5-3 as an example. As discussed in Section 5.1, the audio shape does not effectively show the durations of the audio clips. If the template is scaled according to duration, the sizes of image tiles will be different. Simply putting the image tiles side by side according to their time sequences would result in an image such as those shown in the top set of images in Figure 5-7.

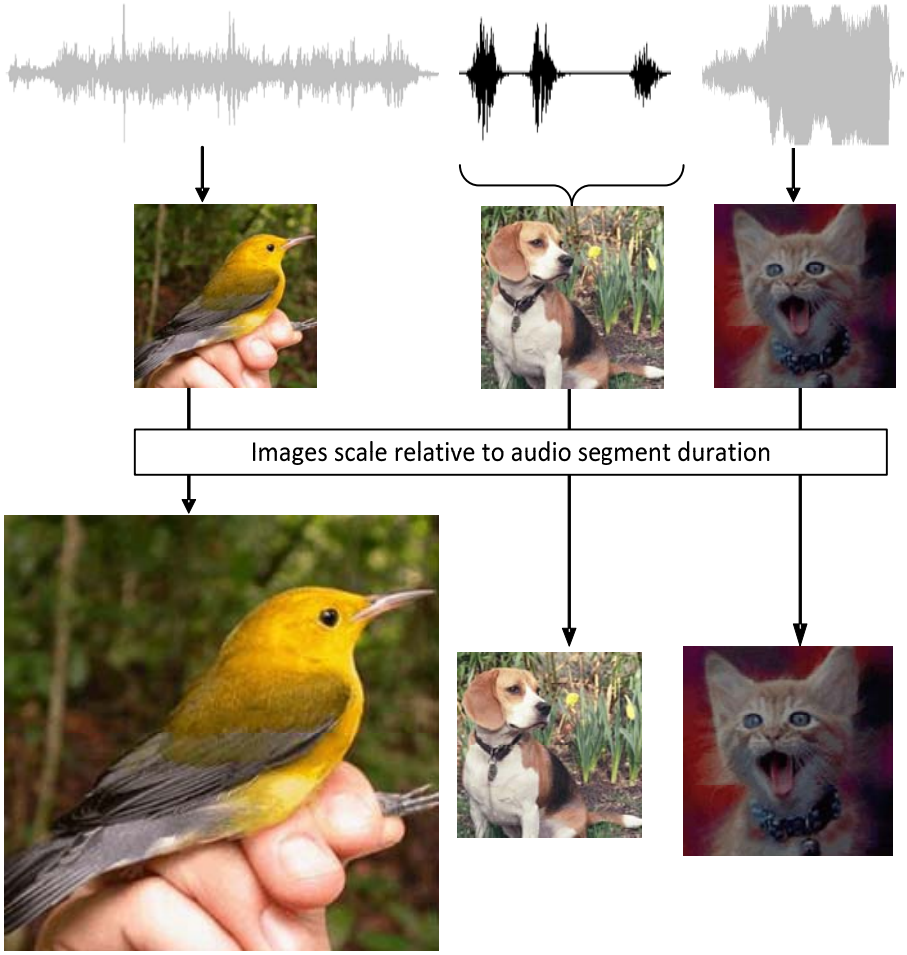

Figure 5-7: Result of image tiles with width of each image used to represent its corresponding audio clip duration.

In this module, the output image consists of rectangular images of different sizes (different widths and heights). The challenge is to merge differently sized images without significant information loss in any of the component images that form the final mosaics. Constructing a blended mosaic image is therefore appropriate since the image sizes will vary, given that embedded audio clips are of varying lengths. Blended image mosaics solve the problem since smaller images can be embedded into a larger image background.

Poisson image editing [185] is an effective approach for seamless image composition in the gradient domain. The new image is created by pasting a region from a source image onto a target image. To construct a time-line mosaic from the component images Poisson image editing [185] is employed to fuse the separate images together without significantly altering the content of each region.

The optimization process seamlessly inserts new content into a subset, Ω, of an existing image, h. It computes a new image, f, whose gradient, ∇f, within Ω is closest to the gradient ∇g, taken from a second image, g. The original boundary, $\partial\Omega$, of region Ω from h is also used as a constraint to ensure that the region Ω blends with the surrounding image, h. The final image constrains an interpolation of the boundary conditions, $\partial\Omega$, inwards while conforming to the spatial changes of the guidance field from g as closely as possible within Ω. The minimization problem is written as:

$$\min_f \iint_\Omega \left|\nabla f - \nabla g\right|^2 \qquad \text{with} \quad f\big|_{\partial\Omega} = h\big|_{\partial\Omega} \qquad \text{(E 5-5)}$$

The reader is directed to Pérez et al. [185] for the discretization of the problem and for suggested iterative solvers. Using Poisson image editing, any two images with the same height can be merged seamless to one image (see Figure 5-3).

For those image tiles with different heights, there will be holes at the top and bottom of the smaller image (see the empty regions on the top and bottom of image dog and image cat). These holes must be padded in an unobtrusive way.

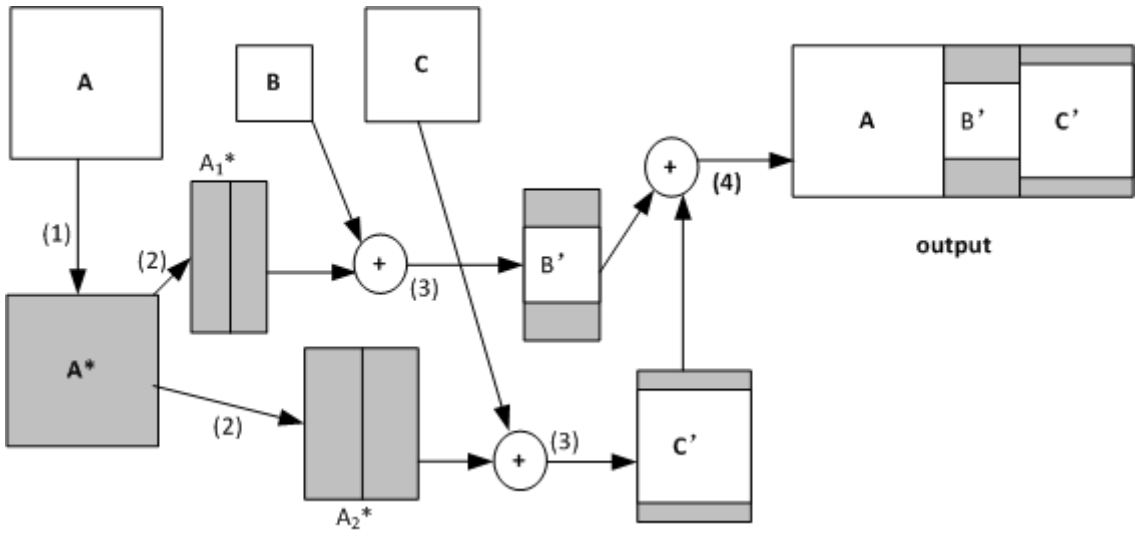

Figure 5-8: Background texture generation.

To achieve this, the stages depicted in Figure 5-8 are followed to generate the final image. In this mosaic image there are three tiles (A, B and C) and their sizes are shown in the figure. Their resultant image should be the output image at the right of Figure 5-8. The height of the resultant image will be the same as the highest image tile, which is image tile A in this example. Tile A does not require padding and is placed directly into the output mosaic. Heights of tiles B and C are smaller and require varying degrees of padding. For this, the highest tile (A) is selected to generate a seamless background texture to pad the smaller tiles. As most of the important information within our archetypal images is in the centre, padding image A is conservatively scaled to 90% of its original size and the result is integrated into the original tile A with Poisson blending to generate additional, low-content padding. This process repeats 10 times. The final image A* has the same size as the original image A. The content of the A* image is the padding texture that was generated from the edges of the original image.

If the width of the background texture image A* is bigger than the width of the processed image tile, the left and right boundaries are cut and merged to generate a background image (see note (2) in Figure 5-8). This is based on the assumption that the important information is in the middle of the image. When the processed image tile is wider than the generated background image, the width of the background texture image A* is then scaled to fit the width of each image tile. The A_1* is used for the background texture for image B and the A_2* is used for C. Tile B is then merged into the middle of A_1* with Poisson blending to generate the image for tile B′. When all the image tiles are padded, they are merged seamlessly by Poisson image editing in the order of their time sequences.

The audio file in Figure 5-2 and Figure 5-3 is employed to show how the background image is generated and how to merge the image tiles to the final resultant image. The input audio of the first example includes a long bird sound, three short dog barks and a relatively long meow of a cat. Note that the power or amplitude of the sounds cannot be read directly from the signal because the audio clips are normalized to show their shapes. This is necessary for audio clips with small amplitudes.

After segmentation, the input audio file is separated into five audio clips each containing only one sound. The three dog's barks are separated into three clips each containing only one sound. After classification, the classes that the audio clips belong to

are known and the template images for each audio clip have been selected. From the durations of the audio clips, the template image for bird is scaled to the maximum size (512 pixels of the width) and the template image for dog is scaled to the minimum size (128 pixels of the width). The aim of generating a background texture image is to pad up the smaller images so that the image tiles are the same height. The bird template image is selected to generate background texture image because it is the highest one of the scaled template images.

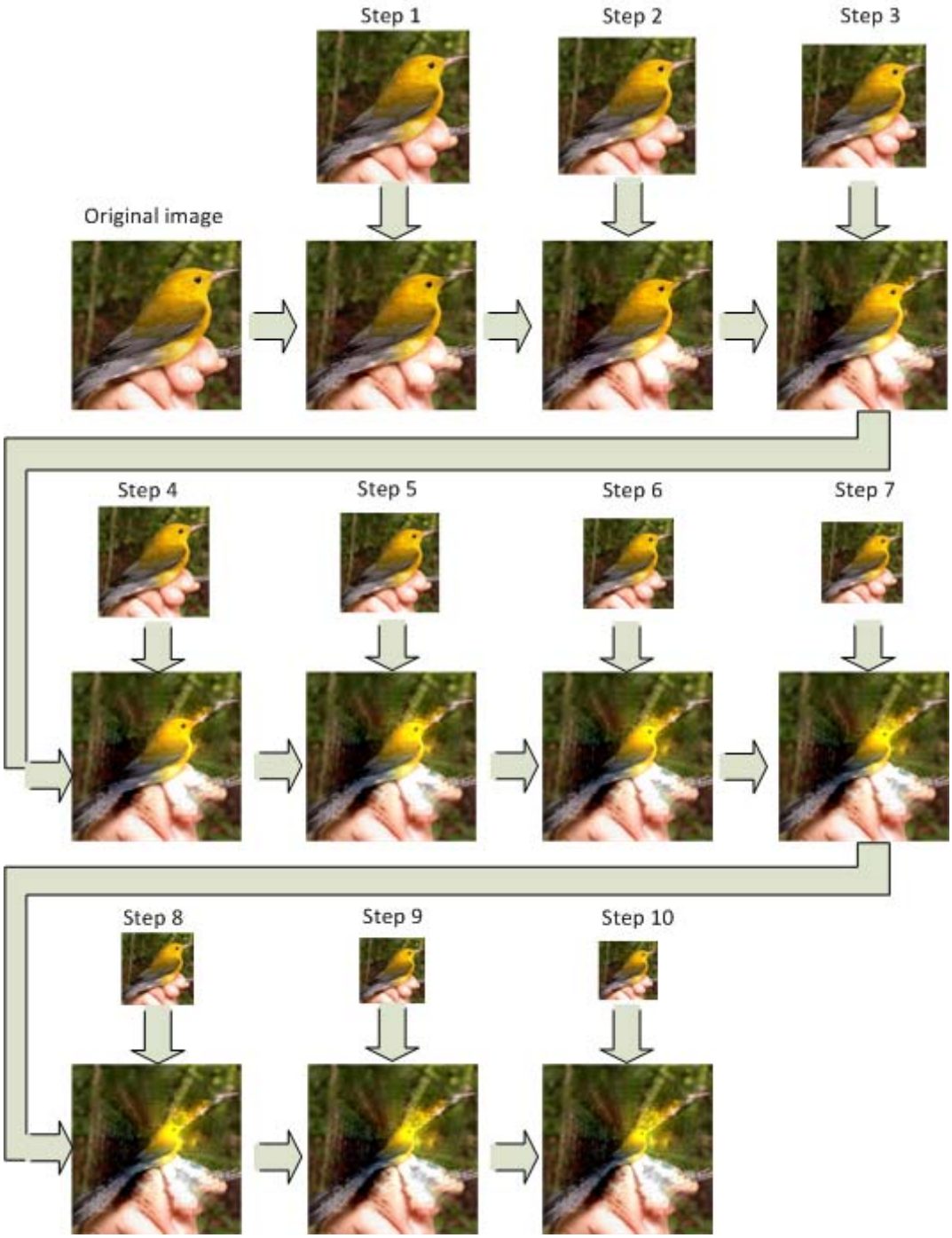

Figure 5-9: Steps for background texture image generation.

The background texture image is generated following the ten steps illustrated in Figure 5-9. The first step is to form a core image by scaling the image to 90% of the original size. It is then merged into the middle of the original image, which is called the target image. Poisson image editing is used for merging the core image and the target image. The resultant image of the first step is used as the target image in the second step, which is used to accommodate the core image (scaled 90% of the size of core image in step 1). After repeating the process ten times, the resultant image contains a much smaller object compared with the original image. If there are boundaries around the object in the image, the boundaries are widened in the resultant image so that it can be used as a background texture image. If the bird is regarded as the object in the template image, when the resultant background image is compared with the original template image, the background above the bird in the template image is extended to a larger area and the object (the yellow bird) is zoomed out to the middle of the resultant background texture image.

After generating the background texture image (shown in Figure 5-9), Figure 5-10 shows how it is used to pad the smaller images to form image tiles of the same height. Two strips are cut from the background texture image then merged together using Poisson image editing. The width of the two strips is determined by the width of the scaled template image. When they merge together, the resulting new background strip has the same width as the template image. The last step is the generation of an image tile by merging the scaled template image into the middle of the background strip. The images in the middle (L) of Figure 5-10 show the generation of an image tile for the dog's barks in the example audio file. The images on the right are for the cat's image tile generation. When all the image tiles are generated (padded to the same height) as shown in the bottom left image of Figure 5-10, Poisson image editing is employed again to merge them together side by side. The bottom right image is the final result and represents the audio file bird-dog-cat. This result indicates what sounds are included in the given audio file and their duration, but it has not yet been subjected to image filtering.

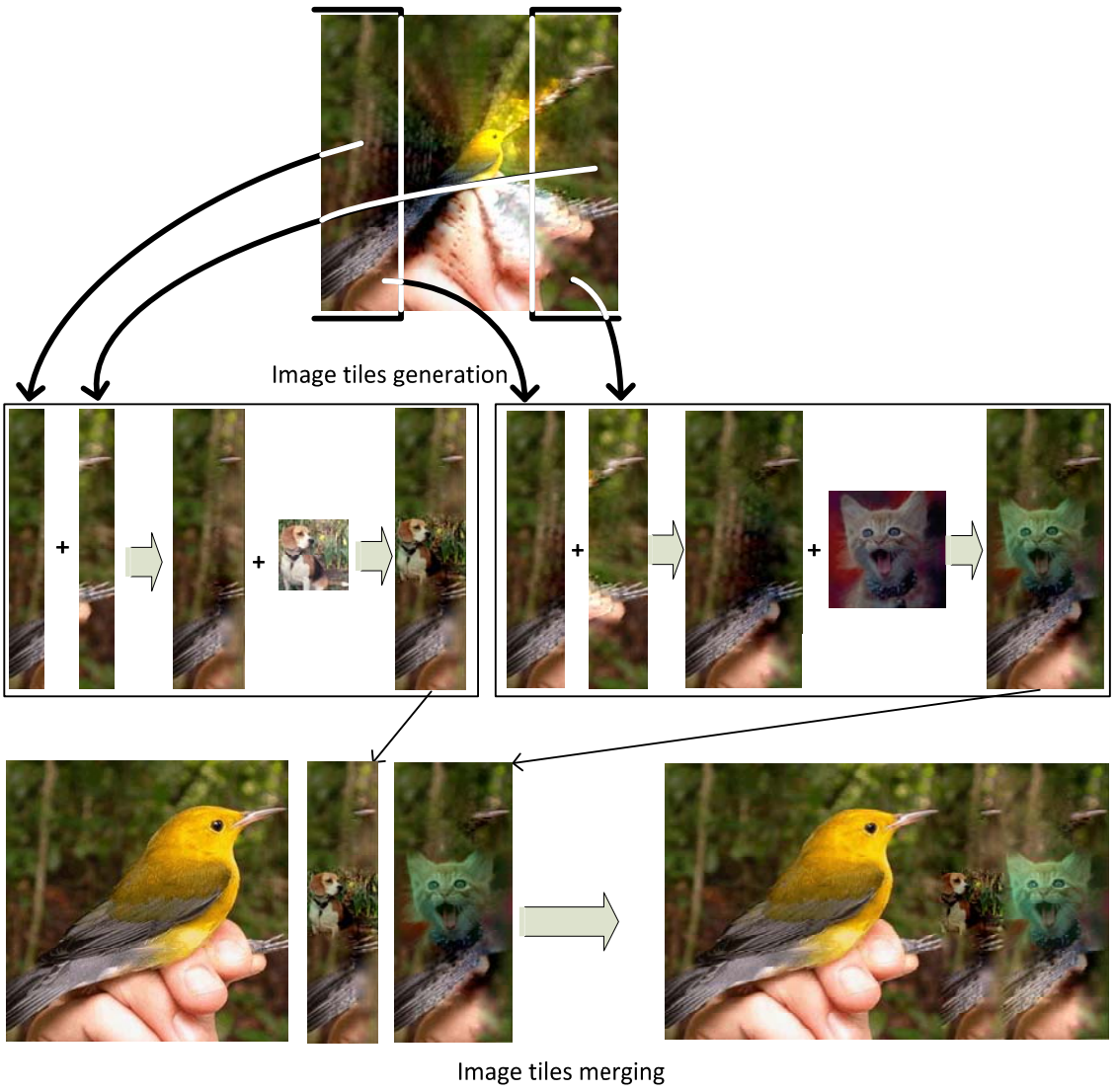

Figure 5-10: Generation of time mosaic images for the audio in Figure 5-2. (Top: Background texture image from Figure 5-9. Middle (L): Generation for image tile-dog. Middle (R): Generation for image tile-cat. Bottom (L): Three separated image tiles. Bottom (R): Resultant time mosaic image.

5.9 Template Image Selection

The techniques used in image tile generation and time mosaic merging were discussed in Section 5.5 and Section 5.6. But a good resultant image for a single sound, or a time mosaic image for an audio file containing multiple sounds, requires not only practical mosaic generation techniques but also suitable template images. In other words, the quality and appropriateness of the template images in the database partly determine the quality of the final results.

The requisites for a good template image include many factors such as the colour, the contrast of the image, the position of important information (objects), and whether the

image can completely describe the class it represents. If a template image cannot represent an audio file, the resultant time mosaic image will not be accurate and meaningful.

Figure 5-11: Bird-dog-cat example with alternate template images.

Figure 5-12: Bird-dog-cat example with alternate template images.

Figure 5-11 and Figure 5-12 are two time mosaic images for the same sound but are generated using different template images. If we compare the dog sounds image tile in the three mosaic generation results (Figure 5-10, Figure 5-11 and Figure 5-12). The first

two use the same template image. In these two results, because the most important information (the dog) of the template image of class "dog bark" is very close to the left edge, part of the audio clip is not clearly represented. The dog image tile in Figure 5-12 is very clear and unabridged because the object (dog) is in the middle of the template image. Another example is the image tile for cat in Figure 5-10 and Figure 5-11. Although the object (cat) is in the middle, it occupies most of the template image. When merging it with background texture image, the colour of the object is changed (because of the Poisson image editing process) as well as the shape of the object. But the size of object in a template image cannot be very small just because it represents a very short audio clip. The object must be able to be seen clearly even when the template is scaled to a very small size.

The best size of the object in a template image is one where it occupies approximately 90% of the image, which means the object has 5% of the image as outer edge. This enables the viewers to see the object clearly. At the same time, when it is merged into a background texture image to form an image tile, the edges around the object can merge with the background image to prevent loss of object information. When the template image is used to generate the background image, the edges can be merged together, over the 10 steps described in Section 5.8, to form a background image with the object removed (see the background texture image in Figure 5-11 and Figure 5-12). The background image in Figure 5-10 is not ideal because the object (bird) occupies most of image. The background around the bird is not big enough to produce a good background image for the other two image tiles.

So that the viewer can see each object clearly, the object and background should have an obvious contrast (like the image tiles in Figure 5-12). The template image, dog, in Figure 5-10 and Figure 5-11 does not produce an optimal result because the object colour is similar to the background colour.

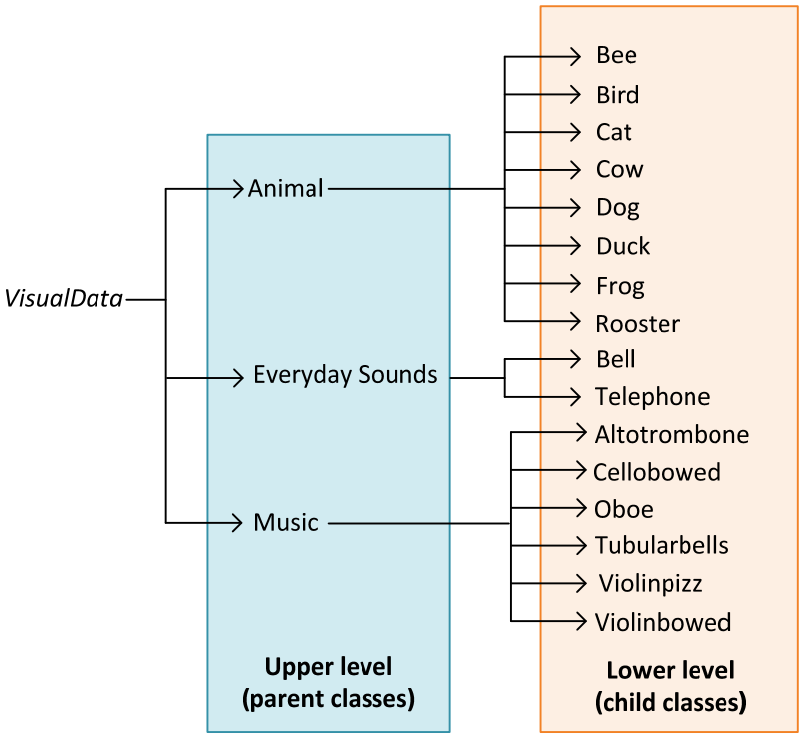

Figure 5-13: Hierarchical structure of the *VisualData* for template image selection.

When merging more than one image tile, if the background colours (boundaries around the main object) of the image tiles are similar, they will result in a harmonious image. In reality it is unlikely that the dissimilar sounds will occur together in the same audio file. For example, it would be very unusual for a duck's sound to be followed by a period of violin playing. We make an assumption that it is more likely that related sounds will occur together in an audio file. Therefore, in a large database containing various kinds of sounds, the template images can be selected by adopting a hierarchical point of view as shown in Figure 5-13. There are three classes in the higher level in the hierarchically structured *VisualData*. They can be divided into 16 sub-classes which are used in the audio visualization system. This hierarchical structure can be used to guide the selection of template images for more homogeneous time mosaics. When selecting a template image for a sub-class belonging to the class "animal", consideration should also be given to template images for other sub-classes belonging to this class because there is a high possibility that template images belonging to the same class will appear together. If all the template images of the classes belonging to the class "animal" have very similar background, when they were merged together the resultant time mosaics would have homogeneous backgrounds.

5.10 Experimental Results

In Section 5.5 and Section 5.6, the selection of suitable template images and the process of time mosaic generation have been discussed with an example. The detailed processes are also demonstrated using the same audio file, the "bird-dog-cat" example. But the results in Figure 5-10, Figure 5-11 and Figure 5-12 were generated from template images without filtering. In this section, a set of experimental results obtained using not only template images but also image filtering driven by acoustic features, are discussed. The template images in Figure 5-12 have been shown to be more suitable for visualization using time mosaics so they are used for further experiments in this section:

Figure 5-14: Bird-dog-cat example with filtering.

Figure 5-14 is a result generated using both the template image and the audio features of each audio clip. In comparison with Figure 5-12 (without filtering) and with the template images, Figure 5-14 contains a visualization that shows that the cat's meow includes significant background noise; it is quite blurry. The noise ratio of the bird's sound is lower than the cat's sound and the sound of the dog has the lowest noise ratio of the three. Therefore the image of the dog is the sharpest and that of the cat is the most blurred. The legend in Figure 5-5 or the template images in Figure 5-14 can be used to evaluate the visual cues in the time mosaic shown in Figure 5-14. The pitch of the bird's sound can be visually recognized as relatively low, in its class, because the image tile is very dark when compared with the corresponding template image. The colour depth of the bird's sound image tile is also lower than that of its corresponding template image tile. Therefore it can be concluded that the audio power is low when compared with the

average power values for that class of audio files. It can also be concluded that the pitch values of the dog's bark and cat's meow are similar to the average pitch value in their respective classes because neither image tile is obviously brighter or darker than the template images.

Similarly, comparing Figure 5-14 and Figure 5-15, it is possible to determine visually which image and thus audio clip is noisier (for example the cat tile Figure 5-14 is noisier that the cat tile in Figure 5-15), which has a higher pitch (the cat tile in Figure 5-15 is brighter than that in Figure 5-14) and which has a higher power value (the colour depth in the bird tile in Figure 5-14 is very low compared with the bird tile in Figure 5-15).

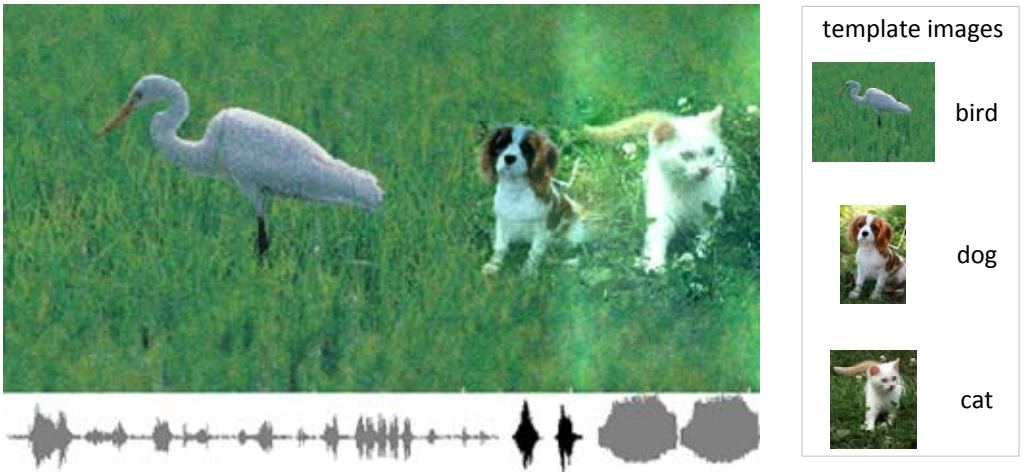

Figure 5-15: Time mosaic image for an audio file different from that in Figure 5-14.

Figure 5-15 is a time mosaic image for another audio file containing bird song, dog bark and cat meow. To enable it to be easily compared with the visualization of a very similar mixed sound audio file (Figure 5-14), the duration of the components sounds and the audio file have been setup so that they are the same. By comparing the image tiles in the resultant time mosaics with the template images, the user may identify that the pitch of the cat sound is high because the image tile of cat is very bright. Additionally, the noise ratio in the bird is higher than the other two sounds in Figure 5-15, but it is not as noisy as the cat image tile shown in Figure 5-14. By comparing the time mosaics in Figure 5-15 and Figure 5-14, it can be ascertained which audio clip has the greatest background noise, which has a higher pitch and which has greater volume.

Figure 5-16: Frog-bee-cow example.

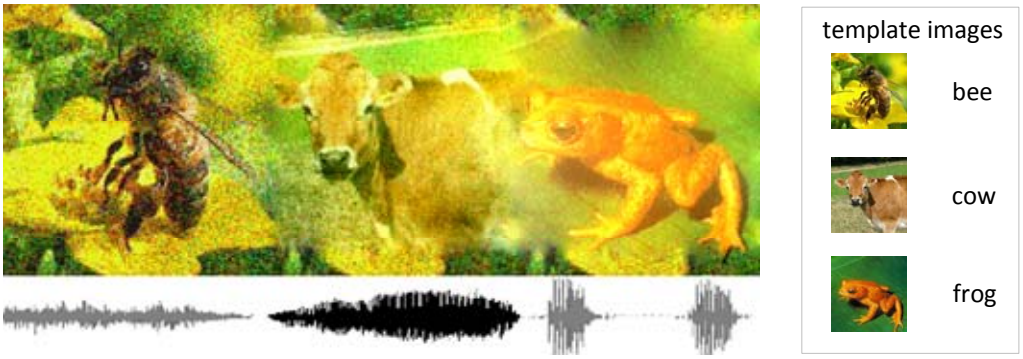

Figure 5-17: An audio file containing sounds from class "bee", "cow" and "frog".

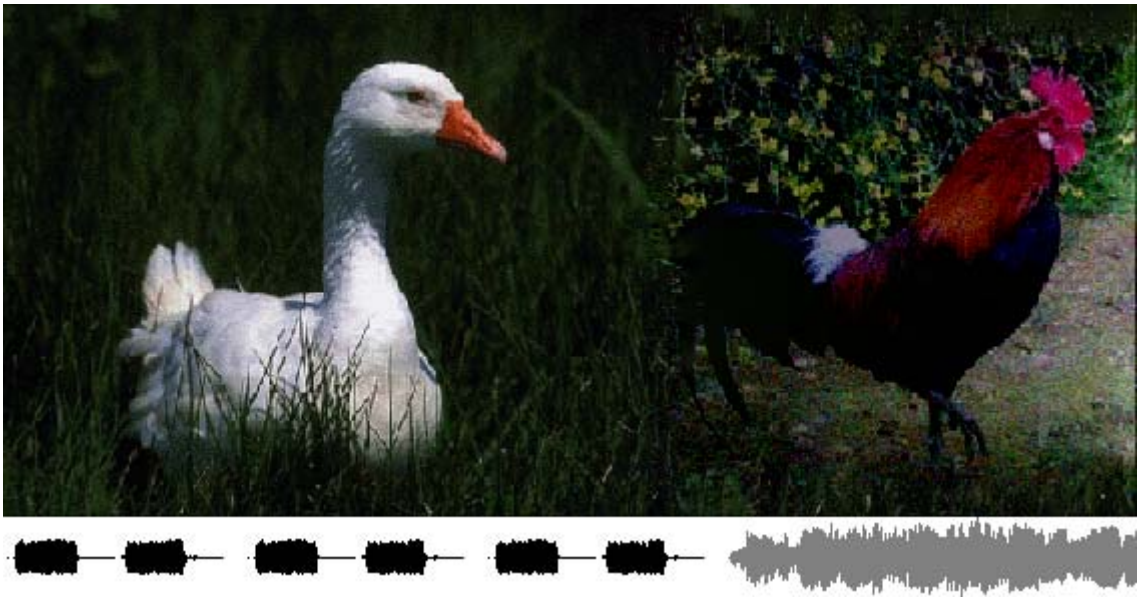

Figure 5-18: An audio file containing duck and rooster sounds**.**

Figure 5-16, Figure 5-17 and Figure 5-18 show three further examples of the visualization of randomly generated audio files, and Figure 5-19 shows an example with five detected sound clips. Figure 5-20 and Figure 5-21 give two examples of the visualization of randomly generated audio files for music. Figure 5-20 contains three music clips: oboe, trombone and cello, and Figure 5-21 has violin and oboe sounds.

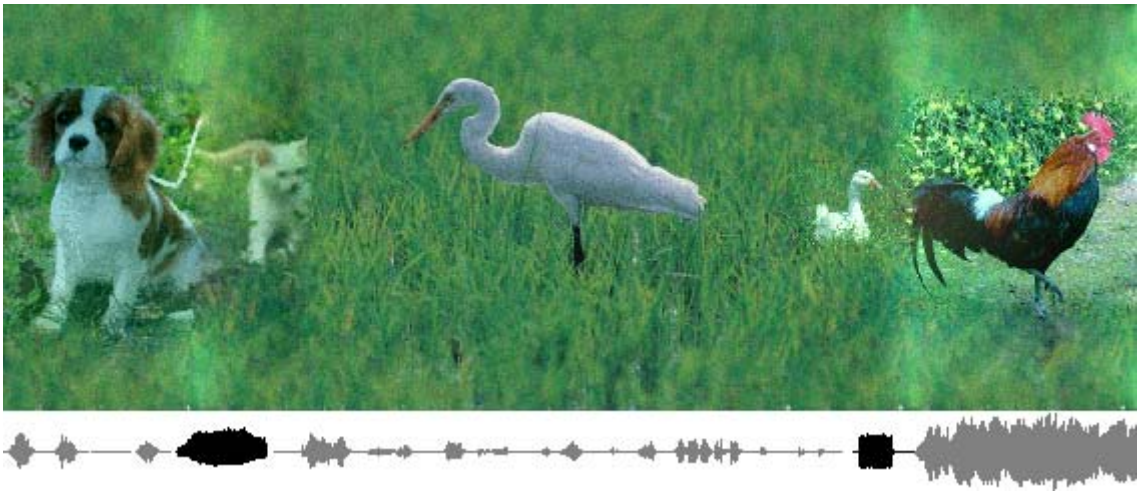

Figure 5-19: A time mosaic of five images representing five different sounds.

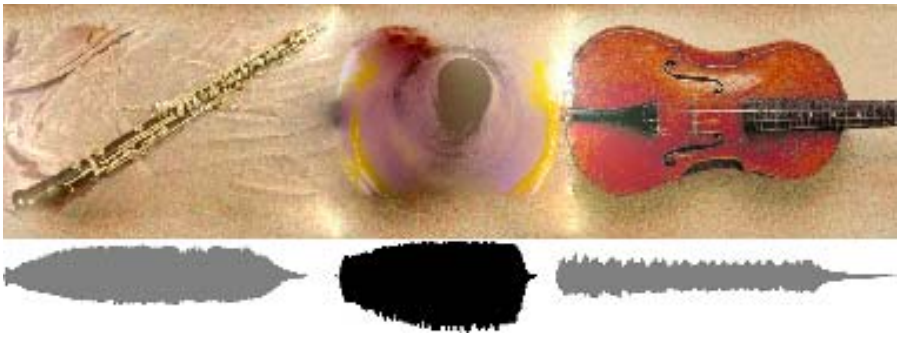

Figure 5-20: A time mosaic for an audio file containing three distinct instrument sounds (oboe, trombone and cello).

Figure 5-21: A time mosaic for a music audio file containing a violin and oboe sounds.

5.11 Limitations

The main limitation of the time mosaic module is in the selection of template images and the quality of the pre-built database. The template images of the classes partly determine the quality of the results. From the time mosaics generated, it is clear that the colour and contrast of the image, the position of important information and whether or not the image can completely describe the class it represents are basic requirements for being a good template image.

When the template images are well selected, the success of the time mosaic module depends on the accuracy of the segmentation and classification modules. An incorrect segmentation and classification result may lead to spurious results under certain conditions. This effect has been minimized by employing the *uncertain* criteria in classification to identify audio segments that may not belong to an existing class in the ontology. These segments may then be removed from the visualization result or the user may choose to allocate a class to the file. If the file is classified into a new class a template image must also be selected for that class.

Figure 5-22: Incorrect segmentation and classification produce an incorrect result.

Figure 5-22 illustrates an incorrect visualization result because of the incorrect segmentation and classification of the audio input. The sound file contains a cat-frog-frog sequence of sounds. The three sounds were segmented to four audio clips. Then the middle two segments were incorrectly classified as belonging to the cow class. Because the two middle segments were classified to the same class they were combined for

visualization. The duration of the entire input audio file is used to determine the width of the resulting image. Three other visual features (brightness, colour depth and "noise" or blur) for the resultant image were determined using the audio properties calculated for each of the three segments. So the resulting time mosaic, Figure 5-22, is generated using three image tiles with each tile's brightness, colour depth and blur filtered by using the corresponding segments audio properties. In this case the anomalous "cow" sounds are not found using the nearest neighborhood check for two reasons; firstly the cat and frog terminal segments both have a relatively long duration and are therefore considered to be correct and secondly because the middle two segments are of the same class and duration they would be considered to be correct. Additionally the "cow" segments are not marked as *uncertain* when using new class detection because their *NFLd* is well below the threshold for new class detection. Therefore Figure 5-22 shows the worst possible outcome for visualization of a mixed sound audio files, an incorrect visualization where the user has been given no indication that the visualization may not be correct. However it is important to note that this would occur only approximately 1% of the time. Ideally of course we would wish for 100% accuracy but given the well reported difficulties associated with audio segmentation and classification we consider this to be a good result.

If an audio file contains an audio clip that doesn't belong to any class in the database, the system can recognize it using new class detection and the *uncertain* criterion. Therefore the system can notify the user that the accuracy of audio visualization is questionable.

Also, due to limited screen space, the durations of audio clips may not be represented correctly. As was discussed in the image tile generation section, when the duration of any audio clip is shorter or longer than the pre-set time limitations, the resultant time mosaic image cannot show their relative durations correctly.

5.12 Other Usages for Time Mosaics

The time mosaic method can be used for purposes other than visualizing audio input, such as navigating an audio database and querying an audio database.

Navigation for audio database

Figure 5-23 illustrates a class in the database where each class, based upon a fixed and pre-determined ontology, contains a single manually chosen template image. The template image for the class in this example is shown top-left. The remaining images that are associated with each audio file in the class are automatically generated by the system using the template and each file's audio properties. The differences in audio features between the audio files in a class determine the different visual features of their corresponding images. As new audio files are added to the database, the audio is analyzed and a new representative image is produced.

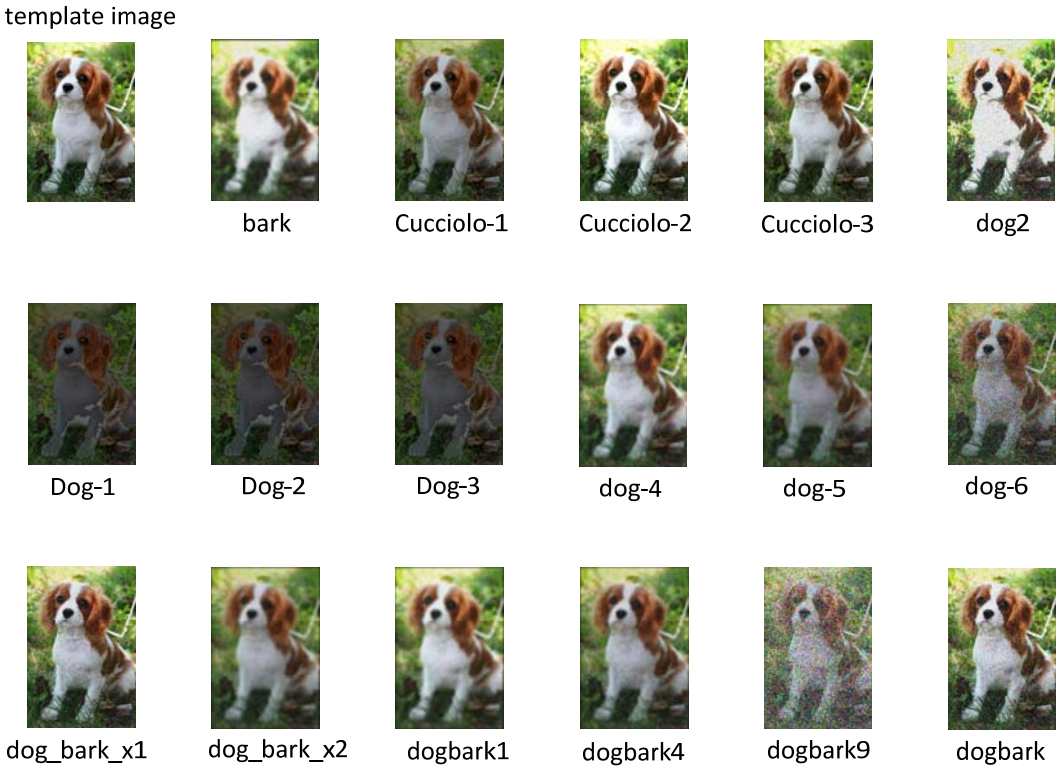

Figure 5-23: Images representing audio files in the "dog" class from the *VisualData* database. The label under each image gives the name for each audio file.

The Figure 5-23 gives an example that would allow for quick visual scanning of a class in an audio database using images generated by the audio visualization system. From the images, viewers can identify different properties and select the sound that they require. For example, one can easily recognize that the images for audio files "Dog-1", "Dog-2" and "Dog-3" are darker than the rest which means their pitch values are lower than the rest of the audio files. The three audio files (Dog-1, Dog-2 and Dog-3) appear to be the same visually. This is because they are relatively acoustically similar in terms

of the audio features pitch, power and SNR when compared with the other files in the class. The duration of the audio files are not shown in this image. To find differences in audio features between Dog-1, Dog-2 and Dog-3 further analysis, additional to the image tile visualization, is required. The same applies to other properties, for example, the audio file "dogbark9" contains the greatest amount of background noise. We propose therefore that at a glance, viewers could visually scan the selected audio files and that this visualization could aide in the selection of an appropriate audio file.

Using different audio-visual feature relationships

The audio-visual feature relationships can be defined by users according to their specific requirements. This makes the application of the audio visualization flexible. For example, in the time mosaics presented in this chapter, the pitch determines the brightness and power is related to colour depth. If colour depth is used for pitch and brightness for power, the resultant time mosaic image for audio in Figure 5-14 will appear as shown in Figure 5-24.

If a viewer is provided with an appropriate legend, it should be possible to recognize from the resultant images that the pitch for audio clip bird is relatively low in the class and its power value is at an average level in its class.

Figure 5-24: Filtered bird-dog-cat example with different mapping relationships.

5.13 Summary

This chapter described the time mosaic generation module in the audio visualization system. The module processes the output audio clips and their template images from the segmentation and classification modules, and results in a seamlessly constructed time mosaic image by positioning the image tiles along the time axis according to their time sequences.

Viewers can assess the audio properties in the sound sequences from the visual filtering operations. As a key module in the proposed audio visualization system, the time mosaic module generates the resulting images for any input audio files. It may also provide a useful way of quickly scanning audio query results and navigating an audio database through browsing.

The time mosaic module could also be adapted to other applications by employing some professional audio features for specific presentations according to users' demands. Therefore when compared with the input audio files, the resultant images could be made to convey features other than those that are perceptible.

Chapter 6
Visualization Approach 2- Video Texture

This chapter presents an approach to the visualization of sound using video texture and video filtering. The method of visualizing audio files is extended from the use of static images to the use of videos. Instead of using static images to represent an audio clip, video components are employed. An output video is composed of video components representing audio clips derived from the audio file. These video components are generated by applying video texture techniques to template videos. The visualization of audio files with video textures is suitable for lengthy input.

Video texture mosaics are a useful enhancement to the time mosaic approach to audio visualization. As mentioned in the framework of the audio visualization system (Section 2.2), image mosaics are used to initially represent audio files. Then, if users are interested in an image or require more information about the audio file, the video texture mosaic can be played by selecting the relevant tile in the image mosaic. If all the image tiles in an image mosaic result are selected, a video texture mosaic result for the entire audio file will be played. We propose that video texture mosaics provide users with more time-based information than a time mosaic.

This chapter begins by describing the advantages of using video frames as an alternative to static images for audio visualization. There is then a review of video texture techniques and how these can be used to generate video components of sounds. The term *video texture component* refers to a piece of video texture which is generated to represent a period of audio input and that corresponds to an image tile in its time mosaic. Different types of video texture mosaics and the methods used to generate them are discussed. Lastly, some examples are given and the video texture generation module tested in the context of the audio visualization system.

The audio file and template video pair for the experiments in this chapter that represent a dog barking were recorded by myself and feature Dr. Stephen Brooks' accommodating beagle *Fred*. The remaining, freely available, video templates (some of which contain watermarks) employed were downloaded from the iStockVideo website [200] and some of the audio files were downloaded from the Comparasonics website [201].

6.1 Limitations of Image Mosaics

In Chapter 5 of this thesis a novel approach to the visualization of audio files using seamless image mosaics along a time axis, was presented. Each component image in the mosaic was mapped directly to a discovered component sound and was subjected to image processing filters driven by specified audio characteristics. A component sound, used to generate a time mosaic, was either a single audio segment or a sequence of combined audio segments where the sequence of segments belonged to the same class. Using time mosaics, viewers are able to scan an image mosaic to discover visual content and thus determine audio properties. Although image mosaics enable users to view the audio content they have some drawbacks:

1. Some acoustical qualities are inherently dynamic and cannot be represented by individual static images. For example, an image of a violin can convey that the sound is related to that of a violin. But the violin can be bowed or plucked and a single static image cannot necessarily effectively convey the way in which the violin is being played.
2. In time mosaics, a component image is subjected to image processing filters driven by the mean values of specific audio characteristics derived from its corresponding component sound. The audio properties cannot be well illustrated in a mosaic if the characteristics change rapidly.
3. When the duration of audio clips in a given input audio are quite different (e.g. one is fifteen seconds of handclap and another is three minutes of music) the image components (tiles) in the corresponding image mosaic maybe either too small to be recognized, or cannot show the duration accurately when the size of the image tiles are pre-defined. As seen in the top image in Figure 6-1, there are two short audio clips and one long audio clip in the input audio. If the template images are scaled according to the real durations of the audio clips, two images are very small and difficult to see clearly (see Result (A) in Figure 6-1). To avoid

the image tiles being too small to be seen clearly, the minimum size of image tiles can be pre-set. However, the image tile for a longer audio clip may be too large to fit the computer monitor if the image tiles are scaled according to the real durations of the audio clips they represent. If both the minimum and maximum size of image tiles is set, the problem then is that the resulting image components do not show the real durations of the corresponding audio clips.

4. Image mosaics do not adequately represent the ideas inherent within hierarchically structured databases. It is difficult to represent a hierarchically structured database within a single image. Traditionally, the parent/child relationships have been visualized using tree-like representations [202] [203] [204]. These representations are significantly limited by screen real-estate.

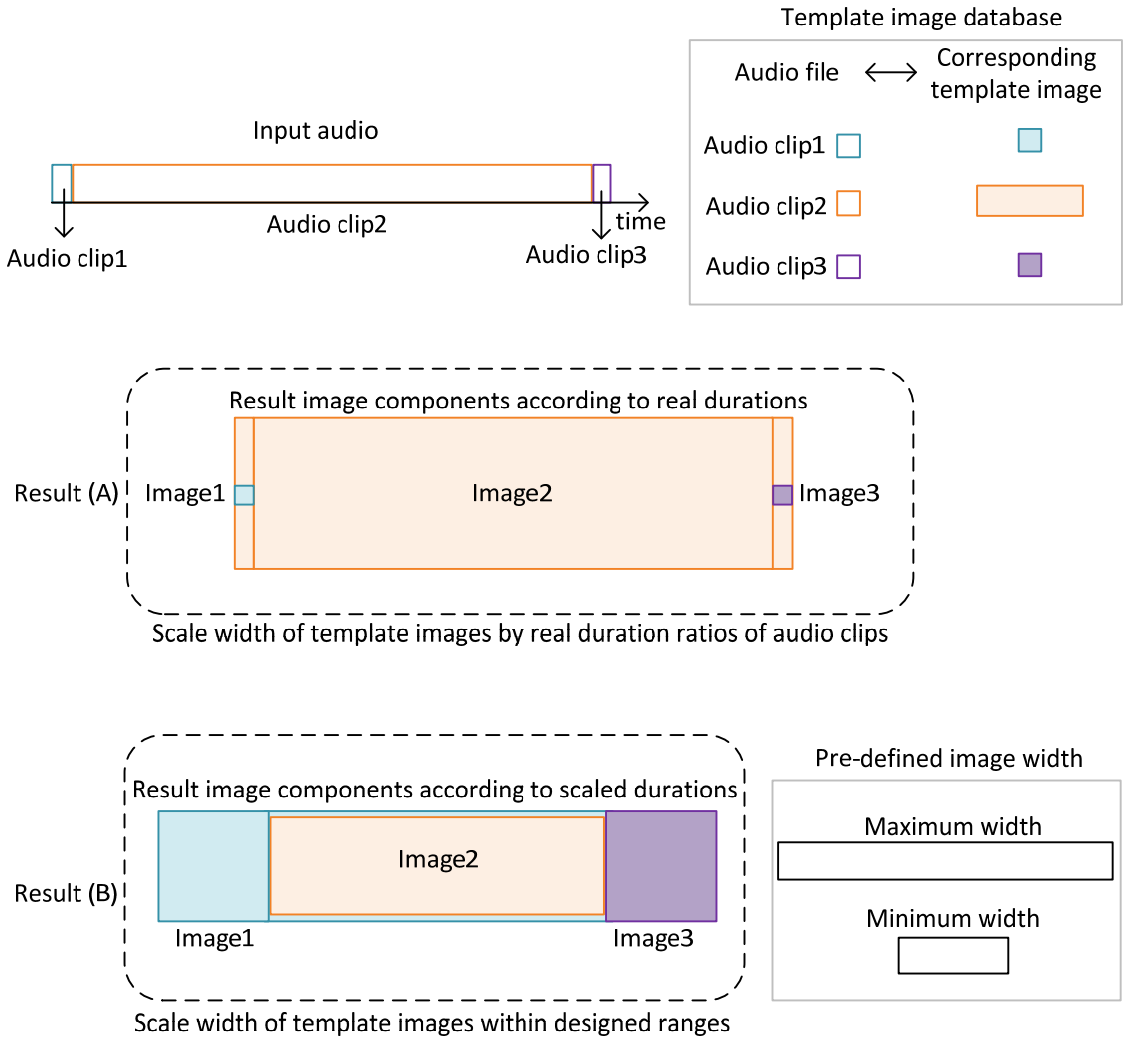

Figure 6-1: Image mosaic generation when the durations of the component audio clips are different; (T): Input audio with three audio clips and their corresponding template images; Result (A): Template images scaled to the actual clip durations. Result (B): A set range for template image scaling.

Video is a potential approach to visualizing audio. Video carries more information than a single image while at the same time holding the viewers' interest. When compared with image mosaics, videos have the following advantages:

1. Video displays dynamic motions that cannot be adequately captured by a single static image. A short video can record an entire motion and thereby enable the viewer to avoid misunderstandings that could occur if only a static image was viewed. For example, it can clearly separate a violin bowing sound from violin pizz.
2. Video contains a series of frames. This feature of video could be useful when attempting to identify and illustrate any changes over time. Users may be able to select key changes in an audio to "summarize" the audio, and to map these audio characteristics to certain frames in a video. Therefore by comparison with the original audio, the video may be shorter and yet show how the audio characteristics change over time.
3. Representation of duration via video run time is both more flexible and more accurate than its representation by image size.

Moreover, videos have been noted to be particularly suitable for natural phenomena [205] or motion, which are difficult to represent with a single image. In an audio visualization system there may be a large proportion of audio files that belong to the category of natural phenomena or that need to be described by motions. Because of these advantages, video is a compelling alternative approach to audio visualization. But we have to acknowledge that using template video also has some potential limitations. A video template must have a finite duration so it is unlikely to fully represent a long audio clip. Moreover, a video template may contain an entire motion with audio signal and silence periods. The audio signal is only part of the whole motion, simply repeating the video template will not always provide a good representation of an audio clip. For these reasons the idea of using video texture was explored.

Video texture is a technique, that uses a given short video clip to generate an infinite amount of video [206].Video textures can be considered to be, an extension of image textures [179] and a subset of all images that exhibit spatial stationarity. An image texture is composed of repeating visual patterns. Similarly a video texture has repeating visual patterns because it is generated by re-using frames from a video. We regard video texture as a suitable representation for sound files because a video texture can be

endlessly generated from a set of key frames, providing a varying stream of images from single or multiple videos. Additionally, video textures can accurately represent dynamic content without any obvious discontinuity in transitions. This unique aspect of video texture offers greater flexibility and visual richness than a single image or the sequence of images a video can provide.

There are other advantages associated with using video textures. For example, they can be used in place of digital photos to infuse a static image with dynamic qualities and explicit action. A single picture or image is static, a video can be dynamic but is finite, whereas a video texture maybe dynamic and either finite or infinite. Whenever a video is played, it shows exactly the same sequence of frames. Even if a video has a repeatable loop and it is played continuously, it is replayed in the exact same sequence of frames. For a video texture, although the individual frames may be repeated from time to time, the video sequence as a whole does not have to be repeated.

Video texture also addresses the problem of temporal discontinuities that is observed when a video clip is looped, and then goes further to convincingly produce a continuously varying video stream that can extend as long as necessary. This is accomplished by locating a number of plausible "loop back frames" within the video sequence and then randomly looping back or continuing as each loop back point is encountered when the video is played back.

Video texture, as a new type of infinitely varying media, has advantages over both static images and video. It is similar to video in representation and has the potential to address the drawbacks associated with video and provide a novel way of visualizing audio.

6.2 Video Texture Literature Review

Video textures are defined as repeating dynamic visual patterns [206] [207] [208] [209] [210]. They take the form of infinitely varying sequences of images extrapolated from a video of finite duration. The output video can provide a varying stream of images, which may come from one input video or different input videos.

Video textures were developed as an extension of earlier work on Video Rewrite (VR) [37]. VR techniques were used to generate videos of human mouth movements driven

by speech. The mouth images were stitched into background footage using a morphing technique. VR segments and matches the audio to the visual images by identifying spoken phonemes.

The use of video texture has recently been extended into applications, including Panoramic Video Textures [211], Animated People Textures [212] [213], Stochastic Motion Textures [214], Editable Dynamic Textures [215] [216], Video Mosaics [180], and Controllable Video Sprites [217]. Video texture generation is a sample-based method, that models true video clips and synthesizes new video sequences [206] [179] [215] [191] [194] [218]. Soatto, Doretto and Wu used a limited sample to express infinite information [207]. They also used a similar method for editing and synthesizing dynamic textures using image-based rendering [216], in which the spatial frequency content of the sequence is edited, modified or reversed. This algorithm is especially good for generating dynamic textures of natural phenomena such as flowing water, steam and smoke. While this method provides a potential approach for an audio/video visualization system it limits the type of audio files that could be visualized. Because the input is a sequence of limited images this technique is unsuitable for representing multiple motions or dynamics and therefore could only be used for a small subset of audio. In a similar approach, Lai and Wu [218] proposed an algorithm for directional temporal texture synthesis. Their work presented a novel temporal texture synthesis algorithm which only needs a single static texture image as input to synthesize temporal textures that approximate true video clips and generate infinite length sequences with semi-regular characteristics. This algorithm is also limited in its application because it can only be used to generate a video when the input is a static texture image. Therefore the resultant video is more limited in terms of motion and dynamics than a video that is generated using video input [206].

Auto-regressive processes were used by Campbell et al. for video texture generation [209]. The video texture generated using their methods may contain images that have not been presented in the sequence of original frames. Video textures with new frames can also be synthesized using wavelets in the algorithm [219]. New frames, based on wavelet coefficients derived from the decomposition of a set of pixel values of neighbouring frames, were constructed. Accordingly, new video textures were obtained by appending synthesized frames. Flow-based video synthesis and editing [215] capture the motions of textured particles in the input video along user-specified flow lines. This

technique can warp the textures along the input flow lines and synthesize textures over the edited flow lines to build virtual landscape animations. All three techniques generate new video frames. However, in our audio visualization system the original video should contain all the frames needed to represent the audio input and we should not require the generation of new video frames. Therefore these algorithms were not employed in our audio visualization system.

Celly and Zordan synthesized video textures of people by rearranging the original frames from a database of source footage videos [212]. Their approach is not employed in our video textures module because their approach needs a database of source videos classified using data motifs, which is especially good for human movements. Data motifs are used to identify the movements of repeated sequences. An example of this is a video of people performing yoga where the video classified according to yoga asana (pose) sequences. In our visualization system most classes only contain one motif or theme and therefore it is unlikely that the use of data motifs for classification and video texture generation would be required.

Bar-Joseph, et al. [194] analyzed whole input signals and constructed hierarchical multi-resolution analysis trees, called MRA trees. They transformed the newly statistically merged MRA trees back into signals, yielding output textures. However, this technique concentrates more on the texture synthesis than a meaningful video result. It is good for automatically generating movie clips of natural dynamic phenomena such as waterfalls, fire flames, or a school of jellyfish. In the generated video, the objects move continuously but the movements are small and repetitive. In our audio visualization system many of the audio files need to be represented using large, specific, directional motions. If we applied the technique of Bar-Joseph et al. it would most likely result in flickering in the generated video texture. While this method may be suitable for representing some classes of audio files in our system it will not provide a generally applicable solution.

Graph cuts have also been used to synthesize 2D and 3D textures [179]. In synthesizing temporal textures, the input videos were concatenated with optimal seams, which were irregular 3D cut surfaces. They allowed for interactively merging images but did not provide a smooth transition between frames when rendering the video. Our proposed video texture generation method will not allow for editing of the object or the texture in

frames of the template video. It should not be required because the template videos will be carefully selected and will therefore represent the content of all sounds in the class. However, in the future there might be cases where there is no suitable template video for a class in the database and in such cases this approach may be employed to generate videos because it allows for editing of the video frames and may therefore provide a better representation of the audio content.

Other related areas of research are video analysis [220] and querying [221] [222] [223]. It may be possible to perform audio searches or queries using the corresponding video textures of audio files by adopting these methods but the audio search functionality falls outside the scope of this thesis. An important component of this branch of research is the development of effective similarity measures between video sequences [224] [225] [226]. Similarity measurement between video sequences would be important if there is more than one template video for a class in the audio database because the output video texture does not need to be generated from a single template video. However for this thesis we only have one template per class so none of these methods have been employed. Other issues include effective indexing and recognition [227] which can be employed in the post-processing of video texture results for the audio input. The resultant video textures can be summarized or indexed by this method. This is perhaps an alternative for future work.

We suggest that video texture offers greater flexibility and visual richness compared to a single image or sequence of images. The video texturing technique can analyze and maintain plausible internal loops for multiple image regions of a video. For example, two children playing on a swing set in a video can be considered independent of one another, therefore multiplying the variations possible for video texture playback [206]. When generating video textures users can even produce video sprites and animate moving video objects together. Video textures can be used to produce very compact videos by varying the number of loops (and frames) in the video sequence. Moreover video texture is not limited to the content of a video template; we can apply the idea of video sprites to our system thereby allowing objects to be added or removed from the template frames.

In conclusion, for our audio visualization system, we believe that video texture is a valuable supplement to image mosaics, as a more detailed way of representing audio

files. The following section discusses the role of the video texture module in our audio visualization system.

6.3 Video Texture Generation

In this section the video texture generation method is described. Firstly the framework of this module is presented. This is followed by a description of a simplified video texture generation method. For better adaption to the proposed audio visualization system, a modified method is then proposed for the generation of the video texture. Then the limitations for template video selection and the number of frames in the resultant video texture clip are discussed in this section. Finally, the matching of audio features to video features is discussed.

6.3.1 The Video Texture Module Framework

The data input to the module is the set of audio clips that were produced by the segmentation module (Chapter 3). The template video for each segment is selected by the classification module (Chapter 4) in the same way as described for time mosaic image template selection. The outputs from this module are video textures representing the given input audio clip. The outputs include video texture components for audio segments and a video texture mosaic for the whole audio input. A video texture component represents the audio segment or audio segments that are represented by a single image tile in its corresponding time mosaic. A video texture maybe played when the viewer selects its related time mosaic image tile. The full video texture representing the entire audio file may be played when the entire time mosaic is selected.

Time mosaics and video textures differ in the way in which they represent the audio file. A time mosaic image visualizes only the signal segments. If the audio file contains silence periods, they are removed in the segmentation process. One of the advantages of audio visualization using video texture is that the output can visualize time-based audio features. The video texture should include both signals and silence periods to be an accurate representation of the audio file. Moreover, the resultant video texture mosaic should have the same duration as the audio input so viewers can view the content and, at the same time, listen to the audio file. Therefore the visualized audio clips in our video texture mosaics are different from those visualized in time mosaics because the video

texture is generated for the original audio file. The segmentation and classification steps are still required in order to select the correct video templates that are need to build the video texture.

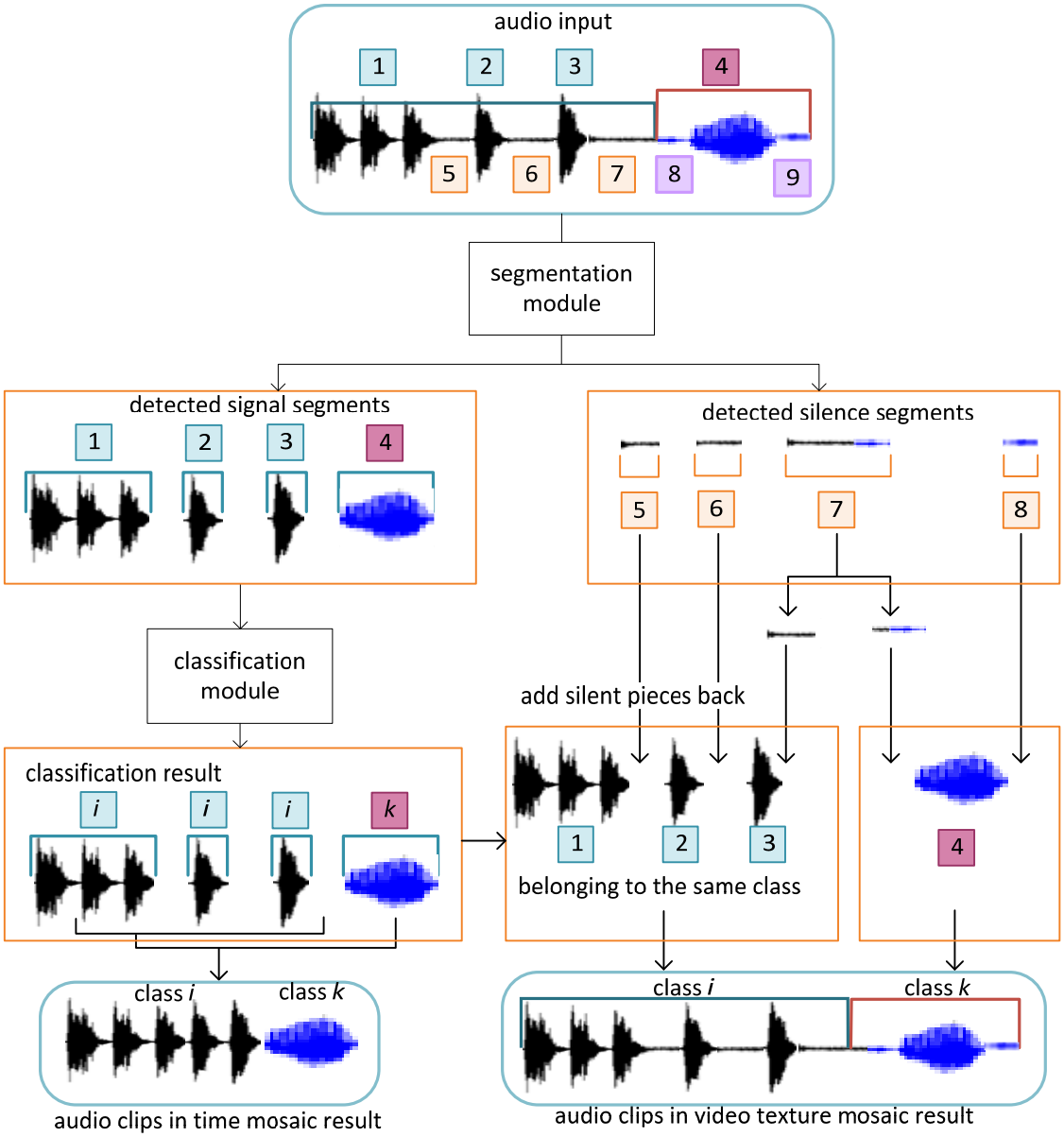

Figure 6-2: Difference between audio clips visualized by time mosaic module and video texture module.

Figure 6-2 illustrates how the complete audio file is visualized by a video texture. In this example the audio input contains two distinct sounds (coloured black and blue respectively). The sound from the first class has three signal segments (segments 1, 2 and 3) and three silence segments (segments 5, 6 and 7). The second sound has two silence segments (segments 8 and 9) and its signal component is segment 4. Segment 7

and segment 8 are adjacent and both are silence segments so they are removed by the segmentation module. Therefore the output of the segmentation module contains only four signal segments (segments 1, 2, 3 and 4). The classification results for these segments will show that the first three segments belong to the same class (class *i*) which is different from the class of segment 4 (class *k*). Segments 1, 2 and 3 are combined and represented by one image tile in the time mosaic. But for the video texture module, the silence segments are re-introduced as shown in the right images in Figure 6-2. Segments 5, 6 and 8 are directly added in. This is simple because their adjacent segments either belong to the same class or are a terminal segment. However, segment 7 is a more complex situation as it has been split in two during the segmentation process. The first part of 7 is combined with the previous segment 6, and the second part is combined with the next segment. Splitting the silence segment between two different sounds can avoid making one adjacent sound too lengthy and the other adjacent sound too short. Moreover, incorrect separation of the silence segment will not lead to an incorrect visualization. Because the spectral power value for a silence segment is quite limited, the colour depth of frames representing a silence segment is always very low. So the content of a visual frame representing a silence segment is not as critical as for a signal segment. Additionally we present, later in the chapter, a method by which the motion of the object in the video texture also represents silence and this further mitigates the issue of silence segmentation.

Figure 6-3 illustrates the framework of the video texture generation module. The duration of the audio input and the template video determine if a newly generated video texture is needed. If the duration of a template video is longer than that of the audio input, a portion of the template video can visually represent the given audio. So the output of the video texture module does not necessarily need to be a newly generated video because in such a case the video template itself is sufficient. The video texture module calculates the number of frames it requires, based on the audio files duration, and compares it with the number of frames in the template video. When a template video has more frames than is needed to represent the audio input, a subset of the template video is randomly selected from the template video. Otherwise when the template video is not long enough, an output video is generated using video texture techniques. There are a number of different methods for generating video texture components. They are discussed in the subsections that follow.

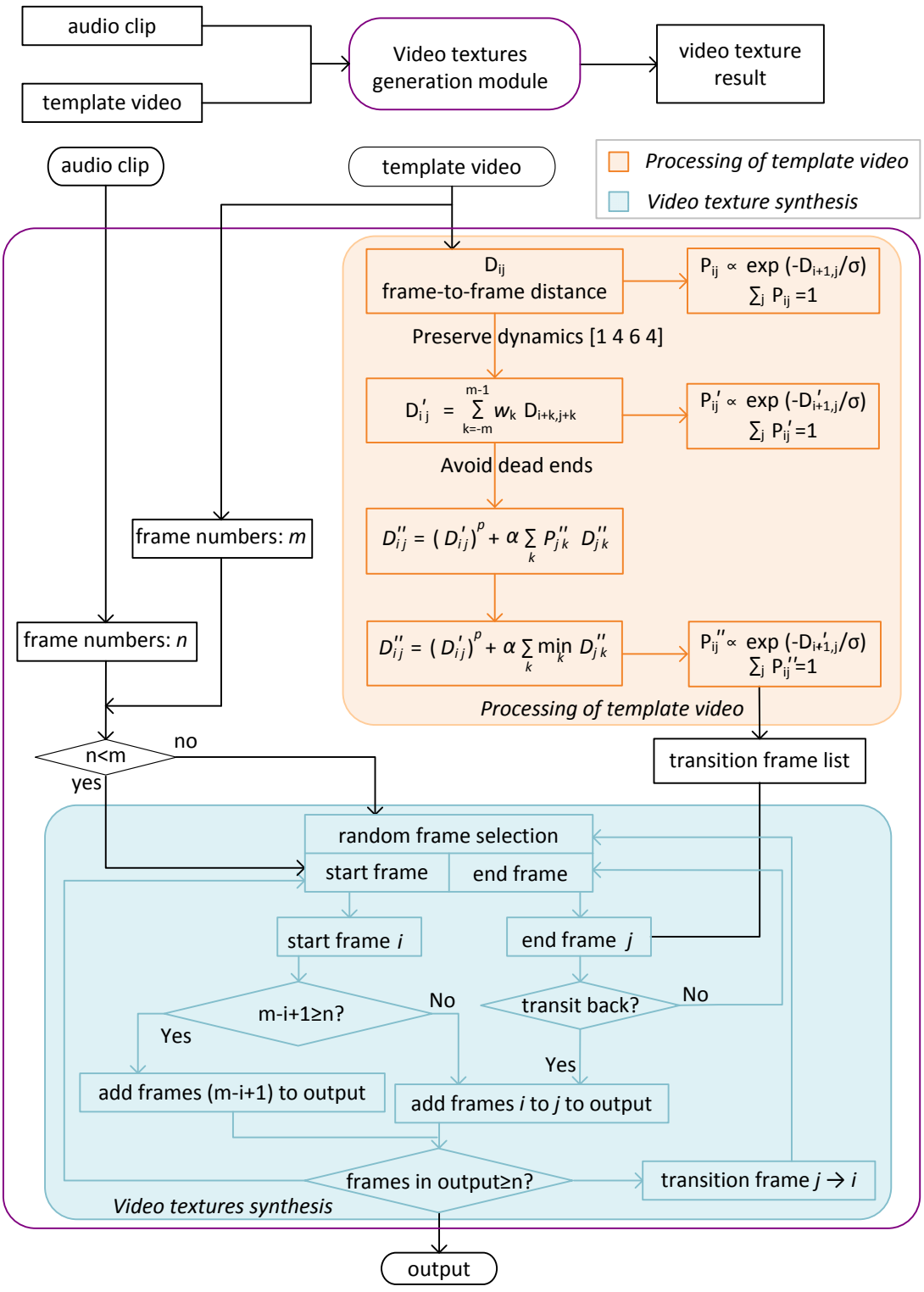

Figure 6-3: Video textures generation method based on random transition.

6.3.2 Video Texture Generation Based on Random Transitions

Schödl's video texture generation method [206] is comprised of three steps: analysis, synthesis and rendering. The first step, analysis, finds fundamental video loop subsets,

transition loops, based on the frame similarities. The second step generates a sequence of frames based these transition loops. Video texture synthesis is then based on a dynamic programming table for optimal loops for any duration. The total computational complexity for the dynamic programming table method is $O(L^2N^2)$, with a space complexity of $O(LN)$, where L is the maximum loop length and N is the number of transitions being considered. The rendering step mitigates any discontinuities in the frame sequence that was determined in the synthesis step.

For our purposes, it is not necessary to follow every detail in Schödl's method [206] strictly because in our system we choose our video carefully, and our video is always continuous. Some of the processing in Schödl's method was designed to accommodate non-continuous video with abrupt changes and is not required. Therefore a simplified video texture technique is proposed for this module that has its foundation in Schödl's method.

The proposed novel video texture generation method consists of a template processing block and a video texture synthesis block (Figure 6-3). The first block processes the template video to find fundamental video transition frame loops and produces a list of transition frames. The number of frames in the output is determined by the input audio clip using equation E6-1. This frame number is fed into the video texture synthesis block, together with the transition list. Based on these inputs, a sequence of frames is generated from the second block.

The extraction of a transition list for template video frames employs the method used by Schödl [206], based on the frame similarity calculation. The first step is to calculate the frame-to-frame distance, which produces a matrix D_{ij}, based on Euclidean distance between any two video frames i and j. To preserve the dynamics of the motion a 4×4 diagonal kernel matrix with weights (1, 4, 6, 4) is used as a filter on the distance matrix D_{ij} to produce matrix D_{ij}'. The final step is designed to avoid dead ends by adding the future cost of each frame to the existing frame sequence. Along with this iteration step, the cost of a transition i is propagated forward to all the frames which are able to do this transition. So the transitions which lead to no graceful exit are eliminated and only the potential graceful transitions remain. After anticipating the future cost of a transition, a final transition cost matrix is calculated as D_{ij}''. After this step, the possibility of each transition is mapped through an exponential function and only the transitions with small

future cost and graceful exit will have high probability values. The probability of the transition is calculated as *P*". The fundamental loops (possible transitions), based on *P*", are found then stored in the output of the template processing block. The list of transition frames is saved together with the template video in the database for later use.

In this list of transition frames, if frame *i* to frame *j* is a possible transition then frame *i* is after frame *j* in the template and frame *i*+1 and frame *j* are sufficiently similar (so there is no obvious jump if frame *j* is played after frame *i*). When a video plays to frame *i*, the system can either play frame *i*+1, or "jump" back to frame *j* after frame *i*. In the transition list, a frame *i* may have more than one transition frame.

A video stream is a sequence of frames, and each frame is a static image. In the resultant video, a frame represents a certain segment of an input audio signal. To generate a video texture that has the same duration as the audio clip and the same frame rate as the template, the number of frames is determined by the duration of the audio input and the input video (frame rate), according to the following equation:

$$frame\_number = duration_{input} \times template\_frame\_number / sec \qquad \text{(E6-1)}$$

If the number of frames for the output video has been calculated to be *n*, then this number *n*, together with the transition frame list from the template processing block, is the input of the video texture synthesis block. There are three functions needed in the simplified video texture synthesis method:

randFrm is a function to randomly select a frame between two given frames (including the two given). It is used to generate start and end frames, and to select next frame for the end frame if there is more than one choice in the transition list.

transFind checks whether or not an input frame can transit to other frames from the transition list. If the frame can transit to other frames, the function returns the index of the given frame. Otherwise, the function tries to find a frame which meets the following three requirements:

1. it can transit to other frames
2. it is located after the input frame in the template video
3. any frame between the input and output frame cannot transit to other frames

If there is no frame meeting all three requirements, the output frame index is 0. ***crsFrm*** generates a new frame using a cross-fading method. It returns a frame generated from two input frames and a coefficient. The resulting frame is used to eliminate the sudden jumps in video texture generation. Although the most similar frame pairs are chosen as the best transitions in video texture generation, some discontinuities cannot be totally avoided because the frame pairs may still have significant differences. Hence the result of ***crsFrm*** is inserted between any transition frames to mitigate the discontinuities.

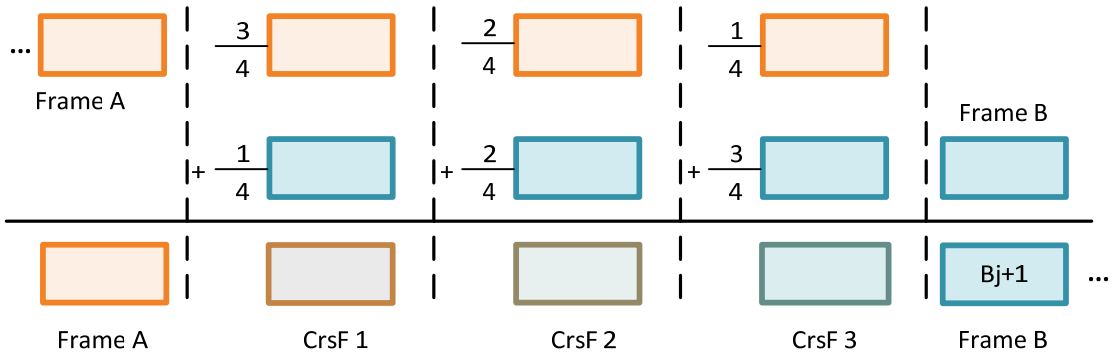

Figure 6-4: Cross-fading method from Frame A to Frame B.

The cross-fading method, that is used to smooth the transitions, is illustrated in Figure 6-4. The jump between frame A and frame B can be replaced by a gradual blend, such as the three frames in Figure 6-4. When frame A transforms to a completely different frame B, three middle frames are produced to avoid sudden discontinuities. These three frames are generated by interpolating the RGB colours of corresponding pixels in order to smoothly transition between frame A and frame B. Cross-fading is used in both video texture generation and the combination of video texture components and will be discussed in Section 6.4.

The coefficient t, as part of the input, determines which of the two input frames has dominant content in the resultant frame. The resultant frame is calculated by $t \times A + (1-t) \times B$. Values of t can be 0.75, 0.5, and 0.25, depending on the position of the frame in the transition. If the frame number of a template is more than 30, which means the template lasts more than one second, each transition will have three cross-fading frames (t=0.75, 0.5, and 0.25). When the template video is less than one second, only one cross-fading frame is needed for any transition (t=0.5).

In the synthesis part of video texture generation, functions ***randFrm*** and ***transFind*** are used to find a start frame and an end frame from the template video. The ***randFrm*** is used to choose a start frame number from the first half of template frames. After obtaining the start frame, the end frame is found between the start frame and the end of the template video. This randomly selected end frame f_i may not be able to transit back so it is not a suitable choice to be the end frame. The function ***transFind*** is employed to find the suitable frame f_j which, of the transferable frames, is the closest frame to f_i. From the description of ***transFind***, $f_j = f_i$ if f_i is able to transit back.

Figure 6-5 is the pseudo code for video texture synthesis. Suppose the frame number of the template is *m*, and the frame number for the resultant video texture component is *n*:

1. When $m>n$, which means the template video has more frames than what is needed for the output, the random frame selection function ***randFrm*** is used to generate a start frame number *i* from one to the frame (m-n). Then frames between *i* and (i+n-1) are output video.
2. When m≤n, there are not enough frames for the output video. In this case the output video is generated by the video texture technique. Simplified video texture synthesis is performed according to the following steps:
    a. Set 0 to the frame number *k* of the output video. Generate random start frame *s* by using ***randFrm***.
    b. Generate end frame *e* by using ***randFrm*** and ***transFind***. Frames between the start and end frame are appended to the output video. k=k+e-s+1.
    c. If $k<n$, obtain transition frames for end frame *e* from the transition list. If there is more than one transition frame, select one randomly. Suppose the transition frame is *i*, set it be the new start frame s=i.
    d. Generate cross-fading frames based on frame *e* and frame *s*. If there are more than 30 frames in the template, three frames with coefficient values 0.75, 0.5, and 0.25 are generated and appended to the output video. When the template frame number is less than 30, one frame with t=0.5 is generated and appended instead of three frames.
    e. If the frames after the new start frame in the template are more than the frames needed by the output video these frames are appended to the output video. Otherwise, repeat the steps b, c, d and e until the output video has enough frames.

```
Pseudocode for "video textures generation"
n: Frame number for the output video clip
m: Frame number for the template video
r: CALL randFrm()

Set outputVideo to Empty
IF n < m
        startFrm = randFrm(1,m-n)
        output = template(startFrm: startFrm+n-1)
ELSE
        Set FrmNum = 0;
        REPEAT
                startFrm = randFrm(1,m)
                crosFrm = crsFrm(output(FrmNum), Frm(startFrm));
                append crosFrm to output;
                IF (m-startFrm)>(n-FrmNum)
                        append template(startFrm:n-FrmNum) to output
                        FrmNum = n
                ELSE
                        REPEAT
                                endFrm = randFrm(startFrm+1:m)
                                endFrm = transFind(endFrm)
                        UNTIL (endFrm > 0)
                        append template(startFrm:endFrm) to output
                        FrmNum = FrmNum + (endFrm-startFrm)+1;
                END IF
        UNTIL FrmNum>= n
END
```

Figure 6-5: Pseudo code for video texture synthesis based on random transitions and three functions.

There is no rendering step in this video texture generation method. As Schödl noted [206], video texture generated by random play has quite poor results with many visible jumps and discontinuations. By inserting the cross-fading frames between transition frames, and selecting the template video carefully, this problem is mitigated and the resultant video textures are improved.

6.3.3 Adaptive Video Texture Generation Method Based on Audio Matching

The video texture generation method based on random transitions discussed in the preceding section can create a unique, continuous and interesting video clip for a given audio file. But the method requires that all motions in the template video are able to represent the input audio. In other words, any small piece of sound in the class should be represented by the template video correctly and accurately. For example, in a

template video for class "speech", all frames should be able to represent speech. If a video clip for face is used as the template, the mouth in the face should keep moving. But in any speech there may be short breaks between sentences. Although in the segmentation module and the classification module, these short silence periods have been removed, they will be included in the input for the video texture generation process. The inclusion of silence periods is required to ensure that the video texture and its corresponding audio are of the same duration and are synchronised.

For the silence periods within an audio clip, the template video cannot generate suitable representative video frames, especially when the silence duration is long in comparison with the duration of the sentences. Moreover, silence periods between two adjacent audio clips pose difficulties for visualization with video textures. The silence period needs to be either split into two pieces at the middle, and each piece joined to its neighbouring audio clip, or be represented by cross-fading frames generated from the edge frames of the two video texture components for the audio clips. But neither of these two methods works properly in practice. If the silence piece is joined to the audio clip, the template cannot represent the silence correctly. But representing a silence period by frames generated by cross-fading of the last frame in the preceding video textures clip and the first frame in the following one is not suitable either. When the two neighbouring audio clips belong to different classes and the content of the video frames is quite different, the cross-faded frames become blurred and meaningless.

This section presents an adaptive approach to derive video textures from template audio/video sets. Compared with the normal video texture generation method, this approach better suits the audio visualization system because its resultant video clip can represent both audio signals and acoustical silence periods. The difference between normal video texture generation and the adaptive method is that in the adaptive method, the input video contains not only video frames but also corresponding audio (sound and acoustical silence).

If the output video has the same duration as the input audio, the number of frames can be calculated by the equation E6-1 from both audio input and the frame rate of the video template. As with the simplified video textures method, this method also uses a template processing block and an input audio processing block. The best frames can be found to

represent a given audio because there is an additional audio matching block included in our system that compares audio features in order to find the best matched segments.

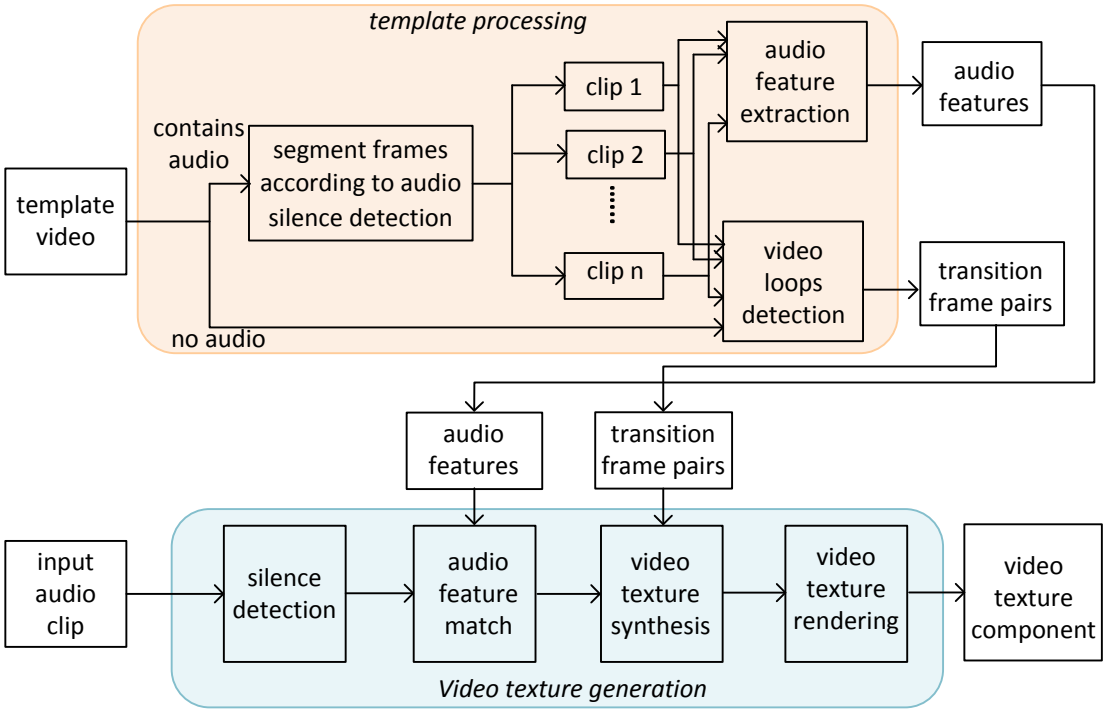

Figure 6-6: Framework for adaptive video textures generation method.

Figure 6-6 depicts the framework for the adaptive video texture generation method. The upper portion of Figure 6-6 shows how the video template is processed. For an input template file which is comprised of video/audio pairs, the audio part is separated into the same number of frames as the video of the template. Then the template is separated into segments based on silence detection of the audio frames. Their corresponding video frames in the template video are separated into video segments; each video clip has the same duration as the associated sound clip. Finally, the video frames for each segment are processed to generate a transition list for future video texture synthesis. The transition list is generated in the same way as discussed in the simplified video texture generation method of Subsection 6.3.2. To make sure a video segment (a sequence of video frames) can be used to generate an interesting video texture component, it needs to have a certain number of frames. In the audio visualization system, it is pre-determined that 30 frames, (one second) is the shortest video segment for video texture generation. The audio feature vector of each sound clip in the template is calculated, and this is used to match the new input audio clip in the processing of input audio.

Here there is an audio feature match step between the target audio segment and template audio segment, to find the best matched video frames. After finding the suitable video frames, the last step is generating video texture output using the simplified video texture method described in Subsection 6.3.2. Figure 6-7 gives an example of a result from the template processing block where the audio file is of a dog barking.

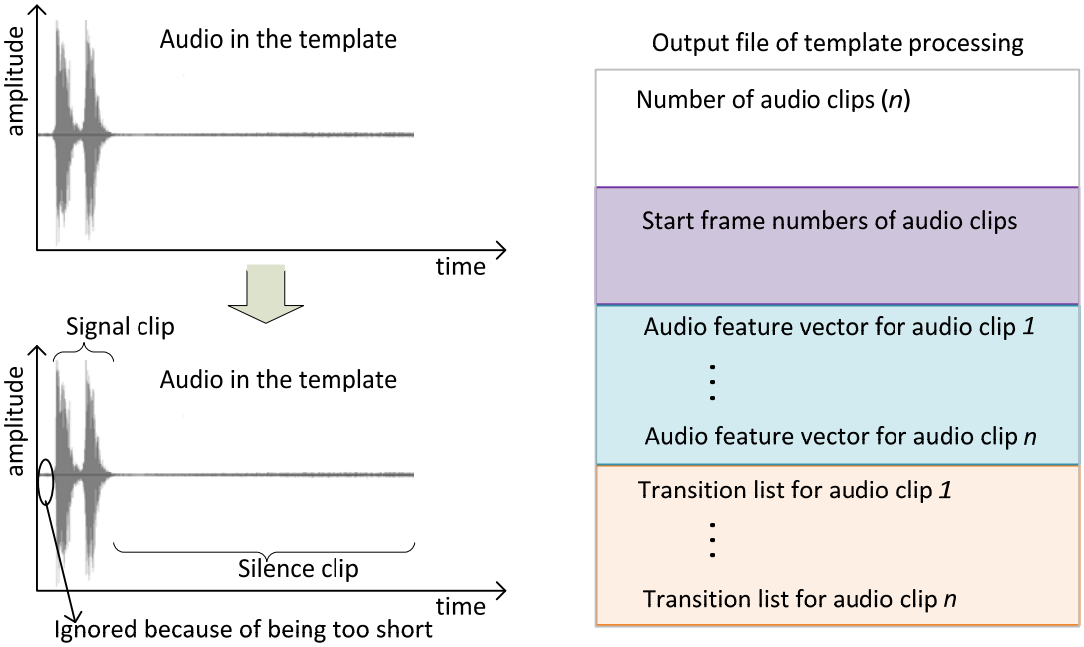

Figure 6-7: Example of a template processing result.

Both the audio and the visual components of the template are separated into frames. The frames for the audio component are named *AudioF*, and *VisualF* is the name for the visual part of the template. *AudioF* and *VisualF* are associated so that each *AudioF* can be represented by its corresponding *VisualF*.

The result of the template processing block is a file that contains information for the audio-video pairs after audio segmentation based on silence detection and extraction of the fundamental frame loops. The four components for the template processing result are shown in Figure 6-7: the number of audio clips, their starts/ends, their audio feature vectors, and their transition frame lists.

The audio feature vector used is a normalized twelve feature vector comprised of the Total Spectrum Power, Brightness, Bandwidth, Pitch, ZCR and 8 order Mel Frequency Cepstral Coefficients (MFCCs). This is the same procedure as that described in the

segmentation chapter. For each feature, the values of the audio frames are mapped to the range [0, 1] so that each feature has the same weight in the distance calculation.

In the given example of a dog bark, the silence detection finds three audio clips in the template audio. The first and the third are silence periods and the second is a signal clip. Because the first silence clip is shorter than one second, the corresponding video frames are not used for video texture generation, and will therefore not be shown in the result. An example input audio is given in Figure 6-8. This example is used for the following discussion on how the audio match block finds suitable frames and generates a video texture to represent a given audio file.

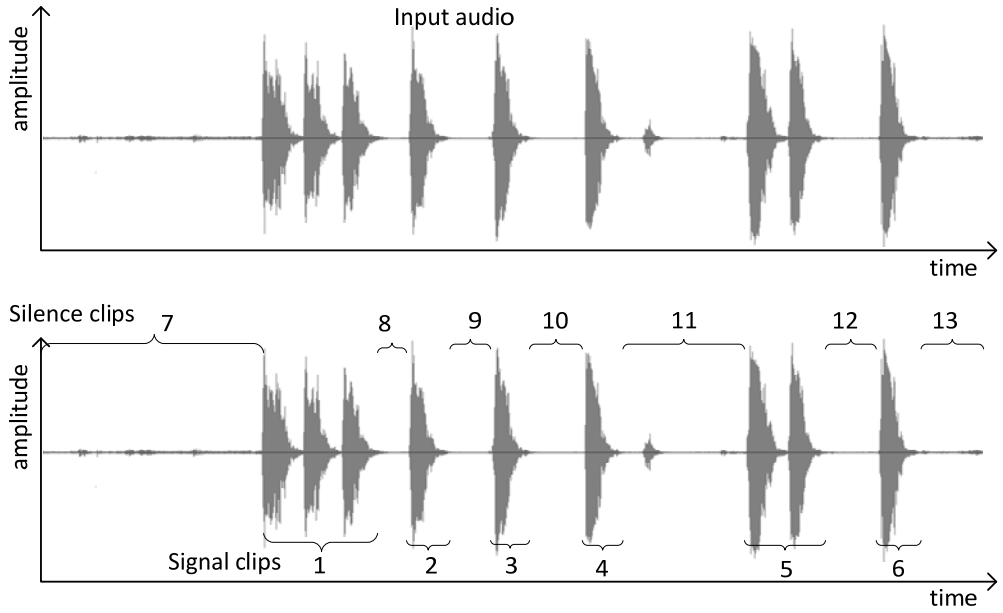

Figure 6-8: An example input audio file (dog barking).

When the example audio input was segmented using silence detection thirteen segments were found (Figure 6-8 (bottom)). The signal clips are marked as clip one to clip six, and the remaining clips are periods of silence.

In the audio matching step, there are three possible methods which could be used to find suitable video frames for an audio piece: *frame-based likelihood*, *feature vector curve similarity* and *nearest neighbouring feature vector*. The *frame-based likelihood* method separates input audio clips into frames and extracts the audio feature vectors for each frame. Then, for each frame, it finds a frame from the template that has the most similar audio features. The *feature vector curve similarity* method not only considers a single

frame, but also the dynamics of audio features. It can be considered to be an audio feature curve fitting in high-dimensional audio feature space. The *nearest neighbouring feature vector* method extracts audio features that are based on audio clips instead of frames. It then finds the nearest audio vector (by Euclidean distance) from the template.

The template file in Figure 6-7 and the signal clip five of the input audio file in Figure 6-8 are used to show why the two methods, *frame-based likelihood* and *feature vector curve similarity*, are not suitable for audio matching. Adaptive video texture generation based on the audio matching method requires that an input audio segment is visualized by the most acoustically similar segments in the template. If a segment in the audio input contains sound signals, it is represented by the signal frames in the template. The frames of a silence period in the template are used to represent the silence segments in the audio input. Although the *frame-based likelihood* and *feature vector curve similarity* methods can meet these requirements, the resultant video textures either flicker or do not acoustically match the audio input. Only the *nearest neighbouring feature vector* method is suitable and can be employed to find suitable video frames for audio pieces in the audio matching block of our proposed method for the visual representation of audio.

***Frame-based likelihood* method**

The *Frame-based likelihood* method represents an audio input by matching each frame in it with the frames in the template. The given audio input is separated into frames *InputF* of the same duration as *AudioF*. Audio features are extracted for the *AudioF* and *InputF*. Each *InputF* can find a most similar frame from the *AudioF* using Euclidean distance. For an *InputF* IF_i, suppose its most similar *AudioF* is AF_i, which has an associated *VisualF* VF_i. When compared with other *VisualF*, VF_i should be the best one to represent IF_i. When all the best *VisualF* are found and connected using the sequence of their corresponding *AudioF*, they generate a video to represent the audio input. The assumption in this method is that in the template, the video frame can successfully represent its corresponding audio signal, and the content of a given input audio file could also be represented with the template video. If a video frame is not needed for the given input audio, its corresponding audio signal should not appear.

We also note that this method depends on the continuity of audio feature vectors. However, the audio feature vectors for adjacent audio frames are not always continuous for video dynamics.

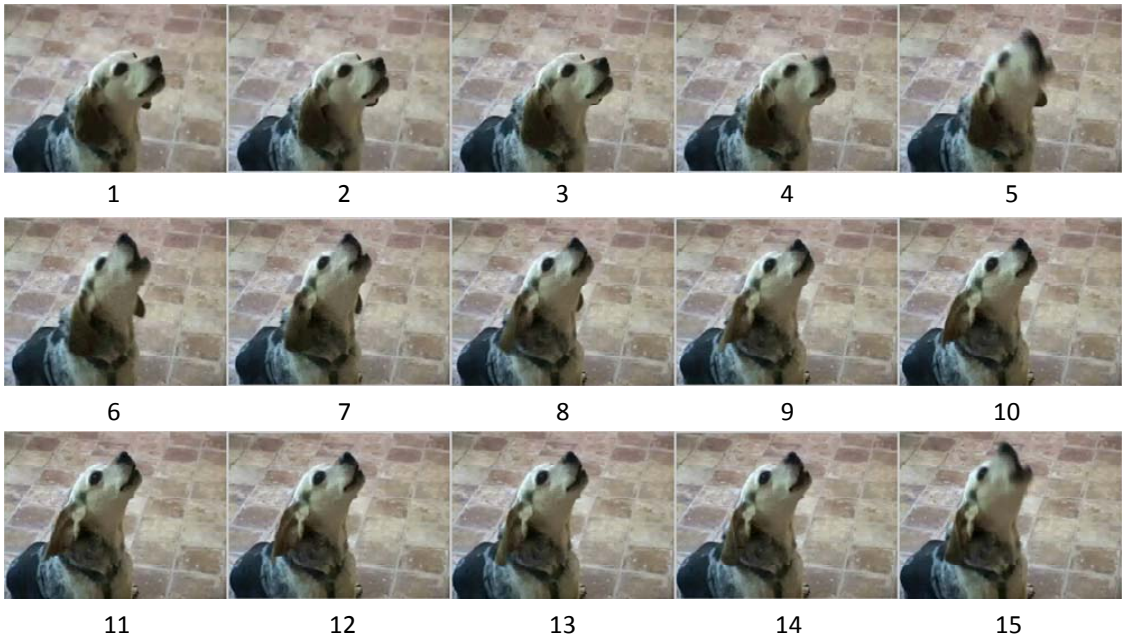

Figure 6-9: Frame images from the template video.

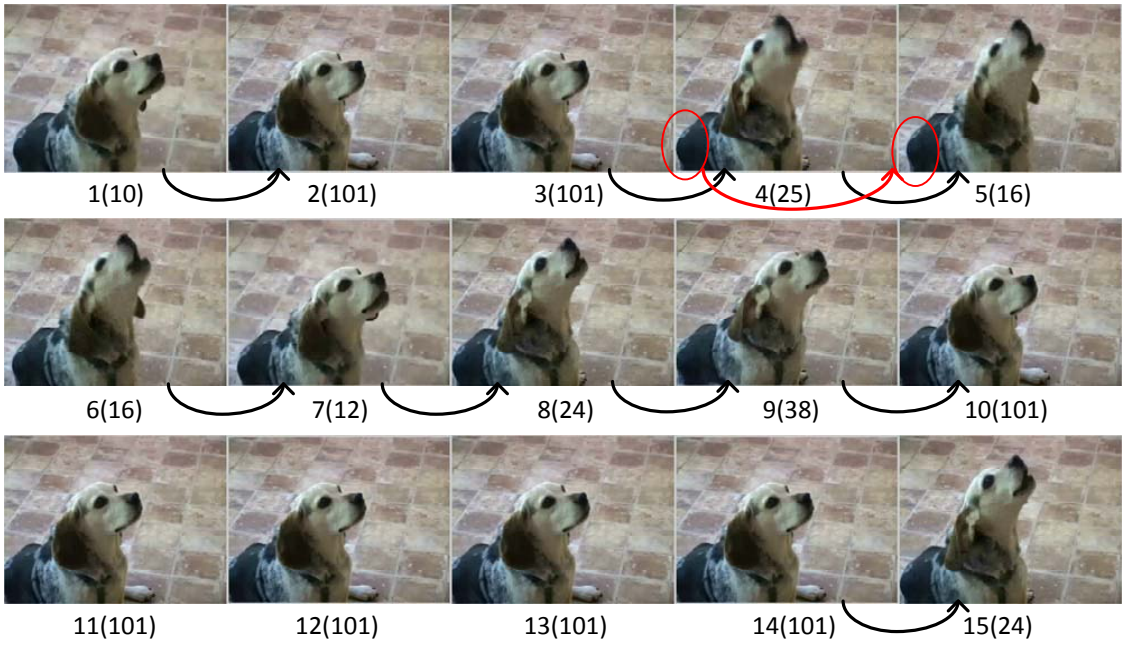

Figure 6-10: Resultant frames by *Frame-based likelihood* method. The numbers X(Y) where X is the frame number in the given input audio file and Y is the frame number in the template which has the closest audio features to the frame in the template. The black arrows and red circles show the incontinuity between frames.

Figure 6-9 contains frames for the signal clips in the template. They are continuous and represent a dog barking. Moreover, these frame images are continuous with both object motion and the position. When the template video frames shown in Figure 6-9 are played, the dog stays in the same position. From frame 1 to frame 4, it raises its head. The dog barks from frame 5 to frame 9. Then it barks again from frame 15. The frames

are therefore considered to be continuous and to represent whole motions. The resultant video should have similar quality. However the *frame-based likelihood* method does not give satisfactory results (see Figure 6-10) and produces discontinuities in position and motion.

Figure 6-10 shows the frames extracted using the *frame-based likelihood* method for signal clip 5 of Figure 6-8. The audio frames are from 1 to 15. The numbers of their corresponding frames (best matched audio frames) in the template are marked in the brackets together with the video frames in Figure 6-10. From the frame numbers and the frame images, it can be seen that there are a number of obvious jumps between adjacent frames. These are illustrated by arrows in the figure. Frames 4 and 5 show the same motion (dog barking) but the dog is in a different position in each frame (this difference is highlighted using red ovals). These frames although placed sequentially in the video texture are clearly from two different motions within the template video. Connecting these frames will cause the video to flicker. The quality of the template can be compared with the quality of the sequence of fifteen frame images for a similar audio piece shown in Figure 6-9 by examining the discontinuities that are apparent between adjoining frames. If the frames in Figure 6-10 are played, the transition from frame 1 to frame 2 is discontinuous because the dogs in the two frames are in different positions. This discontinuity can lead to an obvious flicker. Frames 2 and 3 do not change as they are exactly the same. Then the transition from frames 3 to 4 has a discontinuity in both the object's position and the object's motion (the dog in frame 3 is silent and in frame 4 it is barking). The frame 4 to frame 5 transition results in a flickering due to the abrupt shift in the object's position. If we consider the transition between two adjacent frames, a discontinuity exists in eight of the transitions (Figure 6-10). The remaining six continuous transitions exist because the adjacent frames are exactly the same. Same frames do not represent a motion so ideally they should be avoided in video texture generation. The video texture result illustrated in Figure 6-10, generated using frame-based likelihood, does not produce continuous video texture and this method is therefore not suitable for our visualization.

***Feature vector curve similarity* method**

The *feature vector curve similarity* method is used to resolve the problem of sudden jumps that exists in the *frame-based likelihood* method. The main idea behind this method is the ability to retain the dynamics of the audio/video frames by matching a

sequence of frames instead of a single frame. It is equivalent to a polygonal discrete curve fitting in high-dimensional audio feature space. The input audio (*InputF*) is separated into frames and the audio features of each frame are extracted using the same methods as was employed in the *frame-based likelihood* method. The dimension of audio feature space is the same as the number of audio features selected for matching. The audio feature vector for each frame in either the video template (*AudioF*) or the audio input (*InputF*) is a point in this high-dimensional audio feature space.

The sequence of audio feature vectors for all frames in the template (*AudioF*) forms a discrete curve in the audio feature space, which is called the *source curve*. Again the template is the original video *VisualF* with its original, concurrently recorded audio *AudioF*. The audio input, *InputF*, is separated into sections each containing a number of user defined frames. A comparison between a section of *InputF* (*target curve*) and a sliding window section along the *AudioF source curve* is performed using a curve fitting method in audio feature space (Figure 6-11). In our experiment, we used sections that were comprised of ten frames each of 0.033 second duration (30 fps). The sliding window on the *source curve* was also 30 fps giving it the same size as the *target curve* sections. The feature vectors of each 30 fps section also form a discrete curve in the high-dimensional audio feature space. Combining these section curves forms an audio feature vector curve called the *target curve*.

The aim of this method is then to find the best matching section on the *source curve* so that the most appropriate portion of video is chosen to represent that section of the input audio file (*InputF*). This best matching piece of curve is called the *matched curve* in this *feature vector curve similarity* method. In this approach we use the Fréchet distance [228] [229] to measure the similarity between the two curves taking into consideration the location and order of the points on the curves. So the *matched curve* has the minimum Fréchet distance to the *target curve* when it is compared with any other curve section from the source curve. An example of the distance between two curves is illustrated in Figure 6-11.

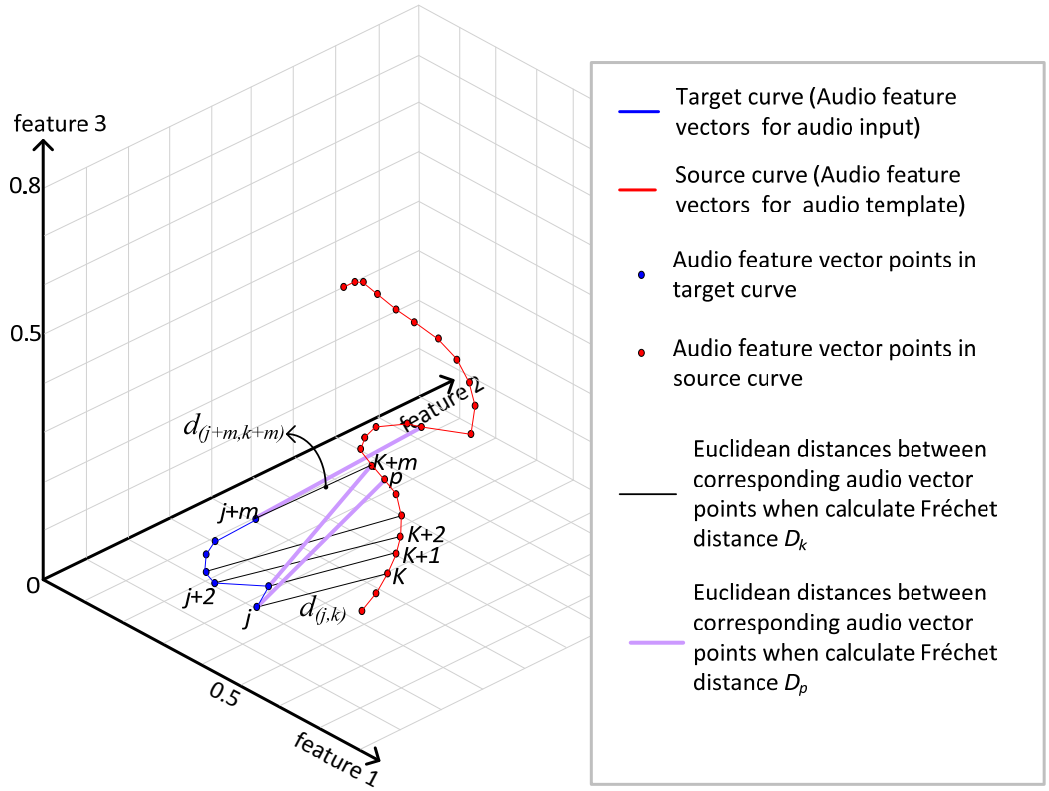

Figure 6-11: 3D *feature vector curve similarity* distance calculation.

This figure is illustrated in three dimensional feature space for clarity. The blue curve represents a *target curve* and the *source curve* is in red. Suppose the section *target curve* starts at j^{th} frame in the audio input curve and the frame number in the *target curve* is *m+1* (from frame *j* to frame *j+m* in the audio input curve). Then the Fréchet distance D_k between the target curve and a piece of source curve (starting at point *k*) is the sum of the Euclidean distances between corresponding points in the target curve and the source curve:

$$D_k = \sum_{i=0}^{m} d_{(j+i,k+i)} \qquad \text{E6-2}$$

where $d_{(j+i,k+i)}$ is the Euclidean distance between the point *j+i* on the *target curve* to the point *k+i* on the *source curve* and *k* is the start point of the sliding window along the *source curve*. The sliding window starts at the first point on the *source curve* and moves along the *source curve*, the values of D_k form a curve of the Fréchet distance between the *target curve* and each section on the *source curve*. The minimum Fréchet distance value gives the starting point on the *source curve* for the best matched section of the *target curve* (Figure 6-12).

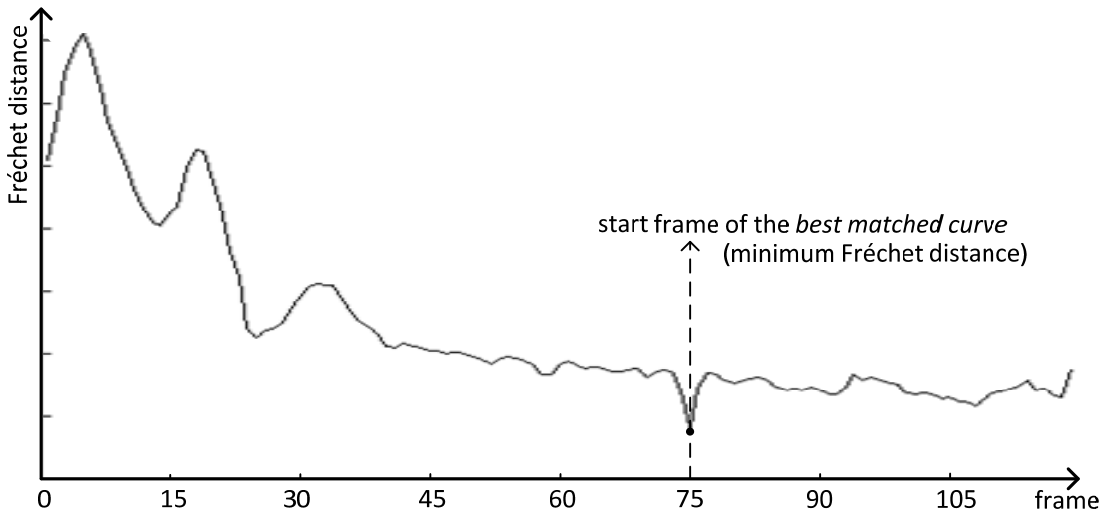

Figure 6-12: Similarity distance curve of *feature vector curve similarity*.

The template video (*VisualF* and *AudioF*) and audio input file (*InputF*) used in this experiment is the same file as was used to evaluate the *frame-based likelihood* method. In this example, there are 150 frames in the source curve and ten frames in the target curve. In the matched curve, there are 141 points. The remaining points in the source curve are not used because there are not enough frames in the target curve (the target curve must always be shorter than the source curve). The start of the matched curve in this template is frame 75 because the minimum Fréchet distance occurs at frame 75. However, the corresponding video frames in the template (video frame 75 to video frame 85) are not appropriate for the input audio piece, where a dog is barking, because these video frames show motions that represent silence (e.g. the dog's head moves but the mouth does not).

To give a clear idea of the difference, the magnitudes of the audio feature vectors in the target curve and source curve are illustrated in Figure 6-13. Although the curves in Figure 6-13 cannot represent the location of the curves in high-dimensional audio feature space, they can at least illustrate the silence period and the signal period. The absolute magnitude of a frame in a silence period is not zero because some audio features are scaled to non-zero values after normalization (see the discussion of normalization in audio segmentation in Section 3.5). The blue line in Figure 6-13 represents *InputF* the section of the *target curve* that is being matched to the *source curve*. The red line represents the *source curve* and from frame 40-120 clearly shows a silence period in the *source curve* but the *target curve* clearly is not composed of silence but instead has two distinct audio signals. This means that the section of video chosen to

represent the target curve section *InputF* will not be representative of the sounds in the audio as discussed in the preceding paragraph.

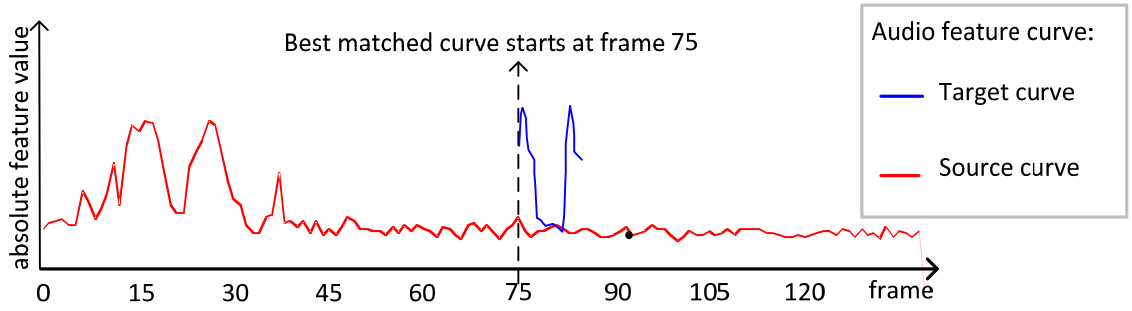

Figure 6-13: Absolute length for audio feature vectors in template and target audio piece.

From what discussed above, this method does not work for the following three reasons:

- The absolute magnitudes of the value target curve (see the blue curve in Figure 6-13) show that if the input audio file is divided into small pieces of fixed duration, it is not practical to ensure that each piece contains a single and complete sound.
- If each of the signal pieces mentioned above contains a short period of signal in the template but with different duration, there is a high possibility it will be matched incorrectly because their audio feature vector numbers are different.
- It is hard to find a suitable size for any given signal piece.

Therefore, although it seems that the *feature vector curve similarity* method can preserve dynamics and find the best matched audio piece from the template, this method cannot always locate the suitable video frames successfully. Another method is required to find the matched audio piece by the audio feature similarities from the template. The method should be able to solve the discontinuity problem in the *frame-based likelihood* method and the inability to locate audio pieces by the *feature vector curve similarity* method.

Audio Match using Segments

Only the audio matching based on segmentation method is able to find the most suitable frames from the template. This method contains several processes including segmentation, audio matching and video texture generation method (discussed in 6.3.2).

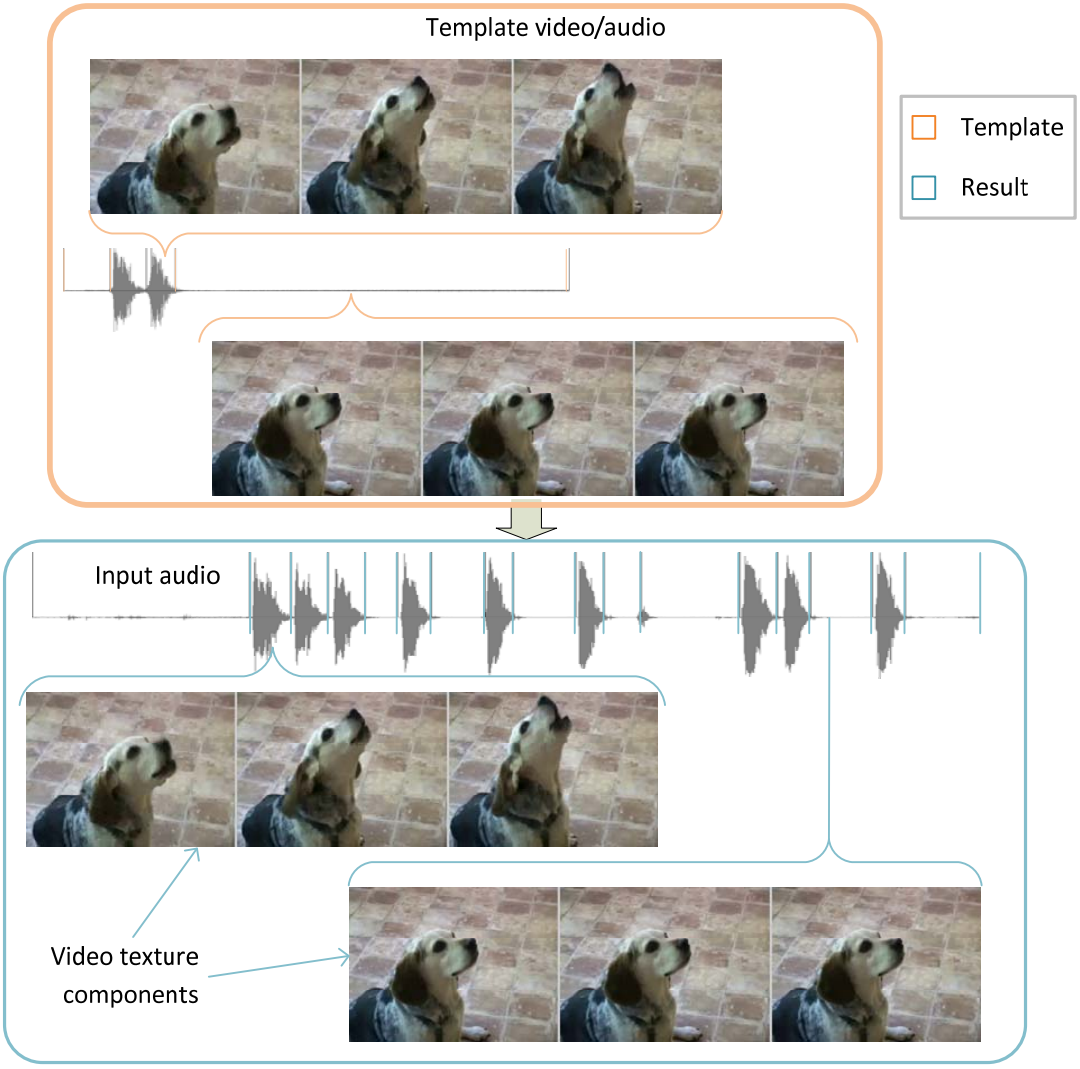

Figure 6-14: An example of visualizing audio with video textures.

To make the resultant video texture smoother and more continuous, the method described in 6.3.2 is modified. The example in Figure 6-8 is used to illustrate the details. For the audio input, the first video texture component is generated for silence clip 7, which is the first audio clip found using segmentation based on silence detection. A video texture component is generated from the silence clip in the template (Figure 6-7). It is the first component for the final resultant video texture. After this component, a video texture component is needed to visualize the second segment of the audio input, which is marked as *signal clip 1* in Figure 6-8. To make these two components transition smoothly, the last frame of the preceding video texture component should be similar to the first frame in the following video texture component. Two adjoining segments, *silence clip 7* and *signal clip 1* in Figure 6-8, are used as examples to explain the way a smooth transition between video texture components is achieved. Suppose the last frame of the video texture component for segment *silence clip 7* is frame *i*. When a

video texture component for *signal clip 1* is generated from the signal clip frames in the template, the most similar frame *j* to frame *i* is found using Euclidean distance. This frame *j* is selected to be the start of the video texture component for *signal clip 1* in the audio input. When a cross-fading frame is inserted between them the two components can transit without a visible jump. Following this rule, any video texture component starts with a frame that is the most similar to the end frame of the video texture component in front of it.

Figure 6-14 shows a video texture generated using this method. The sound segments are denoted by the audio clip indicator at the bottom of the image with representative images of the video file extracted from the template video/audio pair at the top of Figure 6-14.

This adaptive method can be extended to multiple sounds in the template video/audio too. For example, the template for class "baby's laugh" can include a sound of "baby laugh" and a different sound "baby giggle". Figure 6-15 is an example with only baby's laugh in the input audio file. After segmentation, there is only one audio clip in the input sound. So the result is one video texture component generated using the closest audio clip in the template audio/video pair.

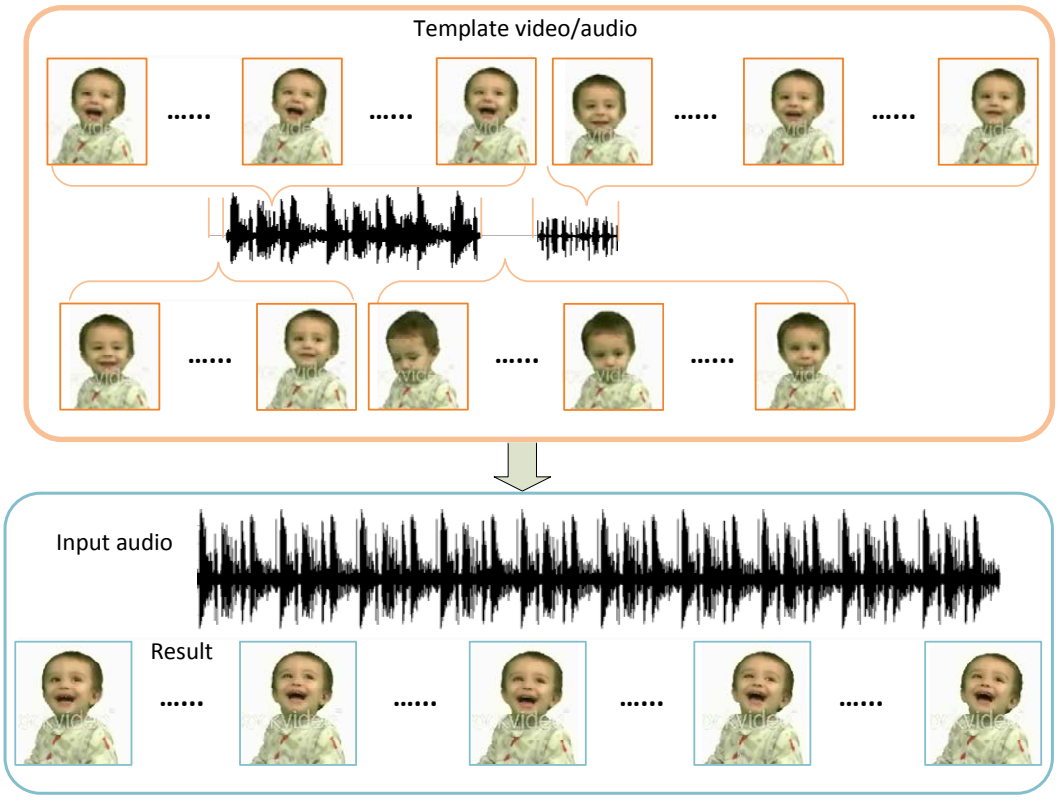

Figure 6-15: Single sound input result.

6.3.4 Template Video Limitations

In the preceding subsections, the discussion is based on the assumption that the input template videos are suitable for video texture generation. But not all videos can be used as templates for the generation of video texture. In addition, some audio files are not suitable for representation by video textures. This section and Subsection 6.3.5 will discuss the requirements for, and limitations of, video texture generation.

Video textures leverage a finite set of video clips to generate a non-repeating and infinite stream of video. The template videos determine the quality of the final resultant video texture. Typically the new video sequence is generated by re-sequencing frames from the finite video source in such a way that the perception of the transitions between two sequences is minimized. Phillips and Watson [230] pointed out that the key problem of video texture generation is transition frame detection. Video texture techniques work only if suitable transition frame pairs can be found in the input template video. The transition frames must look similar enough, in other words, the motions in the video must be periodic or quasi-periodic. Otherwise the results either have illogical changes in motion, or have obvious abrupt jumps (not smooth between transitions). The quality of the resultant video texture clips relies on the similarity between the transition frame pairs selected from the original video, and the completeness of motions. A sequence of frames representing a complete motion, which means that the first frame and the last frame are similar and can be used as transition pairs, is named a fundamental loop in video texture generation.

Because of the requirements of appropriate transition frames, not all videos are suitable for producing video textures. The selection of suitable input video for video texture generation has been discussed by Phillips and Watson [230]. They pointed out that suitable input video should have large dynamic details and at the same time have enough transition frame pairs to enable the formation of the fundamental loops in the video. A proper candidate for video textures can be described by its distance matrix between frames. The distance matrix is generated in the way described by Schödl [206] and discussed in Section 6.3.2. To view the distances between frames, the distance matrix is scaled to a grey value image. Figure 6-16 gives some examples of input video structures which are possible candidates for the generation of video textures.

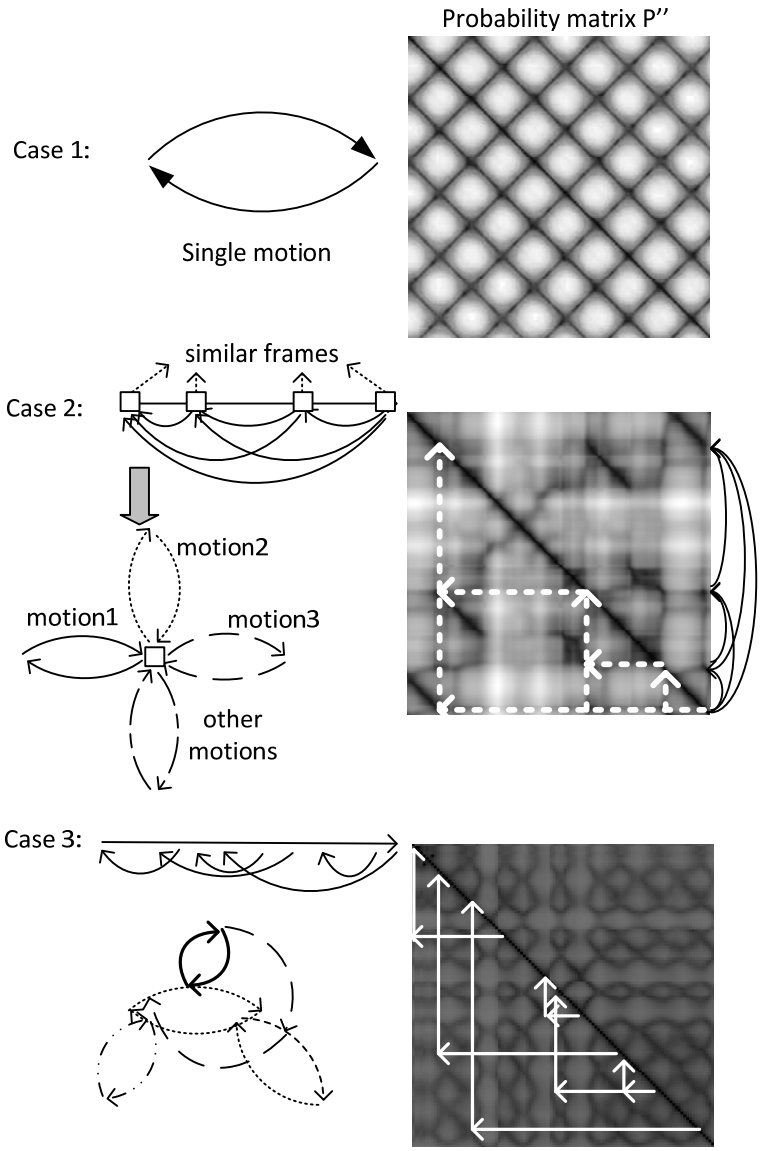

Figure 6-16: Structures of input videos which are suitable for video textures generation.

Case 1 in Figure 6-16 has a single motion which makes smooth transition video textures, but the content is limited and simple (e.g. the clock sequence in [206]).

Case 2 includes key frames which are marked "similar frames" in the figure. The video has multiple motions with each motion starting and ending at a key frame. Video texture components based on this kind of video can be smooth and the resulting contents are better than those from case 1. But in the resultant video texture sequence, a transition only happens when a motion is finished, so a transition can not happen until a motion is completed.

Case 3 has a complicated structure. There is no global key frame for all the motions. Each motion is a loop of frames and the transition frame pairs may be located at any

position of a fundamental video loop. In other words, there are overlapping frames between motions. This kind of video is ideal for generating various interesting video textures.

Figure 6-17 has two structures that are not suitable for video texture generation. The left image shows the distance between frames from which there is no obvious loop. This may happen when the motion in the template video does not repeat itself, or there are some transition frame pairs but these frames are too limited. The left image of Figure 6-17 is a similarity map of frames in a video that has no suitable transition pair. Each frame in it has to be followed by the next frame in the video. As discussed above, if there are no transition frame pairs, a video texture cannot be generated. The right image in Figure 6-17 is a distance matrix that has multiple fundamental video loops but these loops are separated. Smooth transitions are possible in any single loop but there is no global transition for the whole video. Because no two loops have a transition frame pair there can be no smooth transition between loops. When the video plays to the last loop, it will only transit within the frames of the last fundamental loop. In other words, the video texture result will be limited to a certain set of frames.

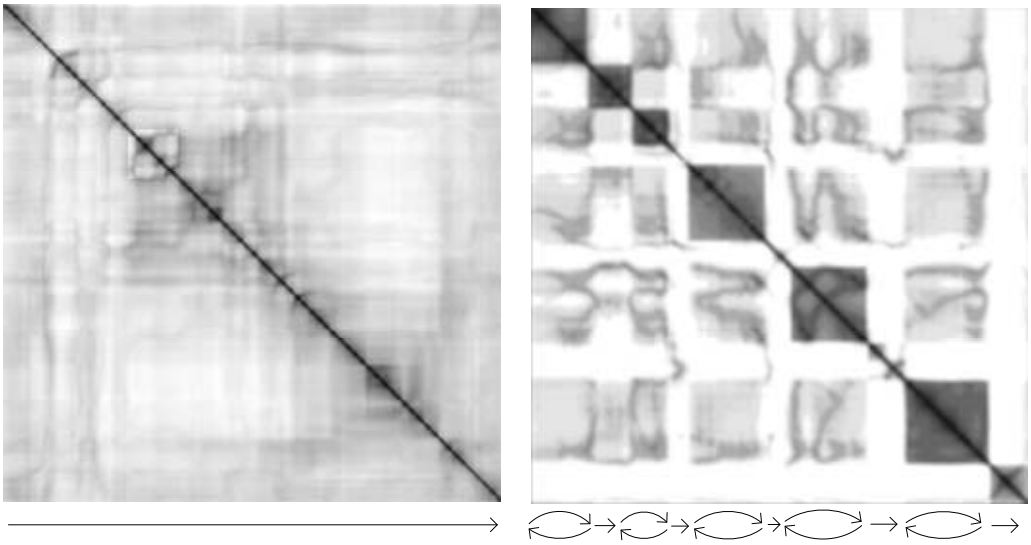

Figure 6-17: Structures of video frames which are not suitable for video textures generation.

When the input video contains only small detail movements, which always have a homogeneous similarity map, it is also not suitable for video texture generation because the transition frame pairs will be too numerous. The results tend to be similar to a random re-ordering of the input video frames. Figure 6-18 gives an example for such a video containing only similar frames. The left image represents the distance matrix of

the frames. There is no clear structured motion within it. This distance matrix map shows that there is no good transition for video texture generation. If the frames are compared with a totally blank frame (as in the distance map at the right in Figure 6-18) the distances between each pair of frames are very small. Any two frames are similar enough to be a transition frame pair. The resultant video textures would look as though it had been generated from a single image because users would not be able to differentiate among frames.

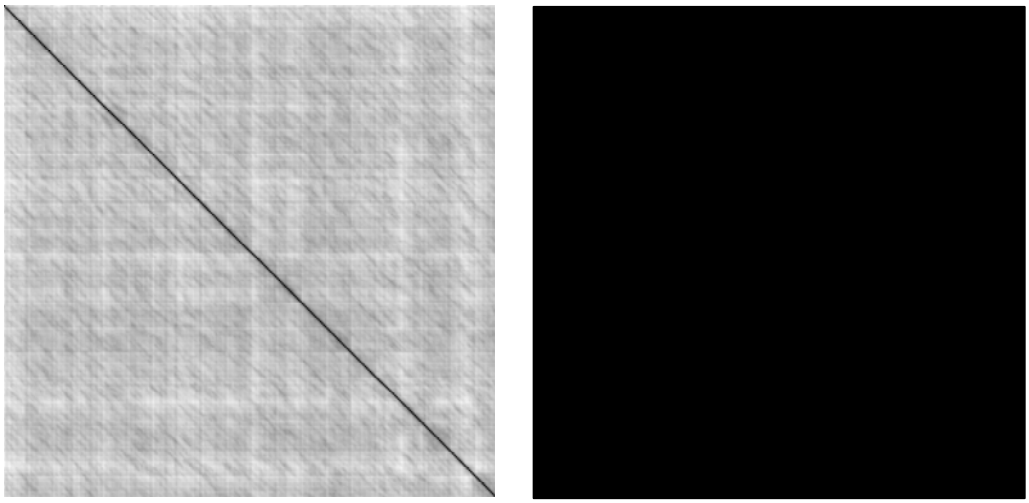

Figure 6-18: Random structure of input video.

When a video has a an inappropriate structure for being a video texture template, as discussed above, the possible transition frame for any frame in it is 0. In other words, none of them can "transit back".

Besides the existence of transition frame pairs, the quality of the template video is also a factor that needs to be considered. Not only the resolution and frame size, but even some minor factors can affect the quality. When the audio/video database is built, whether or not the content of a video can thoroughly represent all the sounds in the class is also an important criterion.

With the limitations discussed above, the template video should have multiple dynamics (different motions) and, at the same time, enough transition frame pairs.

6.3.5 Video Texture Clip Duration

The input audio file needs to be suitable for video texture generation. To generate interesting and meaningful video texture components the input audio clips represented by those components must be appropriate. If any of the audio clips in the input file are very short, the video texture component for this audio clip would be also very short unless users manually extended the output video texture. But extending short sounds leads to misrepresentations of actual sound length in the input audio. Moreover, because of the video texture generation techniques, the cross-fading frames would make up the majority of the video texture components. This is not a desirable result for the proposed audio visualization system. Hence an empirical length of 30 frames is the minimum duration for visualization. Only sounds lasting more than 1 minute (30 frames) are suitable to be represented by video texture. If an audio clip has a sufficient duration (more than one second) a video texture, based on the template of the class the audio clip belongs to, is generated to represent that audio clip. For the sounds that are very short (e.g. less than 20 seconds), video texture is not suitable. In the audio visualization system, a template image such as that mentioned in the Time Mosaics chapter is employed instead of video texture to represent a very short audio clip.

In the preceding subsection the process of calculating the number of frames needed in the resultant video texture component for an input audio clip was discussed. The output video texture lasts the same time as the audio input. However, when the audio input lasts a long period and the audio features do not change frequently, it takes a long time to visually scan its video texture but there is little change in the image. To be more flexible, the video texture method can be used to visually "summarize" the audio input. For example, using one frame in the resultant video texture to represent a piece of audio file which would otherwise have more frames. The number of frames needed in the output video depends on the various requirements of different purposes. The "summarized" video texture can visually illustrate the time-based audio features in a shorter period.

6.3.6 Mapping Audio Features

Because the video can be treated as a sequence of frames, which are the same as images, the audio feature mapping can use the same method as that used in time mosaics. Time

mosaics differ from video textures, in that adjacent audio segments that are classified into the same class are combined and represented by a single image tile in the time mosaic. In this case the calculated audio features for the sound are the average of the combined segments. For video texture generation for each frame in the output video the audio features are calculated from the corresponding piece of the audio file. The video frame and audio sequence are matched using time. In visualization by video textures, the audio properties change with time so that a sequence of feature vectors can be mapped to a sequence of frames.

Each of the frames in the output video texture is subjected to image processing filters which are driven by the characteristics of the corresponding audio piece. When users view the video texture component they can simultaneously view the audio content and the audio features changing over time.

Three perceptual audio features, power, pitch and signal-to-noise ratio, the same as those used in time mosaics chapter, are mapped to image processing filters for frames of the generated video texture to further convey aspects of the input audio file.

6.4 Blended Video Texture Mosaic Generation

The preceding section proposed methods for video texture component generation for an input audio clip. For those audio inputs which have only one audio clip, the resultant video texture components are the final results from the video texture generation module in the audio visualization system. But if the audio inputs are separated into more than one audio clip by the segmentation module, their video texture components are generated separately. This section discusses different ways to combine these video texture components so that the resultant output video for the input audio can be more efficient when users visually scan it. These methods utilize the basic ideas integral in time mosaic generation. Therefore we have coined the name video texture mosaic to represent the output of the video texture module. A video texture mosaic is the output of the video texture generation module in the audio visualization system when the audio input has more than one audio clip.

6.4.1 Parallel and Sequential Video Texture Mosaic Generation Methods

Once all audio clips have been detected and their respective video texture components have been identified, playing those video texture components in the same sequence as their corresponding audio clips results in an output that is of the same duration as the input audio file. This is illustrated in (A) and (B), Figure 6-19.

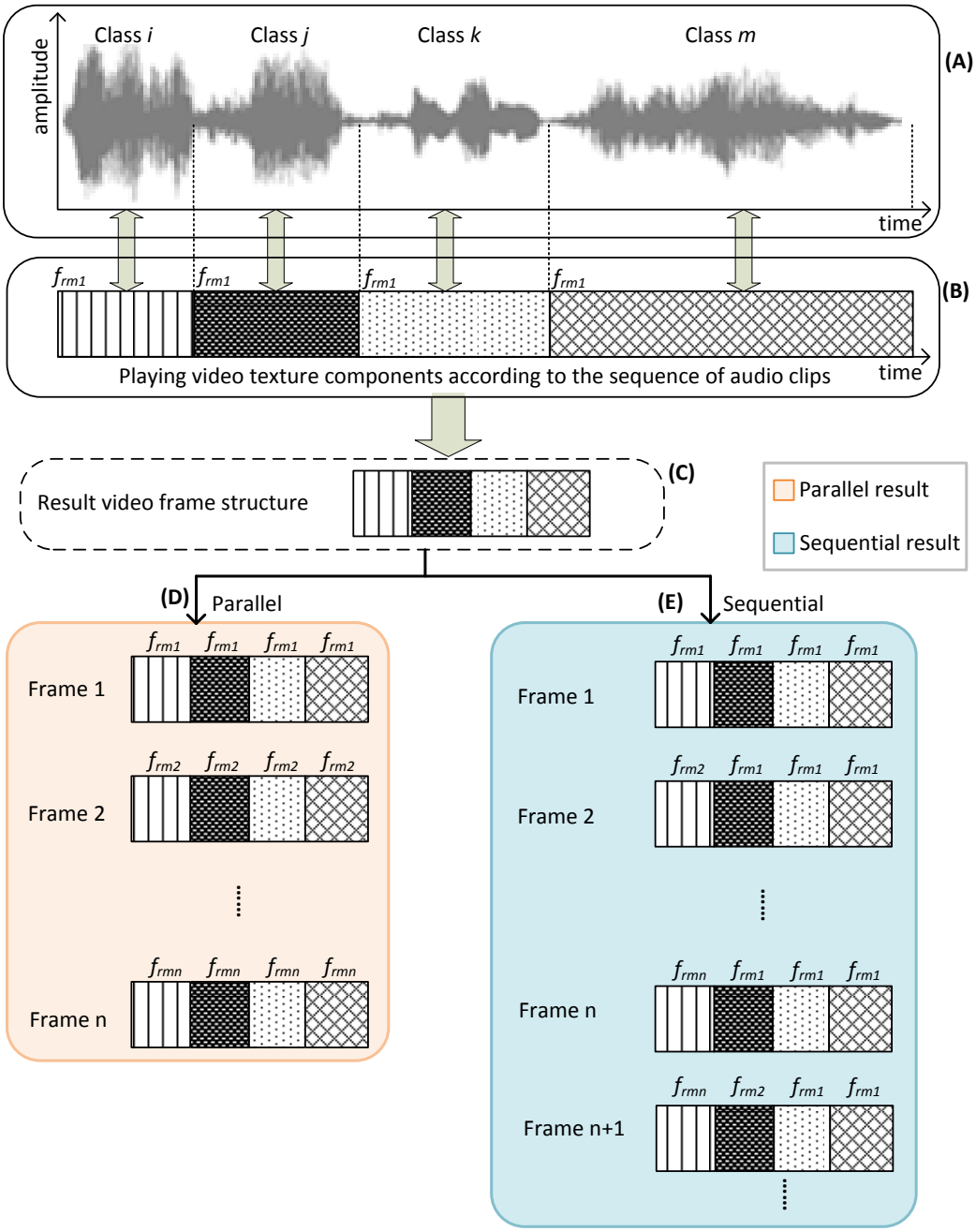

Figure 6-19: Combination of video texture components.

As all sounds in the audio input were visualized by separated video texture components, the final resultant video texture should allow viewers to view all these components at

one time. To achieve this, a frame in a video texture component for a sound is treated as an image tile in a dynamic Time Mosaic. When frames are selected from all the video texture components, each for one sound, they can be combined using Poisson image editing [185] to form a frame for the final resultant frame. The position of a frame of the original video texture component in the resultant video frame is determined by its corresponding audio clip's sequence. Figure(C) in Figure 6-19 illustrates the structure of a frame in a resultant video texture mosaic and an example is given in Figure 6-20. When any frame is viewed, users can visually scan the different contents in the audio input.

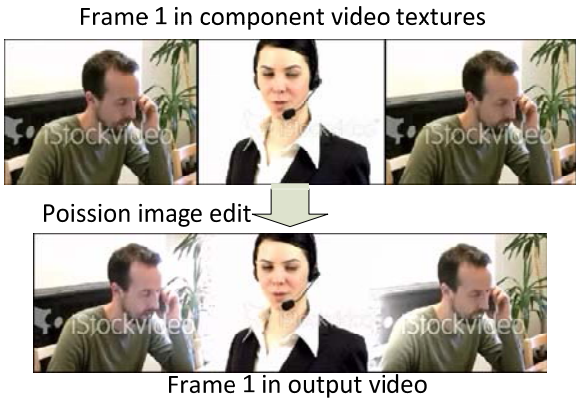

Figure 6-20: A frame generated from three individual frames using Poisson image editing.

The resultant video can be made either parallel or sequential (see Figure 6-19 (D) and (E)). For the parallel case video texture components run simultaneously, like playing all the components at the same time in different windows. The first frame in the resultant video is composed from the first frames of all the video texture components. Then the second frame is generated from all the second frames accordingly. When the frames of a single component run out, its last frame is used to generate the rest of the frames needed in the resultant video.

For an input audio clip, suppose the component video texture number is n, and the frames in these components are called component frames. Each frame in the final video is called a target frame, which should be generated by n component frames from the component video textures. For a parallel video result, the target frame i is generated using the i^{th} frame from all the component video textures. For those component video textures which have only j frames and $j < i$, the last frame is used to generate target

frames after frame *j*. The resulting parallel video plays all the components of the video texture at the same time independently.

In the case of sequential video texture mosaics, each target frame in the final output video is also generated by *n* component frames from the component video textures. The difference between this and parallel video is that the component video textures are played one after another according to their corresponding sound time position in the input audio. It is as though all the component video textures are opened at different windows but at any time only one component is playing. Only when the frames in the current video texture component run out, the frame of the next video texture component begins to change.

Although the target frames have all the component frames from the component video texture, only one of them changes from target frame *i* to *i+1*. So the duration of the output video is the same as that of the input audio file.

To summarize, if all the component video textures play at the same time, the result is a parallel blended video mosaics video. In the sequential blended video mosaics, although all the video texture components are shown in each frame of the final result, only one of them plays at each time and which one is playing is determined by the time sequences of their corresponding audio clips in the given audio input.

Figure 6-21 and Figure 6-22 are results for the same audio input. The former depicts the parallel blended video mosaics results. Figure 6-22 shows the sequential blended video mosaics results. The audio input has three different sounds: bird chirping, dog barking and kitten meowing. The sound clips are denoted by three grey progress bars with a red slider showing the position in the audio that the frame is representing. Figure 6-21 shows three frames of the parallel blended video mosaics result and Figure 6-22 shows three frames from the sequential blended video mosaics result. Both results are seamless blended video mosaics showing all component video textures at the same time but all video texture components play at the same time in the parallel video result and the sequential one plays only one component video texture at each time point.

With the parallel video result, viewers require a relatively shorter period (time for the longest component audio) to view all the component sounds. With the sequential video result, only one video texture component is running at a time so it takes longer to view

it but users can hear the audio and view the visual representations for all the sound clips in the audio input at the same time.

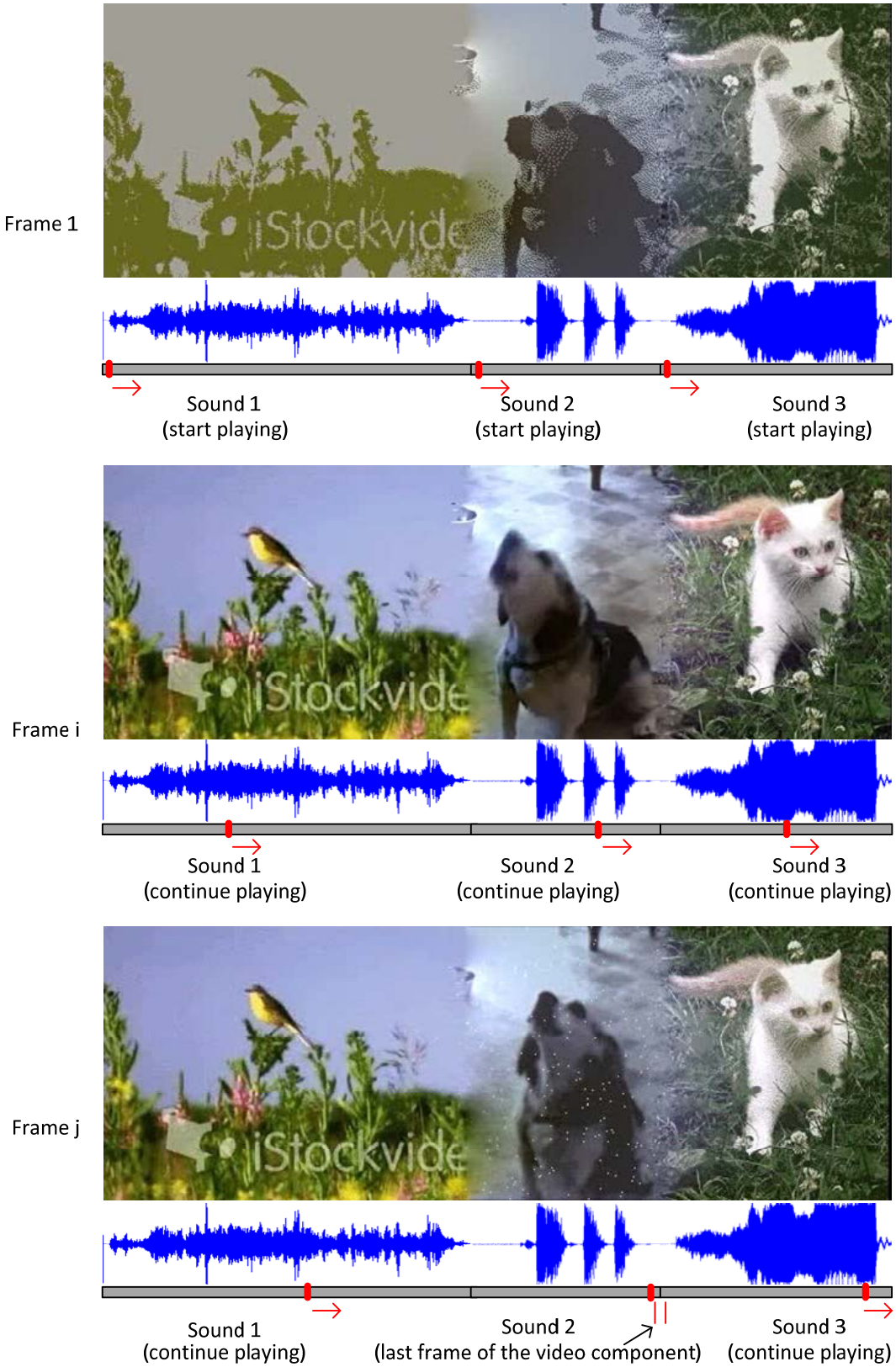

Figure 6-21: Parallel blended video mosaics results with audio features; Each component video texture plays at the same time.

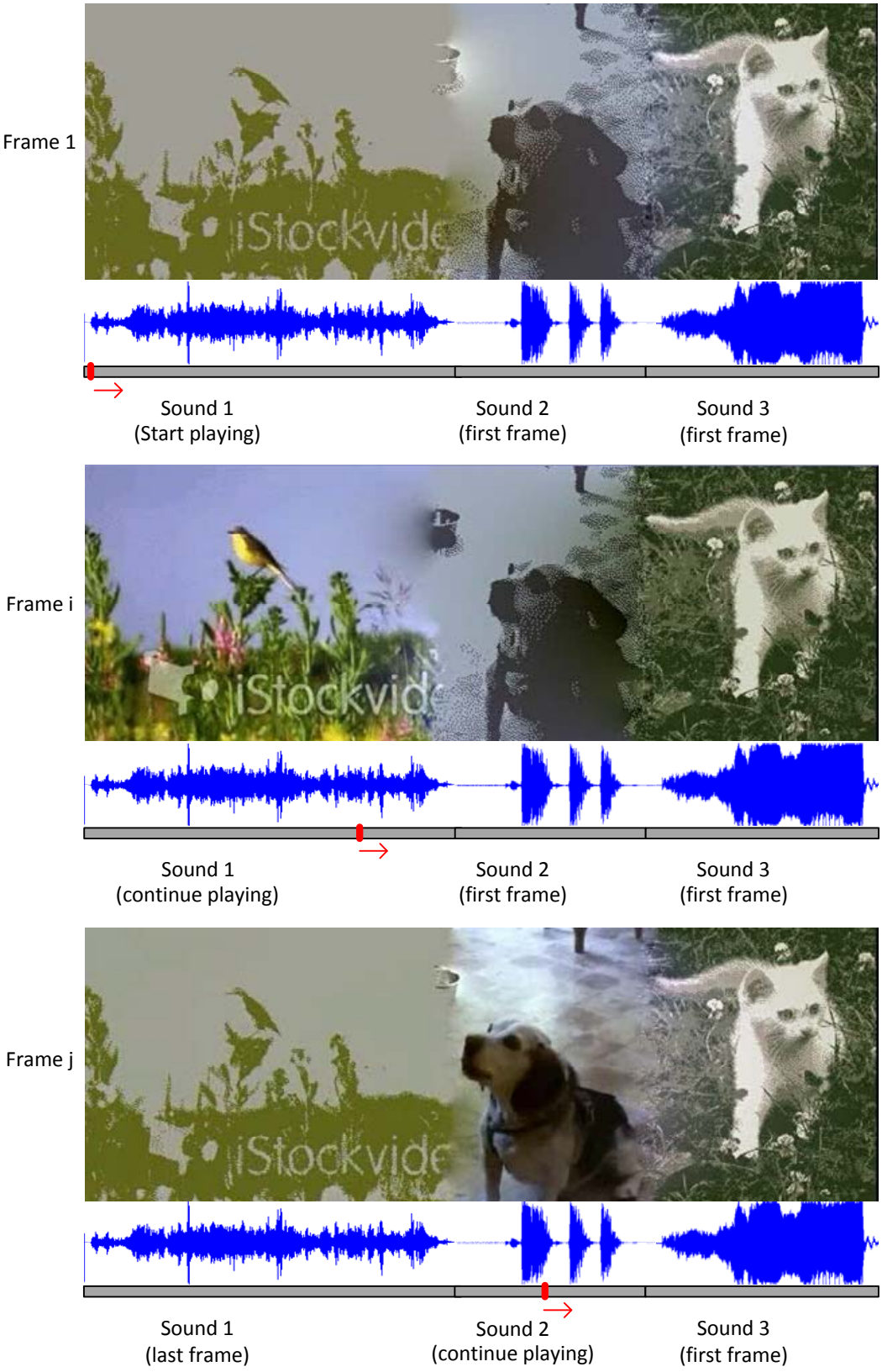

Figure 6-22: Sequential blended video mosaics results with audio features; Each Component video texture plays one after another.

6.4.2 Blended Video Texture Mosaic for Hierarchical Databases

This subsection presents how the blended video texture mosaic can be used for hierarchically structured databases. It improves the hierarchically structured database visualization and solves the problem of size limitation that comes from the computer screen. None of the video and video texture results presented previously in this chapter can convey audio files classified in a hierarchical ontology. One possible way of conveying hierarchical structures is tree maps. Time mosaics and video texture mosaics could be presented in a similar tree structure, however they are limited by screen real estate.

Video components provide the possibility of visually representing a hierarchically structured database. Some components representing audio clips belonging to the same parent class in an upper level can be grouped together and played in a sequence. For the visualization of a hierarchical database, video texture components are combined in a different way than previously discussed. Figure 6-23 shows how to combine the video texture components for a hierarchical database. The video texture components for the same class in a higher level are conjoined sequentially and treated as one. The resultant video is combined using the method previously described. The resultant video could be parallel or sequential.

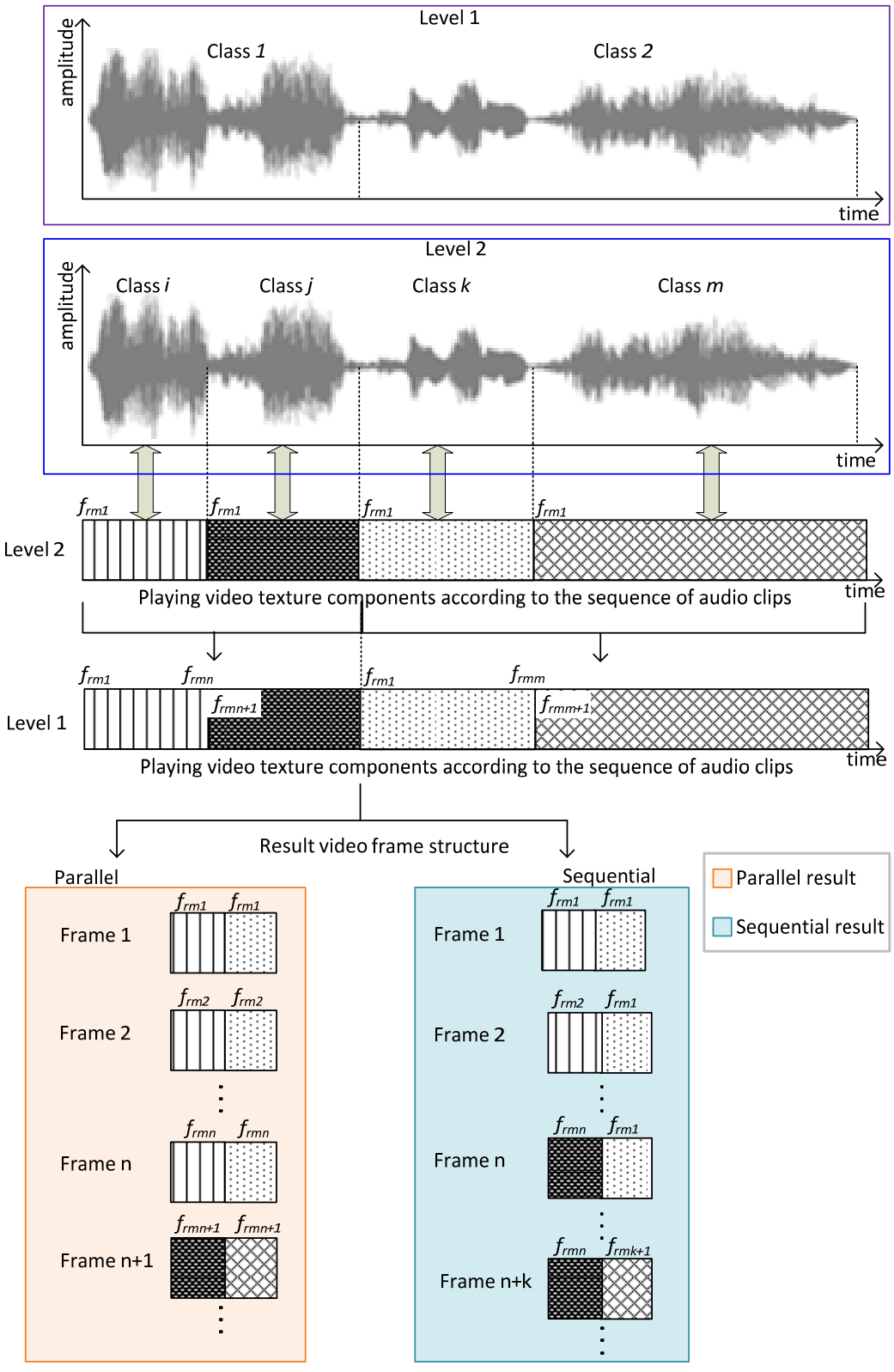

Figure 6-23: Combination of video texture components for a hierarchical database.

Figure 6-24 gives an example of the visualization of audio using video components. The upper-left image is a hierarchical schema of a hierarchically structured database and the lower-left image is a given input audio which contains speech and laughter. The left image is used to represent laughter. The left images in the video texture frames may belong to different classes in the lower level, but all of them belong to the same laughter class in the upper level. Accordingly the right images in the upper level are from the speech class.

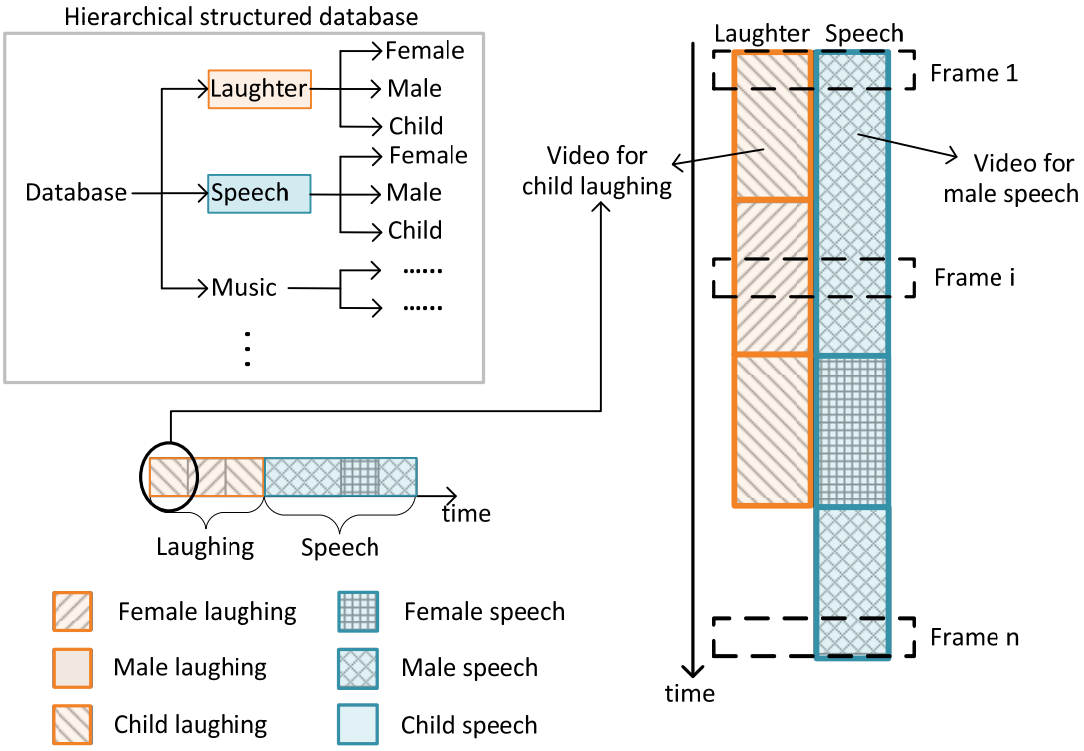

Figure 6-24: Visualization using a hierarchically structured database and video components.

Figure 6-25 shows a parallel blended video mosaics result for an audio input that contains six different audio clips belonging to classes "Laughter→Child", "Laughter→Female", "Laughter→Child", "Speech→Male", "Speech→Female" and "Speech→Male". The first three belong to the same parent class “Laughter” and the last three belong to the parent class "Speech". So the video texture components from the same parent class are connected one after another to generate a new video texture component for a class in the upper level of the hierarchically structured database. Then the resultant video texture mosaic is generated applying the method discussed in the preceding subsection (Subsection 6.4.1).

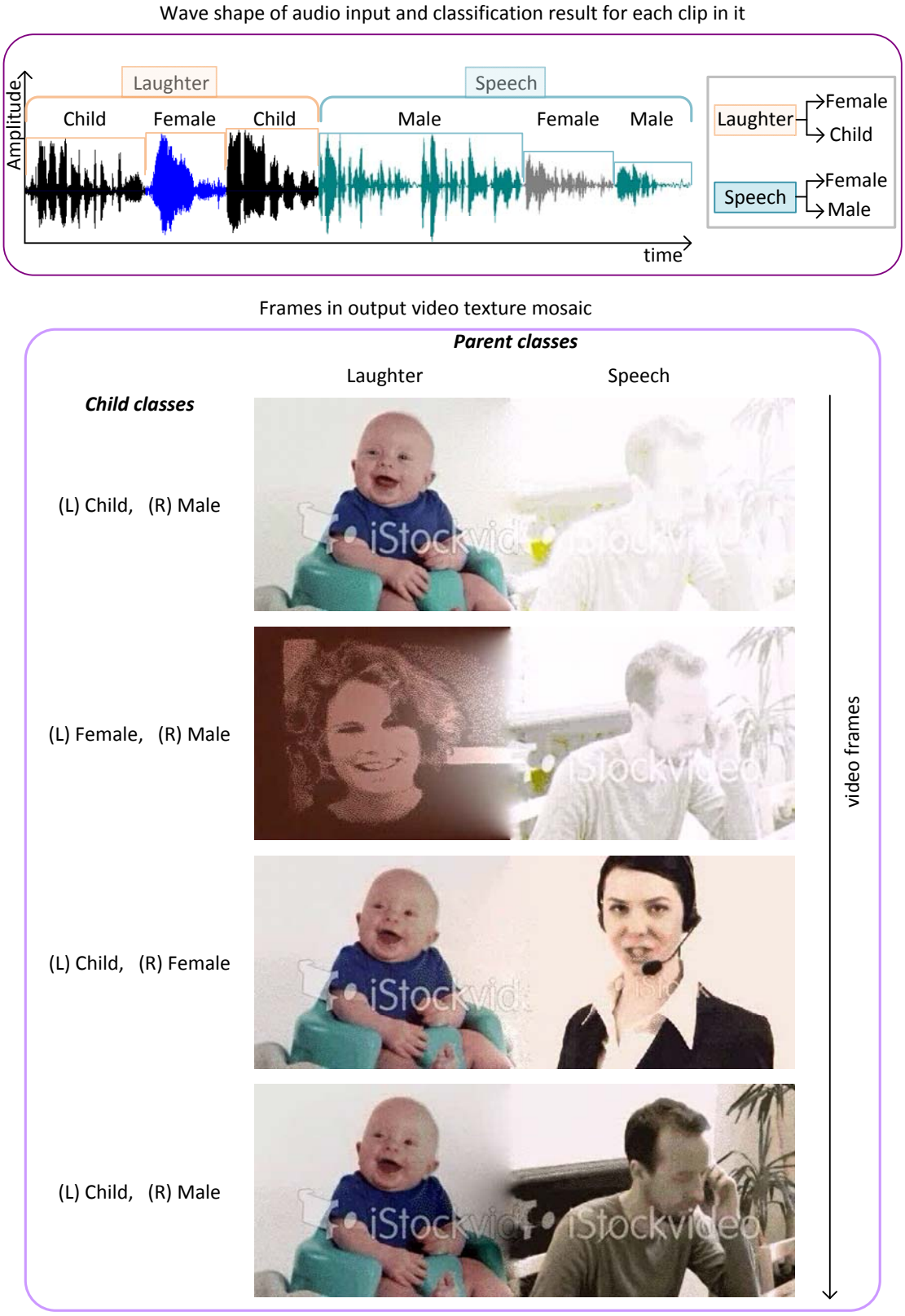

Figure 6-25: A parallel blended video texture mosaic result for sounds from two parent classes. The component video textures play independently but simultaneously.

When video texture mosaics are generated for audio inputs which have multiple sounds belonging to different classes, the image size of each sound in the resultant blended

frame will be limited due to the size of the computer monitor. We can solve the limitation of screen size utilizing classes at a higher level of a hierarchically structured database. Super class video textures are used to represent adjacent sounds that classify to a descendant class in a video texture mosaic. By selecting mosaic tiles the user may mine deeper into the hierarchy to find a more accurate representation of the areas of interest.

6.5 Summary

This chapter presented the methods used in the video texture generation module of the audio visualization system. The inputs of this module are the audio clips and their corresponding classification results; just the same as for time mosaics generation module. To generate accurate and meaningful video textures, appropriate template videos are required for the classes in the audio database. The result from this video texture generation module is a video texture for audio input containing a single sound or blended video texture mosaics for input audio files containing heterogeneous sounds belonging to different classes. The result can be either of the same duration or shorter than the audio input. Each frame in the resultant blended video texture mosaics contains image elements for all the sound clips in the audio input, allowing viewers to scan the content of a heterogeneous input audio file. At the same time, the acoustic characteristics are illustrated over time by using the selected audio properties to drive the image processing filters on the frames.

The novel adaptive video texture generation method was introduced in this chapter to further extend the general video textures by using acoustically similar detection to produce a resultant video texture that is more accurate in visualizing the audio input.

Video textures inherit the advantages of video such as capturing motions, and at the same time, acquire the merits of time mosaics. They improve on the time mosaics approach by presenting the audio characters over time. If an image tile in a time mosaic represents more than one audio segment, its visual features are driven by the mean values of these audio segments. However a frame of a video texture visualizes only audio features of the audio segment it represents. A video texture allows viewers to identify a segment with certain properties from an audio file (such as a relatively noisy segment from an audio file). In the proposed audio visualization system, for those audio

files which are not suitable to be visualized by a single image, the video texture method generates interesting video that illustrates their audio properties.

Moreover, the blended video texture mosaic is suitable for the visualization of mixed sounds from a hierarchically structured audio database. It enables viewers to see the relationship between superior and subordinate classes in a hierarchical structure. Video texture mosaics solve the problem of representing audio files that contain many different sounds on a screen of limited size by representing sounds by frames along a time continuum instead of by a static image. The relationship of sounds in a hierarchically structured database can be represented clearly because the video texture components can be grouped appropriately to generate the video texture mosaics,

The video texture generation module and the time mosaics module form the visualization capability in the audio visualization system presented in this thesis. The resultant video texture of an audio file can be played when viewers select all the image tiles in its time mosaic result. If users choose one image tile in the resultant image mosaics, the generated video texture for its corresponding audio segment or segments will be played. Time Mosaics and Video Texture Mosaics are complementary and alternative visualization methods, and make the visualization system more flexible.

Chapter 7
Summary and Future Work

This final chapter firstly summarizes the research undertaken for each module of the audio visualization system presented in the thesis. Following this the limitations of the system and possible solutions for overcoming these limitations, are discussed. Finally, some possible directions for future research that may contribute to audio visualization and related areas are outlined.

7.1 Summary

The goal of this research was to design a system that could visualize the content and audio properties of audio files containing general sounds. None of the existing audio visualization systems meet this requirement. Existing audio visualization systems are limited to particular types of sounds, such as music or speech.

An audio visualization system requires three essential components: audio segmentation, audio classification and the representation of audio features by visual features. The first two components are discussed in Chapter 3 and Chapter 4. The last topic is separated into two modules, meaningful image generation discussed in Chapter 5, and video texture generation discussed in Chapter 6.

How our novel system meets the requirements is reviewed here by revisiting and addressing the research questions below:

1. *Can we design a system to visualize heterogeneous audio inputs?*
    a. *Can we separate the different sounds and represent the contents accurately?*
    b. *How can we combine images or video to represent the content of a digital audio file?*

2. *Is it possible to develop an audio classification method that classifies general sounds accurately enough to allow for visualization of an audio file?*
3. *Is it possible to classify an unknown sound by integrating an adaptive new-class detection method into an audio classification method?*
4. *How can we construct an image or image sequence that represents a digital audio file?*
    c. *How can features of audio files be mapped to the features of images?*
    d. *How can we combine images to form a sequence that is representative of a digital audio file?*
5. *How can we synthesize new video sequences that map accurately to the content and sequence of a digital audio file?*
6. *Can meaningful images and video textures be automatically generated for digital audio files?*

Before sounds in an audio database can be visualized they must first be accurately classified. A lack of accuracy of classification would result in inconsistent visualization of sounds and a consequent lack of utility. Audio segmentation and classification are the foundations of any system for visualizing audio input.

The first three research questions have been successfully addressed through the development of a novel accurate audio segmentation method and an adaptive audio classification with new class detection.

The system must first be able to separate different sounds within an audio file according to its audio properties. Our segmentation module contains a novel two-phase method for the segmentation of general audio input. In the segmentation experiments for audio files from two different audio databases reported in Chapter 3, it is shown that the two-phase audio segmentation method provides significantly better accuracy than any of the previous methods.

Once the audio input is separated into homogeneous sounds, their contents need to be classified automatically. This is the main task of the classification module. Our classification process is based on the NFL classification method as this had been reported in the literature as being the most accurate. To improve classification performance, a novel method was developed to detect any sounds falling outside the

existing classes in the training set. This provided the classification module with the additional function of new-class-detection so that it could not only classify sounds belonging to existing classes but could also manage those new sounds that were out of the scope of existing classes. This method avoids potential classification errors and makes the audio visualization system more flexible and practical.

The effectiveness of the audio segmentation and classification modules functioning in combination is tested and reported in Chapter 4. These two processes when used in combination with uncertainty detection in the classification module, significantly improve accuracy by reminding users about potential classification mistakes and result in a system of audio processing that has the level of accuracy necessary for reliable audio visualization. By combining segmentation and classification we have a system that processes the audio file so that we may visualize mixed-audio files and previously unknown audio or sounds that either fit into the audio file ontology or are a new type of sound.

The visualization of audio input was achieved in two different ways in this research: by static images and by video textures.

Question four relates to the ways in which image features can be mapped to audio features and how a sequence of images can be generated in seamless manner to represent and audio file. These issues were explored in the development of a new visualization technique called time mosaics.

The first method results in an audio input being visualized by a time mosaic image that contains image tiles corresponding to the different sounds in the audio file. The selected audio properties of the audio input are used to drive image processing that controls visual features so that the resultant image represents certain selected audio properties.

The second method for sound visualization results in video textures and is particularly suited for illustrating lengthy audio files and audio files from a hierarchically structured audio database. It is also more accurate in its ability to represent time-based audio properties. The novel adaptive video texture generation method reported in this thesis provides a solution to research question 5.

Our audio visualization system is designed for general sounds. The types of sounds are determined with respect to the audio database which is used as the training set for the system. After the training process, the system is able to work independently for any given audio files. The system can create novel results for various kinds of audio inputs: short or lengthy input; single sound or multiple sounds input; sounds belonging to existing classes of the training set or to new classes.

The image mosaic generation module contains processes for visualizing given audio files by static images. With the seamless merging of image tiles for each type of sound in the audio input, the contents of an audio input are visualized and the mean audio properties of each audio segment are illustrated by visual features.

The video texture module is used to show the time-based audio features more clearly. With the adaptive video texture generation method, the generated output video texture can best match the input audio contents.

The research reported in this thesis contributes a system for visualizing general audio files and is a significant advance over previously reported audio visualization systems. Furthermore, the component audio segmentation and classification methods used at various stages of processing could be adapted to other similar applications. We believe that this thesis presents the most comprehensive and generalizable audio segmentation, classification and visualization system developed to date.

7.2 Limitations

There remain some limitations of our system, which may lead to spurious results under certain conditions. An inaccurate visualization can occur when one of the sounds within an audio input is too low in volume for the system to identify. Figure 7-1 illustrates this limitation. The sound file contains a cat-bird-dog sequence of sounds. The bird sound amplitude in this file is too low for our segmentation method to detect and results in a time mosaic with only two component images.

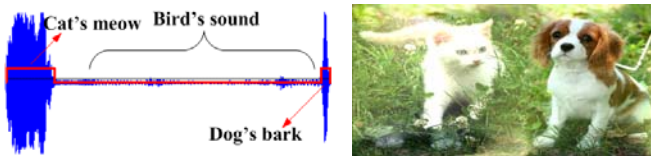

Figure 7-1: Three sounds misrepresented as two images tiles.

Another possible error can occur in situations where sounds overlap. Figure 7-2 shows an audio signal that consists of a cat's meow and a bird sound in which there is a period of overlap between them. The segmentation failed to separate the 2 sounds, resulting in the sound file being treated as a single sound and classified as a rooster for which a single image was produced. This form of error cannot be entirely avoided as it depends on the nature and quality of audio inputs.

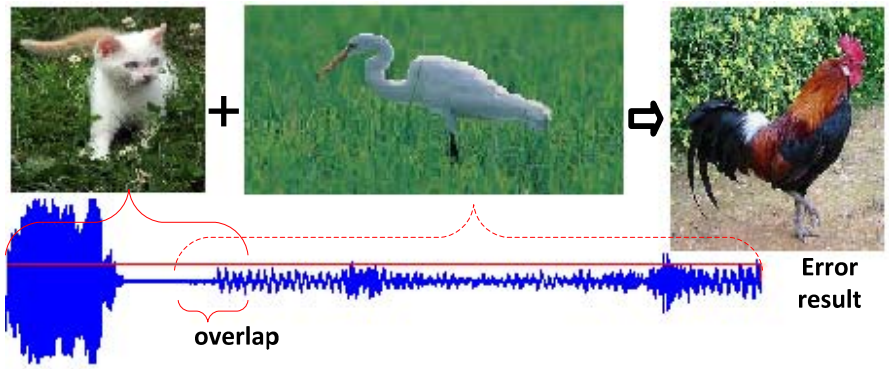

Figure 7-2: Overlapping audio signals producing an incorrect result.

The audio visualization system reported here partly relies on a mapping of sound to image features and the human perception of the changes in the combined image features. Although there have been a large number of studies about the relationships of audio features and visual features, there is relatively little work about the perception of them. What we do know is that perceptions of both the audio features and visual features are affected by factors such as the cultural backgrounds of the people forming the associations as well as physiological considerations. Because of the subjective nature of the perception of audio and visual features and of video texture results, evaluation of the quality of resultant time mosaic images or video textures may be culturally dependent. We aimed to mitigate this problem by designing a system that gives users the ability to state their own preferred audio-visual relationships by overriding the mapping relationships between audio features and visual features.

Moreover, with a legend to illustrate the mapping relationships the user is supplied with a point of reference to aide perception of differences in image features. The extent to which these features overcome the issue of individual differences in perceptual associations would have to be tested in usability studies and is left to future work.

Another limitation of our current audio visualization system lies in possible conflicts between visual features. For example, even with the legend, users may have difficulties in distinguishing the visual noise and the colour fading in a resultant time mosaic image. It is also not known how much a visual feature must change before a viewer is able to perceive a difference. For example, when we add visual noise to two images to represent the NSR of two audio files, can the viewers perceive the differences between the two images? Though, it should be noted that as the differences become smaller, their importance also diminishes.

7.3 Future work

In this section some ongoing work is discussed. Some of this work is directed towards improved audio visualization and some is more relevant to the improvement of the performance of a single module.

7.3.1 Audio Segmentation and Classification

We explored many audio features in our audio segmentation and classification modules and they performed well, but a comprehensive investigation of all possible feature choices and feature combinations has not been undertaken. In the implementation of the audio segmentation module and the classification module, the importance of each feature is equally weighted. It is unlikely this would be the case for all audio databases or audio classes. Therefore, an adaptive weighting of the distance vectors of the audio feature set would be worth exploring as a way of optimizing the accuracy of the segmentation and classification methods.

Similarly, we have used the best reported parameter sets in the literature and it would be worth exploring other possible parameters for our system. Other parameters have not been evaluated.

There are a number of ways that the audio classification method might be improved and the following experiments might be worthy avenues of investigation. The threshold for a parameter was detected linearly in our classification method (Section 4.4.4). If there are two parameters, the parameter space is divided into squares. For three parameters space, it is separated into cuboids etc. However, there are other possibilities for dividing the parameter space that might be worth exploring.

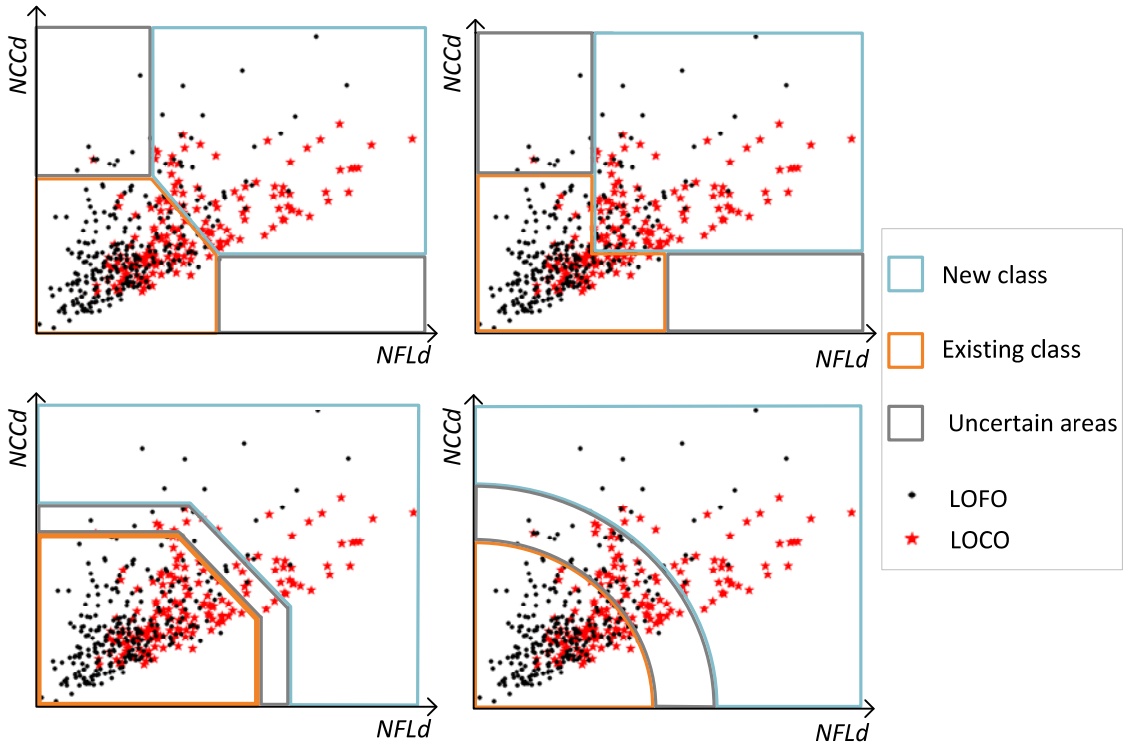

Figure 7-3: Fold-line division for the *uncertain* criterion

For example, *NFLd* and *NCCd* may create the parameter space shown in Figure 7-3 (top left). In this case the space is separated into four using a fold-line. The other three images in Figure 7-3 show three other possibilities for separating parameter space using fold-lines. The separation of parameter space depends on the training process. Each audio file results in two points in the high-dimensional parameter space after LOFO and LOCO experiments. For all the audio files, the points from LOFO generate a cluster and those from LOCO generate another cluster in the parameter space. The separation of two clusters depends on their location in the parameter space and their distribution. In future work it would be worth attempting to separate the clusters with alternate methods.

One more possible improvement for the new-class-detection algorithm lies in the audio feature space selection. In this module, the best feature set for general classification is adapted to new class detection directly. This is not necessarily optimal for detecting new classes but it is sufficient for the image to audio matching required for our system. In future work, to achieve the most accurate results, several audio feature spaces should be compared for the new-class-detection training process. Additionally, changing the structure of the training set to a deeper hierarchical structure is worth further exploration.

7.3.2 Audio Visualization

There are formats that could be explored for the visualization of audio features other than those reported in Chapter 5, such as showing contours of the audio clips (Figure 7-4 (L)). Wave shapes, or contours of audio clips, play an important role when the sound is lengthy because other audio features are mean values of the whole sound. When two sounds have similar audio features, but different durations, their wave shapes can help to show the differences. The wave shapes of audio clips are sometimes more sensitive to changes in sound. For example, the audio in Figure 7-4 (L) has three component sounds each of different amplitude. The differences between the volumes of the three audio clips may be too small to be visualized by visual features such as colour depth. Their durations are the same so the heights of the image tiles are the same. If the image tiles were resized according to the corresponding amplitudes of the sounds this would either make their widths different (which represents the duration) or distort the shape of the image tiles. If the edges were cut directly from the original size of the image tiles, important information on the tiles may be lost.

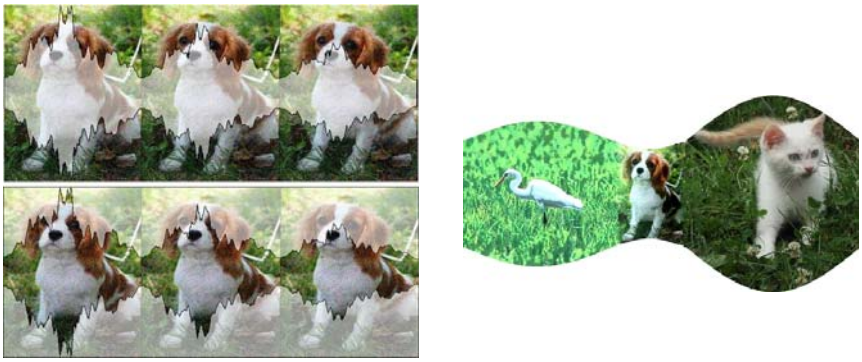

Figure 7-4: Audio visualization: (L) with wave shape illustration; (R) using curve shapes.

Figure 7-4 (R) illustrates a possible solution for merging image tiles, with different heights, without using a background image. A curve is used to make the matching edges of the adjacent image tiles the same height.

Figure 7-5: 3D audio visualization: (L) cylinder mapping; (R) Z-values mapped to feature.

Figure 7-5 shows two alternative 3D visualizations for the input audio illustrated in Figure 5-12 in Chapter 5. Using 3D is one possible approach to visualizing audio files that have a large number of same-sound pieces in a clip where the sounds belong to the same class but their audio features are different. From left to right, the cylinders show the audio sequences. Then from the top to the bottom in each cylinder, the sound pieces are shown sequentially. Z-values of the cylinders could also be used to visualize other audio features Figure 7-5 (R) and the radius of the small cylinders could be used as a visual mapping to audio features such as power or amplitude.

In the audio visualizations generated for this research we used realistic photo images for the class templates. We were interested in being able to utilize the richness of information within a realistic image to explore potential audio visualization approaches and to present a view of the sound. However it may be that approaches such as non-photorealistic rendering of the template images or the use of iconic images as templates may also be effective in conveying the class of the sound. Additionally the simplicity of the image may make user perception of the differences in image features when filters are applied easier than when a photo is used as a template image. Usability studies of these alternative approaches would be a useful direction for further work on the time mosaic visualization concept.

Another approach could be one where the audio clips are represented by image elements instead of by whole image tiles. The image elements for audio clips could be placed in one background image. The resultant images might appear like those shown in Figure 7-6. They contain multiple overlapping image icons on a white background. In this way, the audio clip numbers are visualized and their duration can be represented by the size of the elements. Different elements indicate different classes of audio files.

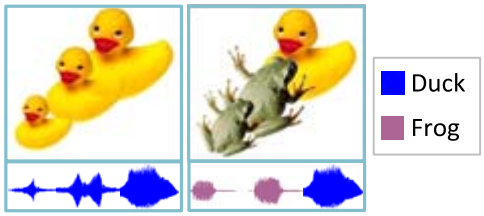

Figure 7-6: Using image elements to represent audio clips.

Figure 7-7 shows another alternate representation that uses small icon elements for small audio pieces in time. In the resultant image, opacity can be used to represent the relative volume of each sound at any point in time so the resultant image is suitable for overlapped sounds.

Figure 7-7: Element mosaic time-lines result.

Figure 7-6 and Figure 7-7 give examples of using non-realistic, iconic images instead of realistic images. If non-realistic images were found to be very suitable for certain classes or purposes, then computer animation techniques might also be worth exploring in order to enhance audio visualization.

The video texture generation method developed in this thesis could also be utilized in other application spaces. For example, in speech visualization the template could be extended to include specific vocabularies (Figure 7-8). This work then goes beyond

general audio visualization and closer to the notion of employing video rewrite for speech visualization.

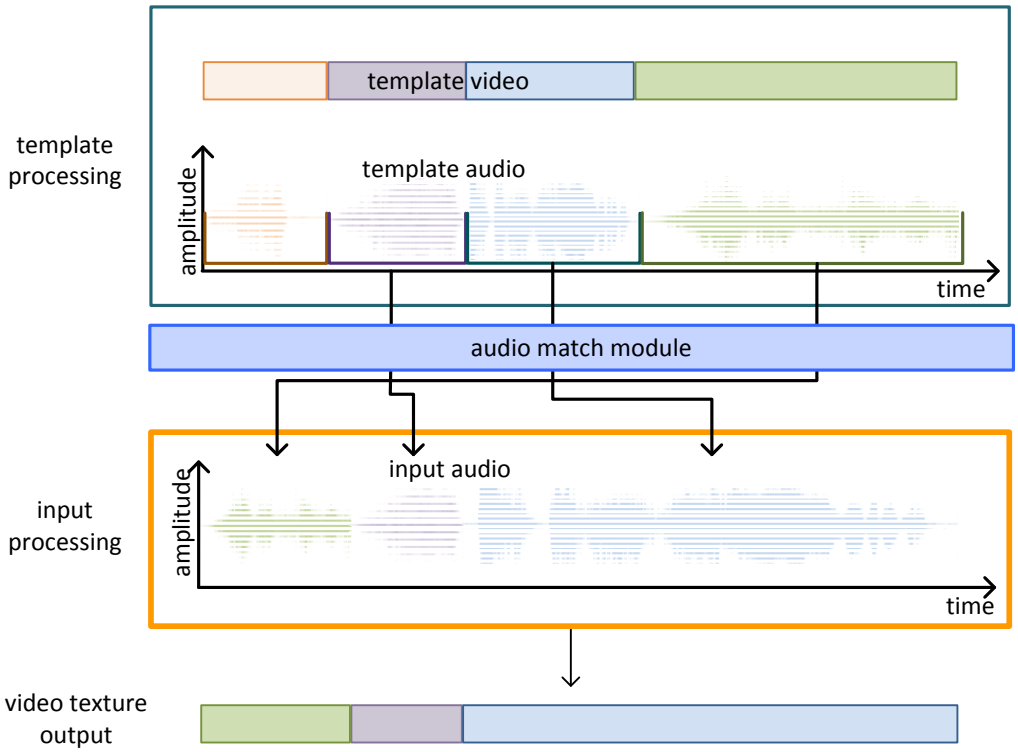

Figure 7-8: Schematic diagram of the speech visualization.

Other applications for the visualization of audio input by video textures could be investigated, such as audio file editing. The editing could be undertaken with simple image manipulation by the user, such as dragging and dropping, cutting and pasting images, to rearrange sound sequences remove sounds and add sounds to an audio file. Scaling an image icon, or image object, could result in an increase in volume of its corresponding sound relative to the other sounds in the file. Such an editing tool would remove the need for manual point detection and splicing of audio segments using audio signal displays.

7.4 User Studies

User studies will play an essential role in the future development of this audio visualization system. Controlled user studies with observational software to capture user interaction could assist in refining the system and designing new audio visualizations.

A number of complex issues regarding user perception will have to be resolved for the best audio-visual feature mappings and visual representations for audio to be designed. These issues raise research questions such as:

- Which types of representation could best be used to represent abstract sounds?
- Which types of representation could best be used to represent concrete sounds?
- What is the limit of human perception for feature mapping? In other words what is the maximum number of features that can be meaningfully mapped?
- Does the use of template image legends assist in user comprehension of the audio visualizations?
- How does mapping the dimensionality of features affect human perception of the visualization of audio?
- What is the range of perception of the visual features in an image?
- Which combinations of visual features aide user comprehension and which combinations conflict and confound the user's comprehension?
- What is the affect of conflicting visual features on the user's ability to interpret a sound visualization?
- Can the user gain information from a mapping of conflicting audio features to conflicting visual features?
- How complicated can the visualizations get before they can no longer be rapidly scanned?

In addition there are the issues around user tasks. For a non expert user of an audio database there is the problem of the time required in order to learn how to interpret a visual representation of an audio. Most non-professional users would expect an intuitive, easy to learn system and do not wish to invest significant amounts of time learning how to interpret and use the system. User studies could be designed to evaluate the speed and accuracy of users in browsing and searching for a specific sound within the database. For professional users the visualization system may speed up audio query and navigation. Comparative user studies with the performance of professional systems could be undertaken to discover whether professionals are assisted by such audio visualizations.

References[2]

1. E. Wold, T. Blum, D. Keislar and J. Wheaton, "Content-Based Classification, Search, and Retrieval of Audio," in *IEEE Multimedia*, Rostock, Germany, 1996, vol.3, pp. 27-36.
2. Service-1.org, Critical Seeker - photo hunt game, 2006. [online]. Available: http://www.downloadatoz.com/criticalseeker/. [Accessed: 31 May, 2009].
3. D. Gerhard, "Audio Visualization in Phase Space," in *In Bridges: Mathematical Connections in Art, Music and Science*, Winfield, KS, USA, 1999, pp. 137-144.
4. K. Giannakis and M. Smith, "Imaging Soundscapes: Identifying Cognitive Associations between Auditory and Visual Dimensions," in *Musical Imagery*, Lisse, Netherlands, 2001, pp. 161-179.
5. S. Nomura, T. Shiose, H. Kawakami, O. Katai and K. Yamanaka, "A Novel "Sound Visualization" Process in Virtual 3D Space: the Human Auditory Perception Analysis by Ecological Psychology Approach," in *Proc. 8th Asia Pacific Symp. Intelligent and Evolutionary Systems*, Cairns, Australia, 2004, pp. 137-149.
6. R. V. Jones, *Sound Visualization and Analysis in the Pre-Electronic Era*, 1999. [online]. Available: http://people.seas.harvard.edu/~jones/cscie129/images/snd_vis/snd_vis.html. [Accessed: 20 May, 2009].
7. R. Hagiwara, *How to Read a Spectrogram*, 2009. [online]. Available: http://home.cc.umanitoba.ca/~robh/howto.html. [Accessed: 18 May, 2009].
8. G. Tzanetakis and P. Cook, "Audio Information Retrieval (AIR) Tools," in *Proc. ISMIR'00*, Plymouth, MA, USA, 2000, pp. 10.
9. D. Politis, D. Margounakis and M. Karatsoris, "Image to Sound Transforms and Sound to Image Visualizations based on the Chromaticism of Music," in *7th WSEAS Int. Conf. Artificial Intelligence, Knowledge Eng. and Databases*, Cambridge, UK, 2008, pp. 309-317.
10. S. Ferguson, A. V. Moere and D. Cabrera, "Seeing Sound: Real-time Sound Visualisation in Visual Feedback Loops used for Training Musicians," in *Proc. 9th Int. Conf. Information Visualisation*, London, UK, 2005, pp. 97-102.
11. H. G. Kaper, E. Wiebel and S. Tipei, " Data Sonification and Sound Visualization," *Computing in Science and Engineering*, vol. 1, no. 4, pp. 48-58, July 1999.
12. F. Sobieczky, "Visualization of Roughness in Musical Consonance," in *Proc. IEEE Visualization*, San Francisco, CA, USA, 1996, pp. 355-357.
13. K. Giannakis and M. Smith, "Towards a Theoretical Framework for Sound Synthesis Based on Auditory-Visual Associations," in *Proc. AISB'00*, Birmingham, UK, 2000, pp. 87-92.
14. B. Evans, "Musical Connections and Heterophonic Maps," presented at the ACM SIGGRAPH'06 Sketches, Boston, MA, USA, 2006.

[2] Referencing is in the IEEE numeric format.

15. S. M. Smith and G. N. Williams, "A Visualization of Music," in *Proc. 8th Conf. Visualization*, Phoenix, AZ, USA, 1997, pp. 499-503.

16. N. Kaya and H. H. Epps, "Relationship between Color and Emotion: A Study of College Students," *College Student Journal*, vol. 38, no. 3, pp. 396-405, Sept. 2004.

17. J. Foote, "Visualizing Music and Audio Using Self-similarity," in *Proc. 7th ACM Int. Conf. Multimedia (Part 1)*, Orlando, FL, USA, 1999, pp. 77–80.

18. W. Chai and B. Vercoe, "Structural Analysis of Musical Signals for Indexing and Thumbnailing," in *Proc. JCDL'03*, Houston, TX, USA, 2003, pp. 27-34.

19. M. Cooper and J. Foote, "Automatic Music Summarization via Similarity Analysis," in *Proc. ISMIR'02*, Paris, France, 2002, pp. 81-85.

20. T. Bergstrom, K. Karahalios and J. C. Hart, "Isochords: Visualizing Structure in Music," in *Proc. ACM Int. Conf. Graphics Interface*, Montreal, Canada, 2007, vol.234, pp. 297-304.

21. *Music visualization*, 2009. [online]. Available: http://en.wikipedia.org/wiki/Music_visualization. [Accessed: 20 May, 2009].

22. D. Politis, D. Margounakis and K. Mokos, "Visualizing the Chromatic Index of Music," in *Proc. WEDELMUSIC'04*, Barcelona, Spain, 2004, pp. 102-109.

23. R. Hiraga and N. Matsuda, "Visualization of Music Performance as an Aid to Listener's Comprehension," in *Proc. AVI'04*, Gallipoli, Italy, 2004, pp. 103-106.

24. K. Lubar, "Color Intervals: Applying Concepts of Musical Consonance and Dissonance to Color," *Leonardo*, vol. 37, no. 2, pp. 127-132, May 2004.

25. S. Malinowski, *Music Animation Machine*, 2007. [online]. Available: http://www.musanim.com/player/. [Accessed: 14 May, 2009].

26. D. V. Oppenheim, "Compositional Tools for Adding Expression to Music," in *Proc. ICMC'92*, San Jose, CA, USA, 1992, pp. 223-226.

27. F. Watanabe, R. Hiraga and I. Fujishiro, "Brass: Visualizing Scores for Assiting Music Learning," in *Proc. ICMC'03*, Singapore, 2003, pp. 107-114.

28. S. Dixon, W. Goebl and G. Widmer, "The Performance Worm: Real Time Visualization of Expression Based on Langner's Tempo-Laudness Animation," in *Proc. ICMC'02*, Goteborg, Sweden, 2002, pp. 361-364.

29. R. Hiraga, S. Igarashi and Y. Matsuura, "Visualized Music Expression in an Object-oriented Environment," in *Proc. ICMC'96*, Hong Kong, China, 1996, pp. 483-486.

30. R. Hiraga, F. Watanabe and I. Fujishiro, "Music Learning Through Visualization," in *Proc. WEDELMUSIC'02*, Darmstadt, Germany, 2002, pp. 101-108.

31. R. Hiraga, R. Mizaki and I. Fujishiro, "Performance Visualization: A New Challenge to Music Through Visualization," in *Proc. 10th ACM Int. Conf. Multimedia*, Juan-les-Pins, France, 2002, pp. 239-242.

32. J. Hailpern, K. G. Karahalios, J. Halle, L. DeThorne and M.-K. Coletto, "Visualizations: Speech, Language & Autistic Spectrum Disorder," in *Proc. CHI'08*, Florence, Italy, 2008, pp. 3591-3596.

33. K. Karahalios and T. Bergstrom, "Visualizing Audio in Group Table Conversation," in *Proc. 1st IEEE Int. Workshop Horizontal Interactive Human-Computer Systems*, Adelaide, Australia, 2006, pp. 131-134.

34. J. Donath, K. Karahalios and F. Viegas, "Visualizing Conversation," in *Proc. HICSS-32*, Maui, HI, USA, 1999, pp. 9-17.

35. T. Bergstrom and K. Karahalios, "Seeing More: Visualizing Audio Cues," in *Proc. INTERACT'07*, Rio de Janeiro, Brazil, 2007, vol.4663, pp. 29-42.

36. M. Simunek, "Visualization of Talking Human Head," presented at the CESCG'01, Budmerice, Slovakia, 2001.

37. C. Bregler, M. Covell and M. Slaney, "Video Rewrite: Driving Visual Speech with Audio," in *Proc. ACM SIGGRAPH*, Los Angeles, CA, USA, 1997, pp. 353-360.

38. A. Chaudhary and A. Freed, "Visualization, Editing and Spatialization of Sound Representations using the OSE Framework," presented at the AES 107th Convention, New York, NY, USA, 1999.

39. T. Kunze and H. Taube, "See: A Structured Event Editor - Visualizing Compositional Data in Common Music," in *Proc. ICMC'96*, Hong Kong, China, 1996, pp. 63-66.

40. S. Imai, S. Kurabayashi, A. Ijichi and Y. Kiyoki, "A Music Retrieval System Supporting Intuitive Visualization by the Color Sense of Tonality," in *Proc. IASTED DBA'06*, Innsbruck, Austria, 2006, pp. 153-159.

41. J. Zhu and L. Lu, "Perceptual Visualization of A Music Collection," in *Proc. ICME'05*, Amsterdam, Netherlands, 2005, pp. 1058-1061.

42. E. Pampalk, "Islands of Music: Analysis, Organization, and Visualization of Music Archives," Master' thesis, Dept. Software Technology and Interactive Systems, Vienna University of Technology, Vienna, Austria, 2001.

43. E. Pampalk, A. Rauber and D. Merkl, "Content-based Organization and Visualization of Music Archives," in *Proc. 10th ACM Int. Conf. Multimedia*, Juan-les-Pins, France, 2002, pp. 570-579.

44. E. Pampalk, W. Goebl and G. Widmer, "Visualizing Changes in the Structure of Data for Exploratory Feature Selection," in *Proc. SIGKDD'03*, Washington, D.C. USA, 2003, pp. 157-166.

45. M. Schedl, "An Explorative, Hierarchical User Interface to Structured Music Repositories," Master' thesis, Dept. Austrian Research Institute for Artificial Intelligence, Vienna University of Technology, Vienna, Austria, 2003.

46. P. Knees, M. Schedl, T. Pohle and G. Widmer, "An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web," in *Proc. 14th ACM Int. Conf. Multimedia*, Santa Barbara, CA, USA, 2006, pp. 17-24.

47. A. S. Lampropoulos and G. A. Tsihrintzis, "Agglomerative Hierarchical Clustering For Musical Database Visualization and Browsing," in *Proc. 3rd Hellenic Conf. Artificial Intelligence*, Samos Island, Greece, 2004, pp. 177-186.

48. E. Brazil and M. Fernström, "Audio Information Browsing With The Sonic Browser," in *Proc. CMV'03*, London, UK, 2003, pp. 26-31.

49. G. Tzanetakis, G. Essl and P. Cook, "Automatic Musical Genre Classification of Audio Signals," in *Proc. ISMIR'01*, Bloomington, IN, USA, 2001, pp. 205-210.

50. D. Bainbridge, S. J. Cunningham and J. S. Downie, "Visual Collaging of Music in a Digital Library," in *Proc. ISMIR'04*, Barcelona, Spain, 2004, pp. 397-402.

51. A. Koutsoudis, F. Arnaoutoglou, G. Pavlidis and C. Chamzas, "3D Content-Based Visualization of Databases," presented at the Int. Conf. on KGCM, Orlando, FL, USA, 2007.

52. B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14-24, Aug. 1986.

53. J. F. Nunamaker, M. Chen and T. D. M. Purdin, "Systems Development in Information Systems Research," *Journal of Management Information Systems*, vol. 7, no. 3, pp. 89-106, Oct. 1990.

54. K. A. Boakye, "Audio Segmentation for Meetings Speech Processing," Ph.D. thesis, Dept. Elect. Eng. and Comp. Sci., Uni. California, Berkeley, 2008.

55. N. Collins, "On Onsets on-the-fly: Real-time Event Segmentation and Categorisation as a Compositional Effect," in *Proc. SMC'04*, Paris, France, 2004, pp. 219-224.

56. G. Chen, H. Tan and X.-m. Chen, "Audio Segmentation via the Similarity Measure of Audio Feature Vectors," *Wuhan University Journal of Natural Sciences*, vol. 10, no. 5, pp. 833-837, Sept. 2005.

57. L. Lu, H. Jiang and H.-J. Zhang, "A Robust Audio Classification and Segmentation Method," in *Proc. 9th ACM Int. Conf. Multimedia*, Ottawa, Canada, 2001, vol.9, pp. 203-211.

58. T. Hain, S. E. Johnson, A. Tuerk, P. C. Woodland and S. J. Young, "Segment Generation and Clustering in the HTK: Broadcast News Transcription System," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop Lansdowne*, VA, USA, 1998, pp. 133-137.

59. R. Huang and J. H. L. Hansen, "Advances in Unsupervised Audio Segmentation for the Broadcast News and NGSW Corpora," in *Proc. ICASSP'04*, Montreal, QC, Canada, 2004, vol.1, pp. 741-744.

60. C. Panagiotakis and G. Tziritas, "A Speech/Music Discriminator Based on RMS and Zero-Crossings," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 155-166, Feb. 2005.

61. G. Tzanetakis and P. R. Cook, "A Framework for Audio Analysis Based on Classification and Temporal Segmentation," in *Proc. EUROMICRO'99*, Milan, Italy, 1999, vol.2, pp. 61-67.

62. J. Saunders, "Real-time Discrimination of Broadcast Speech/Music," in *Proc. ICASSP'96*, Atlanta, GA, USA, 1996, vol.2, pp. 993-996.

63. E. Scheirer and M. Slaney, "Construction and Evaluation of a Robust Multifeature Music/Speech Discriminator," in *Proc. ICASSP'97*, Munich, Germany, 1997, vol.2, pp. 1331-1334.

64. P. J. Moreno and R. Rifkin, "Using the Fisher Kernel Method for Web Audio Classification," in *Proc. ICASSP'00*, Istanbul, Turkey, 2000, vol.4, pp. 2417-2420.

65. M. Seck, F. Bimbot, D. Zugaj and B. Delyon, "Two-class Signal Segmentation for Speech/Music Detection in Audio Tracks," in *Proc. Eurospeech'99*, Budapest, Hungary, 1999, vol.6, pp. 2801-2804.

66. M. S. Spina and V. W. Zue, "Automatic Transcription of General Audio Data: Preliminary Analyses," in *Proc. ICSLP'96*, Philadelphia, PA, USA, 1996, vol.2, pp. 594-597.

67. H. Aronowitz, "Segmental Modeling for Audio Segmentation," in *Proc. ICASSP'07*, Honolulu, HI, USA, 2007, vol.4, pp. 393-396.

68. B. Ramabhadran, J. Huang, U. Chaudhari, G. Iyengar and H. J. Nock, "Impact of Audio Segmentation and Segment Clustering on Automated Transcription Accuracy of Large Spoken Archives," in *Proc. InterSpeech'03 -Eurospeech*, Geneva, Switzerland, 2003, pp. 2589-2592.

69. B. Logan and S. Chu, "Music Summarization Using Key Phrases," in *Proc. ICASSP'00*, Istanbul, Turkey, 2000, vol.2, pp. 749-752.

70. E. Batlle and P. Cano, "Automatic Segmentation for Music Classification using Competitive Hidden Markov Models," presented at the Proc. ISMIR'00, Plymouth, MA, USA, 2000.

71. S. S. Chen and P. S. Gopalakrishnan, "Speaker Environment and Channel Change Detection and Clustering via The Bayesian Information Criterion," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998, pp. 127-132.

72. A. Tritschler and R. Gopinath, "Improved Speaker Segmentation and Segments Clustering Using the Bayesian Information Criterion," in *Proc. Eurospeech'99*, Budapest, Hungary, 1999, vol.2, pp. 679-682.

73. S.-s. Cheng and H.-m. Wang, "A Sequential Metric-based Audio Segmentation Method via the Bayesian Information Criterion," in *Proc. 8th European Conf. Speech Communication and Technology*, Geneva, Switzerland, 2003, pp. 945–948.

74. H. Meinedo and J. Neto, "A Stream-based Audio Segmentation, Classification and Clustering Pre-processing System for Broadcast News using ANN Models," in *Proc. Interspeech'05-Eurospeech*, Lisbon, Portugal, 2005, pp. 237-240.

75. L. Lu, S. Z. Li and H.-J. Zhang, "Content-based Audio Segmentation using Support Vector Machines," in *Proc. ICME'01*, Tokyo, Japan, 2001, pp. 956-959.

76. S. Lefèvre, B. Maillard and N. Vincent, "A Two Level Classifier Process for Audio Segmentation," in *Proc. 16th Int. Conf. Pattern Recognition*, Quebec, Canada, 2002, vol.3, pp. 891-894.

77. M. A. Bartsch and G. H. Wakefield, "To Catch A Chorus: Using Chroma-based Representations for Audio Thumbnailing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Platz, NY, USA, 2001, pp. 15-18.

78. M. A. Bartsch and G. H. Wakefield, "Audio Thumbnailing of Popular Music using Croma-based Representations," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 96-104, Feb. 2005.

79. M. Cooper and J. Foote, "Summarizing Popular Music via Structural Similarity Analysis," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 2003, pp. 127-130.

80. M. Goto, "A Chorus-section Detecting Method for Musical Audio Signals," in *Proc. ICASSP'03*, Hong Kong, China, 2003, vol.5, pp. 437-440.

81. L. Lu, M. Wang and H.-J. Zhang, "Repeating Pattern Discovery and Structure Analysis from Acoustic Music Data," in *Proc. MIR'04*, New York, NY, USA, 2004, pp. 275-282.

82. J. Foote, "Automatic Audio Segmentation Using a Measure of Audio Novelty," in *Proc. ICME'00*, New York, NY, USA, 2000, vol.1, pp. 452-455.

83. H. Meinedo and J. Neto, "Audio Segmentation, Classification and Clustering in a Broadcast News Task," in *Proc. ICASSP'03*, Hong Kong, China, 2003, vol.2, pp. 5-8.

84. M. K. Omar, U. Chaudhari and G. Ramaswamy, "Blind Change Detection for Audio Segmentation," in *Proc. ICASSP'05*, Philadelphia, PA, USA, 2005, vol.1, pp. 501-504.

85. S. Pfeiffer, S. Fischer and W. Effelsberg, "Automatic Audio Content Analysis," in *Proc. 4th ACM Int. Conf. Multimedia*, Boston, MA, USA, 1996, pp. 21-30.

86. C. D. Stefano, A. D. Cioppa and A. Marcelli, "An Investigation on MPEG Audio Segmentation by Evolutionary Algorithms," in *Proc. ICDAR'01*, Seattle, WA, USA, 2001, pp. 952-956.

87. M. A. Siegler, U. Jain, B. Raj and R. M.Stern, "Automatic Segmentation, Classification and Clustering of Broadcast News Audio," in *Proc. DARPA Speech Recognition Workshop*, Chantilly, VA, USA, 1997, pp. 97-99.

88. C. Rhodes, M. Casey, S. Abdallah and M. Sandler, "A Markov-chain Monte-Carlo Approach to Musical Audio Segmentation," in *Proc. ICASSP'06*, Toulouse, France, 2006, vol.5, pp. 797-800.

89. D. Kimber and L. Wilcox, "Acoustic Segmentation for Audio Browsers," in *Proc. Interface Conf.*, Sydney, Australia, 1996, vol.10, pp. 60-68.

90. G. Peeters, "A Generic System for Audio Indexing: Application to Speech/Music Segmentation and Music Genre Recognition," in *Proc. DAFx-07*, Bordeaux, France, 2007, pp. 205-212.

91. S. M. B. Homayoon, H. M. Stephane and S. S. Jeffrey, "A Distance Measure Between Collections of Distributions and Its Application to Speaker Recognition," in *Proc. ICASSP'98*, Las Vegas, NV, USA, 1998, vol.2, pp. 753-756.

92. P. Delacourt, D. Kryze and C. J. Wellekens, "DISTBIC: A Speaker-based Segmentation for Audio Data Indexing," *Speech Communication*, vol. 32, no. 1-2, pp. 111-126, Sept. 2000.

93. B. S. Ong, "Towards Automatic Music Structural Analysis: Identifying Characteristic Within-Song Excerpts in Popular Music," Master Thesis, Dept. Computer Science and Digital Communication Department of Technology, Universitat Pompeu Fabra, Barcelona, Spain, 2005.

94. G. Tzanetakis and P. R. Cook, "Experiments in Computer-assisted Annotation of Audio," in *Proc. ICAD'00*, Atlanta, GA, USA, 2000, pp. 111-116.

95. A. Ganapathiraju, L. Webster, J. Trimble, K. Bush and P. Kornman, "Comparison of Energy-based Endpoint Detectors for Speech Signal Processing," in *Proc. IEEE Southeastcon'96*, Tampa, FL, USA, 1996, pp. 500-503.

96. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons Inc, 1973.

97. B. Zhou and J. H. L. Hansen, "Unsupervised Audio Stream Segmentation and Clustering via the Bayesian Information Criterion," in *Proc. ISCLP'00*, Beijing, China, 2000, vol.3, pp. 714-717.

98. S. Z. Li, "Content-based Audio Classification and Retrieval Using the Nearest Feature Line Method," *IEEE Trans. Speech and Audio Process.*, vol. 8, no. 5, pp. 619-625, Sept. 2000.

99. K. D. Martin, E. D. Scheirer and B. L. Vercoe, "Music Content Analysis Through Models of Audition," presented at the ACM Multimedia Workshop on Content Processing of Music for Multimedia Applications, Bristol, UK, 1998.

100. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Burlington, MA: Academic Press, 2006.

101. B. Clarkson and A. Pentland, "Unsupervised Clustering of Ambulatory Audio and Video," in *Proc. ICASSP'99*, Phoenix, AZ, USA, 1999, vol.6, pp. 3037-3040.

102. A. Berenzweig, D. P. W. Ellis and S. Lawrence, "Anchor Space for Classification and Similarity Measurement of Music," in *Proc. ICME*, Baltimore, MD, USA, 2003, vol.2, pp. 29-32.

103. P. Knees, E. Pampalk and G. Widmer, "Automatic Classification of Musical Artists based on Web-Data," *ÖGAI J.*, vol. 24, no. 1, pp. 16-25, 2005.

104. D. Deng, C. Simmermacher and S. Cranefield, "A Study on Feature Analysis for Musical Instrument Classification," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 38, no. 2, pp. 429-438, Apr. 2008.

105. J. Marques and P. J. Moreno. (1999, Jun.). A Study of Musical Instrument Classification using Gaussian Mixture Models and Support Vector Machines. HP Labs. Palo Alto, CA, USA. Available: http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-99-4.pdf

106. A. Eronen, "Comparison of Features for Music Instrument Recognition," in *Proc. WASPAA'01*, New York, USA, 2001, pp. 19-22.

107. G. Agostini, M. Longari and E. Pollastri, "Musical Instrument Timbres Classification with Spectral Features," in *Proc. Int. Workshop Multimedia Signal Process.*, Cannes, France, 2001, pp. 97-102.

108. S. Essid, G. Richard and B. David, "Efficient Musical Instrument Recognition on Solo Performance Music using Basic Features," in *Proc. 25th Int. Conf. Metadata for Audio*, London, UK, 2004, pp. 2-5.

109. B. Kostek, "Musical Instrument Classification and Duet Analysis Employing Music Information Retrieval Techniques," *Proc. IEEE*, vol. 92, no. pp. 712-729, Apr. 2004.

110. J. Eggink and G. J. Brown, "Instrument Recognition in Accompanied Sonatas and Concertos," in *Proc. ICASSP'04*, Montreal, Canada, 2004, vol.4, pp. 217-220.

111. C. Joder, S. Essid and G. Richard, "Alignment Kernels for Audio Classification with Application to Music Instrument Recognition," presented at the EUSIPCO, Lausanne, Switzerland, 2008.

112. S. N. Wrigley, G. J. Brown, V. Wan and S. Renals, "Feature Selection for the Classification of Crosstalk in Multi-Channel Audio," in *Proc. InterSpeech'03-Eurospeech*, Geneva, Switzerland, 2003, pp. 469-472.

113. P. Kokoras and O. Pasquet, "Sound Scale: Perspectives on the Contribution of Flute Sound Classification to Musical Structure," presented at the 4th Conf. for Intersciplinary Musicology, Thessaloniki, Greece, 2008.

114. N. Scaringella and D. Mlynek, "A Mixture of Support Vector Machines for Audio Classification," presented at the MIREX, London, UK, 2005.

115. G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293-302, July 2002.

116. T. Lidy and A. Rauber, "Evaluation of Feature Extractors and Psycho-acoustic Transformations for Music Genre Classification," in *Proc. ISMIR'05*, London, UK, 2005, pp. 34-41.

117. J. J. Burred and A. Lerch, "A Hierarchical Approach to Automatic Musical Genre Classification," in *Proc. DAFx-03*, London, UK, 2003, pp. 8-11.

118. M. I. Mandel, G. E. Poliner and D. P. W. Ellis, "Support Vector Machine Active Learning for Music Retrieval," *Multimedia Systems*, vol. 12, no. 1, pp. 3-13, Aug. 2006.

119. G. Guo and S. Z. Li, "Content-based Audio Classification and Retrieval by Support Vector Machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 209-215, Jan. 2003.

120. Z. Cataltepe, Y. Yaslan and A. Sonmez, "Music Genre Classification Using MIDI and Audio Features," *EURASIP J. on Applied Signal Process.*, vol. 2007, no. 1, pp. 150-150, Jan. 2007.

121. S. Lippens, J. P. Martens, M. Leman, B. Baets, H. Meyer and G. Tzanetakis, "A Comparison of Human and Automatic Musical Genre Classification," in *Proc. ICASSP'04*, Montreal, Canada, 2004, vol.4, pp. 233-236.

122. M. F. McKinney and J. Breebaart, "Features for Audio and Music Classification," in *Proc. ISMIR'03*, Washington, D.C., USA, 2003, pp. 151-158.

123. Y. Feng, Y. Zhuang and Y. Pan, "Popular Music Retrieval by Detecting Mood," in *Proc. SIGIR*, Toronto, Canada, 2003, pp. 375-376.

124. L. Lu, D. Liu and H. J. Zhang, "Automatic Mood Detection and Tracking of Music Audio Signals," *IEEE Trans. Audio, Speech, and Language Process.*, vol. 14, no. 1, pp. 5-18, Jan. 2006.

125. T. Pohle, E. Pampalk and G. Widmer, "Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories," presented at the 4th Int. Workshop on CBMI, Riga, Latvia, 2005.

126. S. Essid, G. Richard and B. David, "Musical Instrument Recognition by Pairwise Classification Strategies," *IEEE Trans. Audio, Speech, and Language Process.*, vol. 14, no. 4, pp. 1401-1412, July 2006.

127. H. Homburg, I. Mierswa, B. Möller, K. Morik and M. Wurst, "A Benchmark Dataset for Audio Classification and Clustering," in *Proc. ISMIR'05*, London, UK, 2005, pp. 528-531.

128. K. El-Maleh, M. Klein, G. Petrucci and P. Kabal, "Speech/Music Discrimination for Multimedia Application," in *Proc. ICASSP'00*, Istanbul, Turkey, 2000, vol.4, pp. 2445-2448.

129. O. Turk, O. Sayli, H. Dutagaci and L. M. Arslan, "A Sound Source Classification System Based on Subband Processing," in *Proc. 3rd IEEE Benelux Signal Process. Symp.*, Leuven, Belgium, 2002, pp. 641-644.

130. T. Zhang and C.-C. J. Kuo, "Hierarchical Classification of Audio Data for Archiving and Retrieving," in *Proc. ICASSP'99*, Phoenix, AZ, USA, 1999, vol.6, pp. 3001-3004.

131. S. Ravindran and D. V. Anderson, "Audio Classification and Scene Recognition and for Hearing Aids," in *Proc. ISCAS'05*, Kobe, Japan, 2005, vol.2, pp. 860-863.

132. S. Kiranyaz, A. F. Qureshi and M. Gabbouj, "A Generic Audio Classification and Segmentation Approach for Multimedia Indexing and Retrieval," *IEEE Trans. Audio, Speech, and Language Process.*, vol. 14, no. 3, pp. 517-523, May 2006.

133. S. Srinivasan, D. Petkovic and D. Ponceleon, "Towards Robust Features for Classifying Audio in the CueVideo System," in *Proc. 7th ACM Int. Conf. Multimedia (Part 1)*, Orlando, FL, USA, 1999, pp. 393-400.

134. D. Li, I. K. Sethi, N. Dimitrova and T. McGee, "Classification of General Audio Data for Content Based Retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 533-544, Apr. 2001.

135. K. Biatov, "The Method of an Audio Data Classification and Segmentation," presented at the NATO Advanced Study Inst. on Computational Noncommutative Algebra and Applicat., Tuscany, Italy, 2003.

136. S. Ravindran, D. Anderson and M. Slaney, "Low-Power Audio Classification for Ubiquitous Sensor Networks," in *Proc. ICASSP'04*, Montreal, Canada, 2004, vol.4, pp. 337-340.

137. E. L. Bocchieri and J. G. Wilpon, "Discriminative Feature Selection for Speech Recognition," *Computer Speech and Language*, vol. 7, no. 3, pp. 229-246, July 1993.

138. S. Chu, S. Narayanan and C. C. J. Kuo, "Content Analysis for Acoustic Environment Classification in Mobile Robots," presented at the AAAI Fall Symp., Arlington, VA, USA, 2006.

139. Z. Xiong, R. Radhakrishnan, A. Divakaran and T. S. Huang, "Comparing MFCC and MPEG-7 Audio Features for Feature Extraction, Maximum Likelihood HMM and Entropic Prior HMM for Sports Audio Classification," in *Proc. ICASSP'03*, Hong Kong, China, 2003, vol.5, pp. 628-631.

140. H.-G. Kim, N. Moreau and T. Sikora, "Audio Classification Based on MPEG-7 Spectral Basis Representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 5, pp. 716-725, May 2004.

141. A. Sheh and D. P. W. Ellis, "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models," in *Proc. ISMIR'03*, Washington, D.C., USA, 2003, pp. 183-189.

142. T. Zhang and C.-C. J. Kuo, "Hierarchical System for Content-based Audio Classification and Retrieval," in *Conf. Multimedia Storage and Archiving Systems III*, Boston, MA, USA, 1998, vol.3527, pp. 398-409.

143. L. Lu, H.-J. Zhang and H. Jiang, "Content Analysis for Audio Classification and Segmentation," *IEEE Trans. Speech and Audio Process.*, vol. 10, no. 7, pp. 504-516, Oct. 2002.

144. P. J. Moreno and S. Agarwal. (2003, Dec.). An Experimental Study of Semi-Supervised EM algorithms in Audio Classification and Speaker Identification. HP Labs. Palo Alto, CA, USA. Available: http://www.hpl.hp.com/techreports/2003/HPL-2003-258.pdf

145. J. C. Brown, O. Houix and S. McAdams, "Feature Dependence in the Automatic Identification of Musical Woodwind Instruments," *JASA*, vol. 109, no. 3, pp. 1064-1072, Mar. 2001.

146. M. Davy and S. J. Godsill, "Bayesian Harmonic Models for Musical Signal Analysis," in *Bayesian Statistics 7 (Proc. the 7th Valencia Int. Meeting)*, Valencia, Spain, 2003, pp. 105-124.

147. G. d. Guo, H. J. Zhang and S. Z. Li, "Boosting for Content-based Audio Classification and Retrieval: An Evaluation," in *Proc. ICME*, Tokyo, Japan, 2001, pp. 997-1000.

148. R. S. S. Kumari, D.Sugumar and V.Sadasivam, "Audio Signal Classification Based on Optimal Wavelet and Support Vector Machine," in *Proc. ICCIMA*, Sivakasi, India, 2007, vol.2, pp. 544-548.

149. P. Dhanalakshmi, S. Palanivel and V. Ramalingam, "Classification of Audio Signals using SVM and RBFNN," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6069-6075, Apr. 2009.

150. B. Han, X. Gao and H. Ji, "Automatic News Audio Classification Based on Selective Ensemble SVMs," in *Proc. ISNN*, Chongqing, China, 2005, vol.3497, pp. 363-368.

151. K. Kashino, K. Nakadai, T. Kinoshita and H. Tanaka, "Organisation of Hierarchical Perceptual Sounds: Music Scene Analysis with Autonomous Processing Modules and a Quantitative Information Integration Mechanism," in *Proc. IJCAI*, Montreal, QC, Canada, 1995, vol.2, pp. 158-164.

152. K. Kashino and H. Murase, "A Sound Source Identification System for Ensemble Music Based on Template Adaptation and Music Stream Extraction," *Speech Communication*, vol. 27, no. 3-4, pp. 337-349, Apr. 1999.

153. P. J. Walmsley, S. J. Godsill and P. J. W. Rayner, "Multidimensional Optimisation of Harmonic Signals," in *Proc. EUSIPCO*, Island of Rhodes, Greece, 1998, pp. 2033-2036.

154. P. J. Walmsley, S. J. Godsill and P. J. W. Rayner, "Polyphonic Pitch Tracking using Joint Bayesian Estimation of Multiple Frame Parameters," in *Proc. IEEE Workshop on Audio and Acoustics*, New Paltz, NY, USA, 1999, pp. 119-122.

155. V. Mitra and C.-J. Wang, "Content Based Audio Classification: a Neural Network Approach," *Soft Computing - A Fusion of Foundations, Methodologies and Applications, Special issue on neural networks for pattern recognition and data mining*, vol. 12, no. 7, pp. 639-646, Feb. 2008.

156. A. Bugatti, A. Flammini and P. Migliorati, "Audio Classification in Speech and Music: A Comparison Between a Statistical and a Neural Approach," *EURASIP J. Applied Signal Process.*, vol. 2002, no. 4, pp. 372-378, Jan. 2002.

157. M. Liu and C. Wan, "Feature Selection for Automatic Classification of Musical Instrument Sounds," in *Proc. JCDL'01*, Roanoke, VA, USA, 2001, pp. 247-248.

158. X. He and X. Zhou, "Audio Classification by Hybrid Support Vector Machine / Hidden Markov Model," *World Journal of Modelling and Simulation*, vol. 1, no. 1, pp. 56-59, May 2005.

159. J. Lopes, C. Lin and S. Singh, "Multi-stage Classification for Audio Based Activity Recognition," in *Proc. IDEAL*, Burgos, Spain, 2006, vol.4224, pp. 832-840.

160. P. Herrera and X. Serra, "A Proposal for the Description of Audio in the Context of MPEG-7," in *Proc. CBMI'99*, Toulouse, France, 1999, pp. 81-88.

161. S. Z. Li and J. Lu, "Face Recognition using the Nearest Feature Line Method," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 439-443, 1999.

162. S. Z. Li, K. L. Chan and C. Wang, "Performance Evaluation of the Nearest Feature Line Method in Image Classification and Retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1335-1339, Nov. 2000.

163. Q. B. Gao and Z. Z. Wang, "Using Nearest Feature Line and Tunable Nearest Neighbor Methods for Prediction of Protein Subcellular Locations," *Computational Biology and Chemistry*, vol. 29, no. 5, pp. 388-392, Oct. 2005.

164. Z. Zhou, S. Z. Li and K. L. Chan, "A Theoretical Justification of Nearest Feature Line Method," in *Proc. Int. Conf. Pattern Recognition*, Barcelona, Spain, 2000, vol.2, pp. 759-762.

165. S. Z. Li, Nearest feature line, 2008. [online]. Available: http://www.scholarpedia.org/article/Nearest_feature_line. [Accessed: 15 May, 2009].

166. M. Markou and S. Singh, "Novelty Detection: A review - Part 1: Statistical Approaches," *Signal Processing*, vol. 83, no. 12, pp. 2481-2497, 2003.

167. L. Zhang, W. Zhou, Y. Lin and L. Jiao, "Support Vector Novelty Detection with Dot Product Kernels for Non-Spherical Data," in *Proc. ICIA'08*, Zhangjiajie, China, 2008, pp. 41-46.

168. X. Wu and Z.-N. Li, "A Study of Image-based Music Composition," in *Proc. ICME'08*, Hannover, Germany, 2008, pp. 1345-1348.

169. S. I. Newton, I. B. Cohen, A. Einstein and S. E. Whittaker, *Opticks: or a Treatise of the Reflections, Refractions & Colours of Light*. New York: Dover Publications, 1952.

170. X. Mao, B. Chen, G. Zhu and T. Hoshino, "Study on Transforming from Painting to Music," in *IEEE 8th Int. Symp. Video/Image Process. and Multimedia Commun.*, Croatia, 2002, pp. 117-120.

171. J. L. Caivano, "Colour and Sound: Physical and Psychophysical Relations," *Colour Research and Application*, vol. 19, no. 2, pp. 126-132, Apr. 1994.

172. K. Giannakis and M. Smith, "Auditory-Visual Associations for Music Compositional Processes: A Survey," in *Proc. Int. Comput. Music Conf.*, Berlin, Germany, 2000, pp. 12-15.

173. K. Giannakis, "A Comparative Evaluation of Auditory-Visual Mappings for Sound Visualisation," *Organised Sound*, vol. 11, no. 3, pp. 297-307, Dec. 2006.

174. D. Margounakis and D. Politis, "Converting Images to Music using Their Colour Properties," in *Proc. 12th Int. Conf. Auditory Display*, London, UK, 2006, pp. 198-205.

175. G. Ghinea and S. Y. Chen, *Digital Multimedia Perception and Design*. New York: Idea Group Publishing, 2006.

176. N. Diakopoulous, I. Essa and R. Jain, "Content Based Image Synthesis," in *3rd Int. Conf. Image and Video Retrieval*, Dublin, Ireland, 2004, vol.3115, pp. 299-307.

177. A. A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," in *Proc. ACM SIGGRAPH*, Los Angeles, CA. USA, 2001, pp. 341-346.

178. M. Ashikhmin, "Synthesizing Natural Textures," in *Proc. ACM Symp. Interactive 3D Graphics*, Durham, NC, USA, 2001, pp. 217-226.

179. V. Kwatra, A. Schödl, I. Essa, G. Turk and A. Bobick, "Graphcut Textures: Image and Video Synthesis using Graph Cuts," in *Proc. ACM SIGGRAPH*, San Diego, CA, USA, 2003, pp. 277-286.

180. A. Klein, T. Grant, A. Finkelstein and M. F. Cohen, "Video Mosaics," in *2nd Int. Symp. Non-photorealistic Animation and Rendering*, Annecy, France, 2002, pp. 22-28.

181. J. Kim and F. Pellacini, "Jigsaw Image Mosaics," in *Proc. ACM SIGGRAPH*, San Antonio, TX, USA, 2002, pp. 657-664.

182. C. S. Kaplan and D. H. Salesin, "Escherization," in *Proc. ACM SIGGRAPH*, New Orleans, LA, USA, 2000, pp. 499-510.

183. C. S. Kaplan and D. H. Salesin, "Dihedral Escherization," in *Proc. ACM Int. Conf. Graphics Interface*, London, ON, Canada, 2004, pp. 255-262.

184. Industrial Research Ltd., Video Mosaics, April 19, 2002. [online]. Available: http://www.vision.irl.cri.nz/level3/mosaics.html. [Accessed: May 14, 2009].

185. P. Pérez, M. Gangnet and A. Blake, "Poisson Image Editing," in *Proc. ACM SIGGRAPH*, San Diego, CA, USA, 2003, pp. 313-318.

186. P. J. Burt and E. H. Adelson, "A Multiresolution Spline with Application to Image Mosaics," *ACM Trans. Graphics*, vol. 2, no. 4, pp. 217-236, Oct. 1983.

187. J. Jia, J. Sun, C.-K. Tang and H.-Y. Shum, "Drag-and-Drop Pasting," in *Proc. ACM SIGGRAPH*, Boston, MA, USA, 2006, pp. 631-637.

188. V. Kwatra, I. Essa, A. Bobick and N. Kwatra, "Texture Optimization for Example-based Synthesis," in *Proc. ACM SIGGRAPH*, Los Angeles, CA, USA, 2005, vol.24, pp. 795-802.

189. S. Lefebvre and H. Hoppe, "Parallel Controllable Texture Synthesis," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 777-786, July 2005.

190. W. Matusik, M. Zwicker and F. Durand, "Texture Design using a Simplicial Complex of Morphable Textures," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 787-794, July 2005.

191. L. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," in *Proc. ACM SIGGRAPH*, New Orleans, LA, USA, 2000, pp. 479-488.

192. J. S. D. Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," in *ACM SIGGRAPH*, Los Angeles, CA, USA, 1997, pp. 361-368.

193. J. S. D. Bonet and P. Viola, "A Non-parametric Multi-Scale Statistical Model for Natural Images," in *Proc. Conf. Adv. Neural Info Processing Systems*, Denver, CO, USA, 1997, pp. 773-779.

194. Z. Bar-Joseph, R. El-Yaniv, D. Lischinski and M. Werman, "Texture Mixing and Texture Movie Synthesis using Statistical Learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 7, no. 2, pp. 120-135, Apr. 2001.

195. Dorota, Songs of the Frog, August, 1995. [online]. Available: http://allaboutfrogs.org/weird/general/songs.html. [Accessed: May 14, 2009].

196. R. N. Shepard, "Circularity in Judgments of Relative Pitch," *J. Acoust. Soc. Am.*, vol. 36, no. 12, pp. 2346-2353, Dec. 1964.

197. H. W. Lie, P. E. Dybvik and J. Rygh, "SCREAM: Screen-based Navigation in Voice Messages," in *Proc. IEEE Symp. Visual Languages*, Bergen, Norway, 1993, pp. 401-405.

198. S. Coren,"Sensation and Perception," in *Handbook of Psychology, History of Psychology*, D. K. Freedheim and I. B. Weiner, Eds., New York: Wiley, 2003, ch.5, pp. 85-108.

199. C. Ware, *Information Visualization - Perception for Design*. San Francisco, CA: Morgan Kaufmann Publishers, 2004.

200. iStockphoto LP, iStockphoto, 2007. [online]. Available: http://www.istockphoto.com/index.php. [Accessed: 27, May, 2009].

201. Comparisonics Co., FindSounds, 2007. [online]. Available: http://www.findsounds.com/. [Accessed: May 14, 2009].

202. B. Shneiderman, "Tree Visualization with Tree-maps: A 2-D Space-filling Approach," *ACM Trans. Graphics*, vol. 11, no. 1, pp. 92-99, Jan. 1992.

203. B. B. Bederson, B. Shneiderman and M. Wattenberg, "Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies," *ACM Trans. Graphics*, vol. 21, no. 4, pp. 833-854, Oct. 2002.

204. G. G. Robertson, J. D. Mackinlay and S. K. Card, "Cone Trees: Animated 3D Visualizations of Hierarchical Infromation," in *Proc. SIGCHI Conf. Human factors in Computing Systems: Reaching through Technology*, New Orleans, LA, USA, 1991, pp. 189-194.

205. G. Mori, A. Berg, A. Efros, A. Eden and J. Malik. (2004, Jun.). Video Based Motion Synthesis by Splicing and Morphing. EECS Department, University of California. Berkeley. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2004/CSD-04-1337.pdf

206. A. Schödl, R. Szeliski, D. H. Salesin and I. Essa, "Video Textures," in *Proc. ACM SIGGRAPH*, New Orleans, LA, USA, 2000, pp. 489-498.

207. S. Soatto, G. Doretto and Y. N. Wu, "Dynamic Textures," in *Proc. ICCV'01*, Vancouver, BC, Canada, 2001, vol.2, pp. 439-446.

208. A. Schödl and I. Essa, "Machine Learning for Video-based Rendering," in *Proc. NIPS'00*, Denver, CO, USA, 2000, vol.13, pp. 1002-1008.

209. N. Campbell, C. Dalton, D. Gibson and B. Thomas, "Practical Generation of Video Textures Using the Auto-Regressive Process," in *Proc. BMVC'02*, Cardiff, UK, 2002, pp. 434-443.

210. G. Doretto, E. Jones and S. Soatto, "Spatially Homogeneous Dynamic Textures," in *Proc. ECCV'04*, Prague, Czech Republic, 2004, vol.3022, pp. 591-602.

211. A. Agarwala, C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin and R. Szeliski, "Panoramic Video Textures," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 821-827, July 2005.

212. B. Celly and V. B. Zordan, "Animated People Textures," in *17th Int. Conf. Computer Animation and Social Agents*, Geneva, Switzerland, 2004, pp. 331-338.

213. M. Flagg, A. Nakazawa, Q. Zhang, S. B. Kang, Y. K. Ryu, I. Essa and J. M. Rehg, "Human Video Textures," in *Proc. Symp. Interactive 3D Graphics and Games*, Boston, MA, USA, 2009, pp. 199-206.

214. Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin and R. Szeliski, "Animating Pictures with Stochastic Motion Textures," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 853-860, July 2005.

215. K. S. Bhat, S. M. Seitz, J. K. Hodgins and P. K. Khosla, "Flow-based Video Synthesis and Editing," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 360-363, Aug. 2004.

216. G. Doretto and S. Soatto, "Editable Dynamic Textures," in *Proc. CVPR'03*, Madison, WI, USA, 2003, vol.2, pp. 137-142.

217. A. Schödl and I. Essa, "Controlled Animation of Video Sprites," in *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation*, San Antonio, TX, USA, 2002, pp. 121-127.

218. C.-H. Lai and J.-L. Wu, "A Novel Algorithm for Synthesizing Directional Temporal Texture Animation," in *Proc. 2nd IASTED*, Puerto Vallarta, Mexico, 2006, pp. 180-185.

219. J. Dong, C. Zhang, Y. Wang and H. Zhou, "A Video Texture Synthesis Method Based on Wavelet Transform," in *Proc. ICWAPR'07*, Beijing, China, 2007, vol.3, pp. 1177-1181.

220. L. Zelnik-Manor and M. Irani, "Event-Based Analysis of Video," in *Proc. CVPR'01*, Kauai Marriott, HI, USA, 2001, vol.2, pp. 123-130.

221. B.-L. Yeo and M. M. Yeung, "Retrieving and Visualizing Video," *Communications of the ACM*, vol. 40, no. 12, pp. 43-52, Dec. 1997.

222. H. Zhang, C. Y. Low, S. W. Smoliar and J. H. Wu, "Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution," in *Proc. 3rd ACM Int. Conf. Multimedia*, San Francisco, CA, USA, 1995, pp. 15-24.

223. S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram and D. Zhong, "VideoQ: an Automated Content based Video Search System using Visual Cues," in *Proc. 5th ACM Int. Conf. Multimedia*, Seattle, WA, USA, 1997, pp. 313-324.

224. Y. Wu, Y. Zhuang and Y. Pan, "Content-based Video Similarity Model," in *Proc. 8th ACM Int. Conf. Multimedia Marina del Rey*, CA, USA, 2000, pp. 465-467.

225. Y. Peng and C.-W. Ngo, "Clip-based Similarity Measure for Hierarchical Video Retrieval," in *Proc. 6th ACM SIGMM Int. Workshop Multimedia Information Retrieval*, New York, NY, USA, 2004, pp. 53-60.

226. P. Sand and S. Teller, "Video Matching," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 592-599, Aug. 2004.

227. P. Saisan, G. Doretto, Y. N. Wu and S. Soatto, "Dynamic Texture Recognition," in *Proc. CVPR'01*, Kauai Marriott, HI, USA, 2001, vol.2, pp. 58-63.

228. T. Eiter and H. Mannila. (1994). Computing Discrete Fréchet Distance. Christian Doppler Laboratory for Expert Systems. TU Vienna, Austria. Available: http://en.wikipedia.org/wiki/Fr%C3%A9chet_distance

229. B. Aronov, S. Har-Peled, C. Knauer, Y. Wang and C. Wenk, "Fréchet Distance for Curves, Revisited," in *Proc. 14th Conf. Annual European Symposium*, Zurich, Switzerland, 2006, vol.14, pp. 52-63.

230. P. M. Phillips and G. Watson, "Generalising Video Textures," in *Proc. TPCG'03*, Birmingham, UK, 2003, pp. 8-15.

Appendix A
Audio feature extraction

This appendix gives a brief introduction to commonly used audio features. The techniques used to extract audio features used in this audio visualization system are then detailed.

Introduction

Audio features can be defined as either physical, acoustical and perceptual features [1] or dynamic features, such as timbre [2] [3]. For the purposes of audio visualization we are only interested in perceptual features. There are more than fifty commonly used perceptual and acoustical features [4]. The features that have proved most useful in audio segmentation and classification of general sounds are listed below:

- temporal features
    - volume root mean square
    - volume dynamic ratio
    - silence ratio
    - frame energy
    - zero crossing ratio
- spectral features
    - centroid
    - bandwidth
    - four sub-band energy ratios
    - pitch
    - salience of pitch
    - first two formant frequencies
    - first two formant amplitudes

Feature extraction is as defined the process in which raw samples of an audio signal are converted into a sequence of feature vectors [5]. The way in which audio features are extracted from the file varies. Some features are extracted in the time domain, some in the frequency domain, and some (such as pitch) may be calculated in either the time or the frequency domain.

It is common for these features to be used in combination and in our system we combine some of these features with the 8 order MFCCs [6]. MFCC is a compact representation of an audio spectrum that takes into account the non-linear human perception of pitch, as described by the Mel scale [3]. MFCC is particularly useful for analyzing complex music due to its low-dimensional, uncorrelated and smoothed representation of the log spectrum, its ability to discriminate between different spectral contents [7] and its ability to discard 18 differences due to pitch evolution. For the audio visualization system MFCC is favoured over cepstrum coefficients [8] because MFCC approximates the human auditory system's response more closely.

Audio feature extraction

In the first step of the feature extraction phase, the signal is differentiated (pre-emphasized) by calculating the difference between succeeding samples

$$S_n : d_n = S_n - S_{n-1} \qquad \text{(A-1)}$$

A tapered window function is applied to each block to minimize the discontinuities at the beginning and end of the frames. The window could be any type of function that cuts a short segment from the signal. We use the Hamming window function [9]. If N_S is the number of samples in the window the Hamming window h_n is defined as:

$$h_n = 0.54 - 0.46\cos\left(\frac{2n\pi}{N_S - 1}\right) \qquad \text{(A-2)}$$

A key assumption in the measurement of the characteristics of an audio signal is that the signal can be regarded as stationary over an interval of a few milliseconds. Given this assumption, the signal is divided into frames of a size comparable to the variation velocity of the underlying acoustic events. The number of sample points in a frame is

called frame size. In our system we adopted a 16 ms frame size as discussed in Chapter 3.

The next step is to perform a standard *Fast Fourier Transform* (*FFT*) [10]. The result of the Fourier transform is $F\omega$ is a Hermitian symmetric function at $\omega/2$. Only the 'real part' of $F\omega$ is used to extract the audio features.

The logarithmic spectral power P was calculated according to equation A-3 where ω_0 is set to $\omega/2$ (half sampling frequency). The logarithmic spectral power is employed because it has been reported to give better results, for audio processing, than spectral power itself.

$$P = \int_0^{\omega 0} |F(\omega)|^2 \, d\omega \qquad \text{(A-3)}$$

We used another audio feature used was the logarithmic subband power P_j. The frequency spectral power is divided into four subbands with intervals of [0 (ω_0/8)], [(ω_0/8) (ω_0/4)], [(ω_0/4) (ω_0/2)] and [(ω_0/2) ω_0]. Four values, one per interval, of P_j.are calculated according to equation A-4 where L_j and H_j are the lower and upper bounds, respectively, of subband j.

$$P_j = log(\int_{L_j}^{H_j} |F(\omega)|^2 \, d\omega) \qquad \text{(A-4)}$$

The brightness audio feature is a frequency centroid and was calculated according to equation A-5.

$$W_C = \frac{\int_0^{\omega 0} \omega \cdot |F(\omega)|^2 \, d\omega}{\int_0^{\omega 0} |F(\omega)|^2 \, d\omega} \qquad \text{(A-5)}$$

Bandwidth is calculated as the square root of the power weighted average of the squared difference between the spectral components and the frequency centroid (A-6).

$$W_C = \sqrt{\frac{\int_0^{\omega 0} (\omega - \omega_c)^2 |F(\omega)|^2 \, d\omega}{\int_0^{\omega 0} |F(\omega)|^2 \, d\omega}} \qquad \text{(A-6)}$$

The number of audio signals crossing the zero in one interval is defined as the zero-crossing rate (ZCR) and is a useful feature for noise recognition in simple audio files. Essentially it gives an approximate measure of the signal's noisiness. ZCR is computed according to equation A-7 where the *sign* function is 1 for positive *x[n]* and –1 for negative *x[n]* while *N* denotes the frame size.

$$ZCR(n) = \frac{1}{N}\sum_{n=0}^{N-1} u[\,sign(X(n)) \times sign(X(n+1))]\,)$$
$$u(x) = \begin{cases} 0 & x \geq 0 \\ 1 & x < 0 \end{cases} \qquad sign(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{A-7}$$

Pitch refers to the auditory sensation of the highness of the tone, but the term will be used here to mean *fundamental frequency* (F_0). Pitch can be measured during the production of a voiced sound, and is traditionally expressed in Hertz (Hz). Pitch estimation is a mature and complex subject that has been addressed with many different approaches in digital signal processing [11]. However, although pitch estimation is considered a mature topic, there is no single pitch estimator with perfect performance in the analysis of all kinds of sounds. Several articles report evaluation results of the most popular implementations and present comparisons between their performances [12] [13] [14] and [15]. We employed the Linear Predictive Coding (LPC) synthesis the details of which can be found in [16].

To calculate the MFCCs, feature vectors, of a frame of audio signal we followed the procedure reported by Xu et al. [17]:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the Mel frequencies.
4. Take the discrete cosine transform of the list of Mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

The parameters for MFCC calculation that we employed are now detailed. The power coefficients are filtered using a triangular bandpass filter bank. The filter bank consists

of K = 19 triangular filters. These filters have a constant mel-frequency interval, and cover the frequency range of 0~ 8000 Hz. The Mel frequency mel(f) is related to the common linear frequency f according to the following equation:

$$mel(f) = 2595 * \log_{10}(1 + \frac{f}{700}) \quad \text{(A-8)}$$

Mel frequency is proportional to logarithmic linear frequency and is employed in favour of the linear frequency because it more closely maps to human subjective aural perception.

The MFCCs are computed using equation A-9, where $S_k(k = 1, 2, \cdots K)$ is the output of the filter banks and N is total number of samples.

$$C_n = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} (\log S_k) \cos[n(k - 0.5)\pi / K] \quad \text{(A-9)}$$
$$n = 1,2,...N$$

Noise-to-signal ratio (NSR) of sounds is visualized in this thesis so the extraction of NSR is also needed. NSR extraction is problematical and no one comprehensive solution has been found. Hirsch and Ehrlicher's method "Hirsch histograms" [18] was employed in this thesis to estimate the NSR. Hirsch histograms were firstly introduced for noise estimation in speech recognition.

References for Appendix A

1 D. Gerhard, "Audio Signal Classification: An Overview," *Can. Artif. Intell.* vol. 45, pp. 4-6, Winter, 2000.

2 B. S. Ong, "Towards Automatic Music Structural Analysis: Identifying Characteristic Within-Song Excerpts in Popular Music," Master Thesis, Dept. Computer Science and Digital Communication Department of Technology, Universitat Pompeu Fabra, Barcelona, Spain, 2005.

3 B. S. Ong, "Structural Analysis and Segmentation of Music Signals," Ph.D. thesis, Dept. Dept. of Technology, Universitat Pompeu Fabra, Barcelona, 2006.

4 G. Tzanetakis and P. R. Cook, "Multifeature Audio Segmentation for Browsing and Annotation," in *Proc. WASPAA'99*, New Paltz, NY, USA, 1999, pp. 103-106.

5 L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals.* Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

6 S. Z. Li, "Content-based Audio Classification and Retrieval Using the Nearest Feature Line Method," *IEEE Trans. Speech and Audio Process.* vol. 8, no. 5, pp. 619-625, Sept., 2000.

7 D. V. Steelant, B. D. Baets, H. D. Meyer, M. Leman, J.-P. Martens, L. Clarisse and M. Lesaffre, "Discovering Structure and Repetition in Musical Audio," in *Proc. Eurofuse Workshop*, Varenna, Italy, 2002, pp. 43-48.

8 W. Jeon and B.-H. Juang, "A Study of Auditory Modeling and Processing for Speech Signals," in *Proc. ICASSP'05*, Philadelphia, PA, USA, 2005, vol. 1, pp. 929-932.

9 Eds.S. Orfanidis, *Introduction to Signal Processing.* Englewood Cliffs, New Jersey: Prentice Hall, 1995.

10 L. R. Rabiner, "On the Use of Symmetry in FFT Computation," *IEEE Trans. ASSP*. vol. 27, no. 3, pp. 233-239, Jun., 1979.

11 A. Loscos, "Spectral Processing of the Singing Voice," PhD thesis, Dept. Music Technology Group, Pompeu Fabra Univeristy, Barcelona, 2007.

12 L. R. Rabiner, M. J. Cheng, A. E. Rosenberg and C. A. McGonegal, "A Comparative Performance Study of Several Pitch Detection Algorithms," *IEEE ASSP*. vol. 24, no. 5, pp. 399-418, Oct., 1976.

13 E. Mousset, W. A. Ainsworth and J. A. R. Fonollosa, "A Comparison of Several Recent Methods of Fundamental Frequency and Voicing Decision Estimation," in *Proc. ICSLP96*, Philadelphia, PA, USA, 1996, vol. 2, pp. 1273-1276.

14 D. Gerhard. (2003). Pitch Extraction and Fundamental Frequency: History and Current Techniques. Technical report. Dept. of Computer Science, University of Regina. Regina, Saskatchewan, Canada. Available: http://www.cs.uregina.ca/Research/Techreports/2003-06.pdf

15 P. d. l. Cuadra, A. Master and C. Sapp, "Efficient Pitch Detection Techniques for Interactive Music," in *Proc. ICMC*, Havana, Cuba, 2001, pp. 403-406.

16 C. Un and S.-C. Yang, "A pitch extraction algorithm based on LPC inverse filtering and AMDF," *IEEE Trans. ASSP*. vol. 25, no. 6, pp. 565-572, Dec., 1977.

17 M. Xu, L.-Y. Duan, J. Cai, L.-T. Chia, C. Xu and Q. Tian, "HMM-based audio keyword generation," in *5th Pacific Rim Conf. Multimedia*, Tokyo, Japan, 2004, pp. 566-574.

18 H. G. Hirsch and C. Ehrlicher, "Noise Estimation Techniques for Robust Speech Recognition," in *Proc. ICASSP'95*, Detroit, MI, USA, 1995, pp. 153-156.

Appendix B

Related publications by the author

In this appendix, we provide three peer reviewed publications related to this PhD thesis.

1. **Authors:** Jessie Xin Zhang, Jacqueline Whalley, Stephen Brooks
   Title: Time Mosaics – An Image Processing Approach to Audio Visualization
   Conference: Proc. of *DAFx'08* pp. 273-280
   Year: 2008
   http://www.acoustics.hut.fi/dafx08/papers/dafx08_47.pdf

2. **Authors:** Jessie Xin Zhang, Jacqueline Whalley, Stephen Brooks
   Title: A Two Phase Method for General Audio Segmentation
   Conference: Proc. of *ICME'09* (to appear in 30th June – 2nd July)
   Year: 2009

3. **Authors:** Jessie Xin Zhang, Stephen Brooks, Jacqueline Whalley
   Title: Audio Classification based on Adaptive Partitioning
   Conference: Proc. of *ICME'09* (to appear in 30th June – 2nd July)
   Year: 2009

TIME MOSAICS - AN IMAGE PROCESSING APPROACH TO AUDIO VISUALIZATION

Jessie Xin Zhang *Jacqueline L. Whalley*

Computing and Mathematical Sciences,
Auckland University of Technology
Auckland, New Zealand
{jessie.zhang|jwhalley}@aut.ac.nz

Stephen Brooks

Faculty of Computer Science,
Dalhousie University
Halifax, Nova Scotia, Canada
sbrooks@cs.dal.ca

ABSTRACT

This paper presents a new approach to the visualization of monophonic audio files that simultaneously illustrates general audio properties and the component sounds that comprise a given input file. This approach represents sound clip sequences using archetypal images which are subjected to image processing filters driven by audio characteristics such as power, pitch and signal-to-noise ratio. Where the audio is comprised of a single sound it is represented by a single image that has been subjected to filtering. Heterogeneous audio files are represented as a seamless image mosaic along a time axis where each component image in the mosaic maps directly to a discovered component sound. To support this, in a given audio file, the system separates individual sounds and reveals the overlapping period between sound clips.

Compared with existing visualization methods such as oscilloscopes and spectrograms, this approach yields more accessible illustrations of audio files, which are suitable for casual and non-expert users. We propose that this method could be used as an efficient means of scanning audio database queries and navigating audio databases through browsing, since the user can visually scan the file contents and audio properties simultaneously.

1. INTRODUCTION

Digital audio files are used widely in a variety of fields, such as film, television, computer gaming, radio, website design and audio book production. However, the query and navigation of audio files presents unique difficulties because of the linear nature of audio. There is considerable interest in making sound signals visible because the human visual system can rapidly scan a structured page of visual information, while sound files must be examined one by one, with each requiring a much longer period of the user's attention. For example, scanning a set of 10 image mosaics representing audio files would require at most a few seconds, independent of the durations of the audio files they represent, while listening to each audio file sequentially would require the sum of the durations.

Simply put, the objective of this work is to "view" what happens in the sound sequence. This project is a new approach to the visual representation of audio files as filtered mosaic time-line images. The resultant visualization is a composite image built from simple graphical elements driven by the audio data. The component images in the mosaics are individually retrieved for the sounds found in the given audio file, where the sounds are matched to audio in a pre-built audio-image database. The relative positions of image tiles in the resulting mosaic show the time sequences of their corresponding audio clips. In addition to identifying the component sounds, the visualization conveys more subtle audio features such as loudness, pitch, noise ratio, etc, on a sound-by-sound basis. This is achieved by mapping audio properties to image filtering operations.

Figure 1. *Visualization of a heterogeneous audio file.*

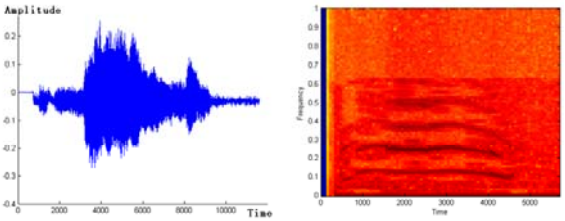

Figure 2. *Oscilloscope (left) and spectrogram (right) for a sound of cat meow.*

Figure 1 shows a simple example that characteristically depicts the results that can be generated by this system. When given a heterogeneous audio file, each single sound is extracted and searched for in a pre-built audio-image database to determine what kind of sound it is. Then an image is generated based on its audio features and the template image that represents this kind of sound in the database. When all the images for the audio clips are generated, these images are combined together seamlessly according to the audio clips' time sequences using a gradient-domain image compositing method. Using the resulting mosaics, the viewer is able to navigate amongst audio files. In addition, the viewer is able to ascertain certain audio features within the image results (e.g. noise ratio and power) at a glance, by the filtered component images.

2. RELATED WORK

Sounds maybe visualized using existing methods such as oscilloscopes in time domain analysis and spectrograms in frequency domain analysis. The oscilloscope, as shown in Figure 2 (left), is

a very common representation that expresses the audio signal by amplitude along a time axis. This might be considered the most direct visualization as it simply depicts the wave shape along the time axis. Conversely, a spectrogram represents the audio signal in the frequency domain, as shown in Figure 2 (right). The magnitudes of the windowed discrete-time Fourier transform are shown against the two orthogonal axes of time and frequency. For comparison, both Figure 2 (left) and (right) represent the same input sound of a cat's meow.

These visualizations are primarily targeted for scientific or quantitative analysis, and may not offer much insight for non-expert and casual users. For example, it would be difficult if not impossible for most users to determine that the graphs (in Figure 2) represent a cat's meow although they may be able to determine for example the average amplitude of the sound. Going beyond simple identification would be even more challenging.

There are other visualizations that are derived from note-based or score-like representations of music, typically from MIDI note events [1]. Malinowski [2] introduced "The Music Animation Machine" (MAM), that displays the music's structure with bars of color representing the notes of the music. Smith and Williams [3] used standard music notation to visualize music by examining a note-based or score-like representation of a song. A trained musician may achieve an understanding of how a musical composition will sound but unfortunately, most people cannot read music in this way. More recently, Bergstrom, Karahalios and Hart [4] provided a new method for visualizing the structure of music. Viewers can grasp the music structure by a salient image of harmony as the music structure changes over time. All these methods are intended for knowledgeable users and are strictly designed to visualize musical sound.

Audio segmentation is also related to the present work and is itself a widely researched topic. Commonly, a segmentation technique applies only to a specific application and cannot therefore be generalized. Some techniques are designed for audio samples that contain only one type of sound while others can be used for audio samples containing heterogeneous sound types. Existing methods can also be categorized into time-domain algorithms, frequency-domain algorithms or a hybrid of the two. These methods can also be further separated into supervised and unsupervised approaches depending on whether the system requires a training set to learn from, prior to audio segmentation. Segmentation approaches are also defined as model-based or non-model methods.

The work of Panagiotakis and Tziritas [5] is a time-domain approach as it uses the root-mean-square (RMS) and Zero Crossing Rate (ZCR) to discriminate speech from music. Tzanetakis and Cook [6] present a general methodology for temporal segmentation based on multiple features. In model-based approaches, Gaussian mixture models (GMM) [7], [8], Hidden Markov Models (HMM) [9], Bayesian methods [10], and Artificial Neural Networks (ANN) [11] have all been applied to the task of segmentation. Unsupervised segmentation [12] and [10] does not require training sets but instead test the likelihood ratio between two hypotheses of change and no change for a given observation sequence. On the other hand, some systems must be trained before segmentation [9],[7].

Automatic audio classification offers ways to efficiently navigate audio databases and provide the necessary control for search and retrieval within those same databases. Sounds have been classified by genre [13], [14], [15], by mood or emotion [16], [17], [18], [19], by instrumentation [20], or by segmentation and classification of an audio stream into speech, music, environmental sound and silence [21], [22].

There are many algorithms that have been applied to audio classification such as Nearest Feature Line [23], Rectified Nearest Feature Line Segment (RNFLS), Nearest Neighbour (NN), k-Nearest neighbor (k-NN), GMM, probabilistic neural network (PNN), Nearest Center (NC) and Support vector machines (SVMs) [24], [18]. A number of studies have been undertaken that compare the accuracy between these classification methods [24], [23], [25]. SVMs, together with a binary tree recognition strategy, a distance-from-boundary metric, and suitable feature combination selection, can obtain an error rate of 10% [24]. The Nearest Feature Line method has also been reported to achieve satisfactory accuracy (9.78% error) if a suitable feature combination is employed [23].

The present work defines a general method for audio visualization, which is more accessible than the existing methods because the filtered images used in the application are directly representative of sounds in the real world.

3. VISUALIZATION FRAMEWORK OVERVIEW

Figure 3 shows the framework of our audio visualization system, which contains a pre-built audio-image database and three modules: Segmentation, Classification and Image Generation.

The input to the system is a single audio file, which may contain a single sound or multiple sounds. The Segmentation module separates the input audio into audio clips, where each contains a single sound. Then each audio clip is classified by the Classification module, which retrieves the template images for the class that each audio clip belongs to. The Image Generation module then generates a distinctive image based on the computed audio features found in the sound clip and its class template image.

The output of the system is an image mosaic, which is composed of all images produced by the Image Generation module, according to the sequence order and duration of audio clips. From the mosaic viewers can determine what sounds exist within a given audio file and can scan visual cues which help the user to determine each clip's audio features.

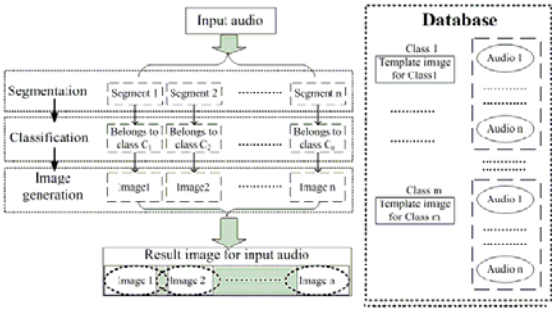

Figure 3. *Framework of the audio visualization system.*

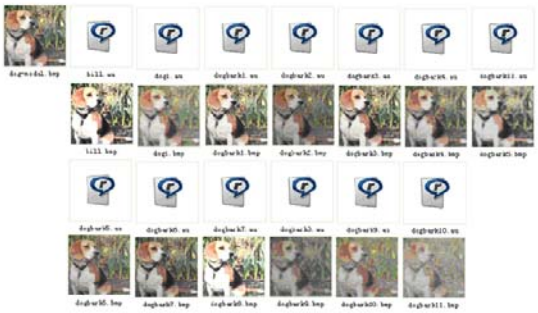

Figure 4. *A class in the database.*

3.1. The Audio-Image Database

The audio-image database is a key component of the system. Sets of audio-image pairs are stored in the database, together with their classification and training information. This comprises a library that associates sounds with archetypal images.

Figure 4 illustrates a class in the database where each class, based upon a fixed and pre-determined ontology, contains a single manually chosen template image, shown top-left. The remaining images that are associated with each audio file in the class are automatically generated by the system based on the template and each file's audio properties. The differences in audio features between the audio files in a class determine the different visual features of their corresponding images. As new audio files are added to the database, the audio is analyzed and a new representative image is produced.

A set of audio files known as the "Muscle Fish" database is widely used as a standard non-musical sound database for the development of audio matching and classification algorithms [26], [23]. There are 16 classes and 410 audio files in the Muscle Fish database, as described in Table 1, with N_c being the number of sounds in each class.

The performance of the Segmentation and Classification modules were initially evaluated using the Muscle Fish database as a benchmark. However, Muscle Fish is not entirely suitable for our application because some of the audio files are too broadly classified. In the "animals" class, for example, there are 9 sounds that belong to 6 different kinds of animal namely: cats (kittens), chickens, dogs, ducks (and geese), horses and pigs. Such a set cannot be meaningfully represented by single archetypical template image. Therefore in order to fully test our time mosaic generation we built a new database named VisualData (in Table 2) that is based on the Muscle Fish database. The VisualData training set has 602 audio files classified at a more suitable granularity. A discussion of our experiments based on these databases is discussed in the following section.

Table 1 & 2: *MuscleFish and VisualData Ontological Structures*

Muscle Fish Classes	N_c
Alto trombone	13
Animals	9
Bells	7
Cello (bowed)	47
Crowds	4
Female Voice	35
Laughter	7
Machines	11
Male Voice	17
Oboe	32
Percussion	99
Telephone	17
Tubular Bells	20
Volin (bowed)	45
Volin (pizz)	40
Water	7
Total	410

VisualData Class	N_c
Bee	36
Bell	14
Bird	15
Cat	45
Cow	79
Duck	15
Dog	13
Frog	99
Rooster	23
Alto trombone	13
Cello (bowed)	47
Oboe	32
Tubular Bells	20
Volin (bowed)	45
Volin (pizz)	40
Telephone	66
Total	602

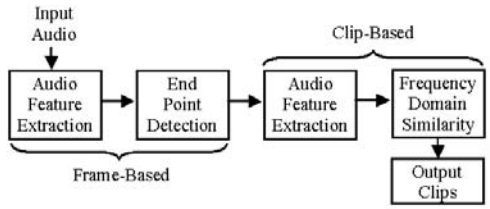

Figure 5. *Segmentation module framework.*

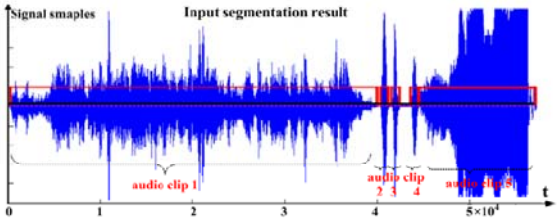

Figure 6. *Segmentation result for long and short signals.*

4. AUDIO SEGMENTATION AND CLASSIFICATION

The first module in our system pre-processes the input audio and performs any required segmentation. The goal of the audio segmentation process is to determine the beginning and ending positions (end-points) for each sound if there is no overlap between the connected audio clips. Where overlapping regions exist, the goal becomes the identification of the overlapping areas between the connected audio clips.

General audio files may have a short period of silence between sounds in the cases where sound clips do not overlap. Therefore silence detection can often be used to separate sounds in the same file. Alternatively, when we cannot rely on the presence of silence, we require a more sophisticated approach to separation of the sound clips. Because a sound tends to be homogeneous, in terms of its audio features, any abrupt changes in audio features may indicate the start of a new sound clip. We can therefore exploit time-domain features to detect silences, and then use frequency domain features to separate any remaining sound clips. Figure 5 shows the stages of this segmentation process.

Our segmentation method is designed for both long and short duration clips. Figure 6 shows results obtained by our segmentation module. In this example, there are 5 audio clips. The 1st and

5th have a relatively long duration and the rest are relatively short. The resulting segmentation is demarked with red lines over the original waveform. After segmentation a series of sound clips has been generated where each sound clip is a segment of audio, containing a homogeneous set of features for each frame.

Once segmentation has been achieved and the set of audio clips has been determined, each clip is then classified and subsequently stored in the database using the classification module (Figure 3). This module is concerned with determining the clips classification based on a comparison of existing sound files in the database using extracted audio features. It is important to obtain an accurate classification in order for the visualization to be meaningful. For each audio clip output from the segmentation module, a Nearest Feature Line (NFL) based classification is performed which has been reported to give the best classification results to date for audio file classification [6].

In this classification method an audio feature set is extracted which is composed of the means and standard deviations of the Total Spectrum Power, Subband Power, Brightness, Bandwidth and Pitch, as well as 8 order Mel Frequency Cepstral Coefficients (MFCCs). Details on the computation of these features can be found in [6]. This feature set is then used to determine which class an audio file is closest to and should belong to.

Our experiments using both the MuscleFish and the VisualData databases, achieved a similar accuracy for classification and segmentation. Using the Muscle Fish data we achieved a 90% accuracy of classification and using the VisualData set we achieved a similar result of 91%. All matching results in this paper were obtained using leave-one-out cross-validation.

The classification system we have developed is currently being extended so that it is capable of identifying new classes of audio files. However, a full discussion of our adapted NFL algorithm and evolving classification system for audio files is outside the scope of this paper and is to be published elsewhere.

5. IMAGE TILE GENERATION

Following audio segmentation and classification, the template images in the database are used to visualize the characteristics of the input audio. The function of this module is to generate images that clearly represent the embedded audio clips, while incorporating as many useful audio features as possible in the form of image filters.

Our system computes 36 audio features in order to perform analysis, segmentation and classification. Some of these features such as pitch and power are easily perceived by humans while others which have no metaphorical analogues in human audio perception are strictly used for audio analysis such as bandwidth and MFCCs. We therefore map only a subset of these audio features to image filtering operations. The image processing operations are sufficient to produce distinctive images for individual audio files within the same class.

5.1. Interpretable Audio Features

There are many audio features that relate to the way humans perceive sound, such as pitch, power and signal-to-noise ratio. Here we are visualizing such features so that viewers can grasp the character of the audio. We now discuss each of the audio features that we have incorporated:

1) Power is perhaps the simplest feature of an audio clip. It is related to the perceived loudness of a sound. The higher the volume (amplitude), the higher the power of the audio, and the louder a human will understand it to be.

2) Pitch is another feature that listeners can readily comprehend. In the field of music analysis, pitch perception is often thought of in two dimensions, pitch height and pitch chroma [27]. But for casual users, it may be sufficient to know that an audio clip with a high pitch sounds high and shrill, while low pitched sound is deep and soft [28]. Acoustics research indicates that pure sinusoids sound sharp and very low frequencies sound flat, compared to a purely logarithmic relationship [29]. As our aim is to let viewers grasp the difference between pitches by a glance, we need only visualize differences within classes. The precise values of the pitches are not required.

3) The Signal-to-Noise Ratio (SNR) can also be understood by the human ear. We understand it as the clarity of the signal. For example, poorly recorded audio, or audio with significant background noise will have a low signal-to-noise ratio.

Correspondingly, there are visual features that may be mapped to the described audio features, such as image brightness and contrast, the color-depth and signal-to-noise ratio in an image.

5.2. Mapping Audio Features to Visual Features

We map the audio features to a set of image processing operations as shown in the legend in Figure 7. These are the systems default audio-to-visual feature mapping, though it would be trivial to allow the user to explicitly over-ride the default mapping with his or her preferences. We first describe the mappings in general terms before defining them explicitly.

The top row illustrates how the audio SNR affects the visual noise ratio. The blue signal under the row of images denotes the pure audio signal for each sound clip. The red wrapping around the signal denotes noise added to the clear audio signals with varying degrees of white Gaussian noise added. The audio files generated become increasingly "noisy" from left to the right with the far left image representing the pure starting audio file and, accordingly, the images become noisier to represent this change in the signal to noise ratio of the input audio file.

Brightness values for the template image are also linearly scaled to fit the pitch of an input audio as is shown in the second row of Figure 7. The template image represents an audio clip with the average pitch of the class. The audio sounds high and

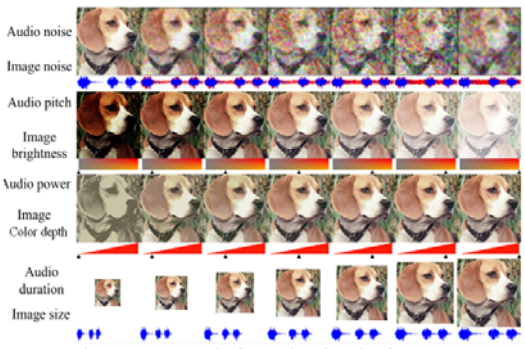

Figure 7. *Legend of visualized audio features.*

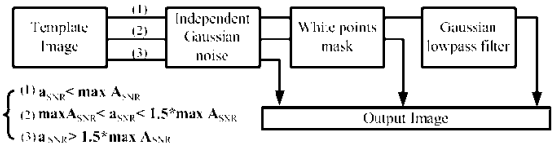

Figure 8. *Algorithm to represent the audio noise ratio.*

shrill when the pitch is high, so the image is set to be brighter for such sounds and darker for lower pitched ones. The pitch comparison is strictly within classes. For example, the pitch of a bees' buzz is always higher than an oboe's sound, but it would not convey additional information to set the brightness of bees to always be very high and the brightness of oboes to be always very dark. Therefore, the visual representation of pitch is made to be relative within the same class, meaning that the viewer can interpret a bright picture of a dog as being a relatively high pitched dog bark.

Power represents the degree of "loudness" or the volume of a sound. The higher the power value, the more clearly it can be sensed. We have chosen to map the power of an audio clip to the color depth of the image. The metaphor employed to illustrate this is that as a sound's power fades, so does the color in an image. The 3rd row of Figure 7 shows this relationship.

In addition to the above relationships, the size of an image is used to represent the duration of the audio as shown in the lowest row of Figure 7. Suppose the template image I represents class A, which audio clip a belongs to. The database contains all audio clips A_i belonging to the same class A and their audio features we denote as $A\vec{v}_i$. The audio features for the given audio clip are $a\vec{v}$. The steps of generate an image I_{output} for given audio a are detailed in the following subsections.

5.2.1. The Audio-Image Database

The audio features we used are $a\vec{v}_{pitch}$, $a\vec{v}_{power}$, $a\vec{v}_{NSR}$ and its duration. Signal-to-noise ratio is a concept defined as the ratio of a signals power to the noise power corrupting the signal. It compares the level of a desired signal (such as music) to the level of background noise. The higher the ratio, the less obtrusive the background noise is. Here we use its inverse (Noise-signal-ratio) $a\vec{v}_{NSR}$ to represent the noise level of the background to audio.

5.2.2. Calculating the brightness scale parameter

The brightness of the template image is used as the standard to represent the median pitch of a class. When the pitch of the given audio is higher than the median pitch of the class, the resulting image is brighter than the template image. For the audio with pitch lower than the median pitch of the class, the resulting image is darker than the template. The brightness is calculated as follows:

$$Brightness = \begin{cases} \dfrac{0.75 \times (a\vec{v}_{pitch} - medianA\vec{v}_{pitch})}{\max A\vec{v}_{pitch} - medianA\vec{v}_{pitch}} & if \quad a\vec{v}_{pitch} \geq medianA\vec{v}_{pitch} \\ \dfrac{0.75 \times (medianA\vec{v}_{pitch} - a\vec{v}_{pitch})}{medianA\vec{v}_{pitch} - \min A\vec{v}_{pitch}} & otherwise \end{cases} \quad (1)$$

Where $a\vec{v}_{pitch}$, is the pitch value of audio clip a; $medianA\vec{v}_{pitch}$, $maxA\vec{v}_{pitch}$, and $minA\vec{v}_{pitch}$, are the median, maximum and minimum values of the pitch values in class A. The adjusted image is then calculated as $I_{output}=I\times Brightness$, with the adjustment made within HSV color space.

5.2.3. Calculating the color depth parameter

The ColorDepth scale is calculated in 3 stages:

1) The audio clip is normalized to set the maximum of amplitude value equal to 1, producing *Normal(a)*. Then the power value is calculated as $max\ a\vec{v}_{power}$.
2) *Normal(a)* is scaled to $0.01\times Normal(a)$ calculate the power value as $min\ a\vec{v}_{power}$. This value is chosen empirically given that small amplitude is not easily heard by the human ear.
3) The color depth ranges from a maximum of 2^8 to a minimum of 2^2. The color depth scale is calculated as:

$$ColorDepth = 2^{2+6\times\frac{a\vec{v}_{power}-min a\vec{v}_{power}}{max a\vec{v}_{power}-min a\vec{v}_{power}}} \quad (2)$$

A new index image, I_{output}, is generated by using minimum variance quantization on *ColorDepth* as calculated in sect. 5.2.2. The number of colours in the new index image is at most ColorDepth.

5.2.4. Representing the audio noise-to-signal ratio

We designed three algorithms for representing different levels of Noise-to-Signal Ratios (NSR) from very clean to very noisy, as shown in Figure 8. Note that a_{NSR} is the NSR for audio clip a. To represent the NSR of a given audio, not only is independent Gaussian noise added to the template image, but also white point noise for very noisy audio files. Furthermore, a degree of blurring is introduced for extremely noisy audio files. Each of the three filtering processes is calculated as follows:

1) Independent white Gaussian noise is added to I_{output}. If a_{NSR} is less than the maximum NSR $maxA_{NSR}$ in the class it belongs to, it is the only form of visual noise that is added. The white Gaussian noise is zero-mean, with a variance of:

$$NoiseScale = 0.15 \times \frac{a\vec{v}_{NSR}}{\max a\vec{v}_{NSR}} \quad (3)$$

2) After adding independent white Gaussian noise, white points are added when a_{NSR} is between *1* to *1.5* times $max\text{-}A_{NSR}$. The number of white points is defined by:

$$PointNumber = 0.1 \times \frac{a\vec{v}_{SNR} - \max A\vec{v}_{NSR}}{0.5 \times \max A\vec{v}_{NSR}} \quad (4)$$

3) The image is blurred with *7 * 7* Gaussian low-pass filter if a_{NSR} is larger than $1.5 * maxA_{NSR}$.

Together, the three filters introduce a wide range of noise distortion in the final image.

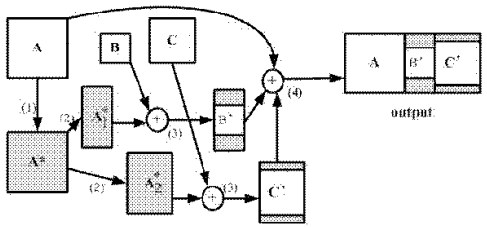

Figure 9. *Background texture generation.*

5.2.5. Resizing the output image by audio duration

Image size is used to represent the duration of the audio clip. The longer the audio lasts, the larger its image size. In order to be able to display multiple timeline mosaics simultaneously, we set maximum width of component images to 512 and the smallest to 128 pixels. If the duration of a given audio is less than all the audio files in its class $a\vec{v}_{duration} \le \min A\vec{v}_{duration}$, the width of its image tile is set to 128 to make sure the image is sufficiently clear. When the duration of a given audio clip is longer than all audio files, then the image is set to 512. Otherwise, the image size is scaled linearly between the smallest and largest size based on the audio's duration. After these five stages, an image tile is generated to represent the given audio clip. All the image tiles are generated independently.

5.3. Mosaic Time-Line Generation

It is not practical to assume that all audio clips have the same length in a general audio file. Constructing a blended image mosaic is therefore appropriate since the image sizes will vary, given that embedded audio clips are of varying lengths. Blended image mosaics solve this problem since we can embed smaller images into a larger image background.

Poisson image editing [30] is an effective approach for seamless image composition in the gradient domain. The new image is created by pasting a region from a source image onto a target image. To construct a time-line mosaic from the component images Poisson image editing [30] is employed to fuse the separate images together without significantly altering the content of each region.

The optimization process seamlessly inserts new content into a subset, Ω, of an existing image, h. It computes a new image, f, whose gradient, ∇f, within Ω is closest to the gradient ∇g, taken from a second image, g. The original boundary, $\partial\Omega$, of region Ω from h is also used as a constraint to ensure that the region Ω blends with the surrounding image, h. The final image constrains an interpolation of the boundary conditions, $\partial\Omega$, inwards while conforming to the spatial changes of the guidance field from g as closely as possible within Ω. The minimization problem is written as:

$$\min_f \iint_\Omega \left|\nabla f - \nabla g\right|^2 \text{ with } f|_{\partial\Omega} = h|_{\partial\Omega} \qquad (5)$$

The reader is directed to [30] for the discretization of the problem and for suggested iterative solvers.

For our application, the output image is generated based on rectangular images of different sizes. The challenge is to merge different-size images without significant information loss in any of the component images that form the final mosaic.

When the image tiles within the same mosaic are not the same size, this creates gaps in the mosaic at the top and bottom of the smaller tiles, which must be padded in an unobtrusive way. To achieve this we follow the stages depicted in Figure 9 to generate the final image. In this mosaic there are three tiles (A, B and C).

Tile A does not require padding and is placed directly into the output mosaic. Tiles B and C are smaller and require varying degrees of padding. For this, the largest tile (A) is selected to generate a seamless background texture to pad the smaller tiles. As most of the important information within our archetypal images is in the center, we conservatively scale padding image A to 90% of its original size and integrate the result into the original tile A with Poisson blending to generate additional, low-content padding. This process repeats 10 times until the final image A* has the same size as the original image A, but with its content comprised of edges from the original image.

The width of the background texture image A* is then scaled to fit the width of each image tile (see note (2) in Figure 9). The A_1* is used for the background texture for image B and the A_2* is used for C. Tile B is then merged into the middle of A_1* with Poisson blending to generate image tile B'. When all the image tiles are padded, they are merged seamlessly by Poisson image editing in order of their time sequences.

6. EXPERIMENTAL RESULTS

We now discuss a set of experimental results. The input audio of the first example includes a long bird's sound, 3 short dog barks and a relatively long cat's meow, as shown in different colors in the bottom of Figure 10. Note that the power or amplitude of the sounds cannot be read directly from the signal at the bottom of Figure 10 because the audio clips are normalized to show their shapes. This is necessary for audio clips with small amplitudes.

After segmentation, the input audio file is separated into 5 audio clips each containing only one sound. The three dog's barks are separated into 3 clips each containing only one sound. After classification, we know which classes the audio clips belong to, and have the template images for each audio clip. A result generated based on these template images is shown in Figure 10. This result indicates what sounds are included in the given audio file, their duration, but have not been subjected to image filtering.

Figure 11 is a result generated based on both the template image and the audio features of each audio clip. Comparing Figure 10 (without filtering) and Figure 11 (with filtering) we can see that the cat meow in the given audio is very noisy because the tile is quite noisy and blurry. The noise ratio of dog's sound is lower compared with the cat's sound and the sound of the bird has the lowest noise ratio of the three.

According to the legend in Figure 7, from the ColorDepth and brightness in the Figure 11, we can read that the power of the bird's sound is relatively low and its pitch is relatively high compared with most audio files in the same class. Comparing Figures 11 and 12, one can easily determine information such as which audio clip is noisier, which has a higher pitch and which has more power value. Figures 13 and 14 show two further examples of randomly generated audio files, and Figure 15 shows an example with 5 detected sound clips. Figure 17 gives an example for random generated audio for music, which contains 3 music clips: oboe, trombone and violin.

Figure 10. *Bird-dog-cat example without filtering.*

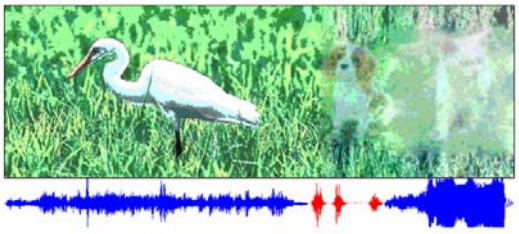

Figure 11. *Bird-dog-cat example with filtering.*

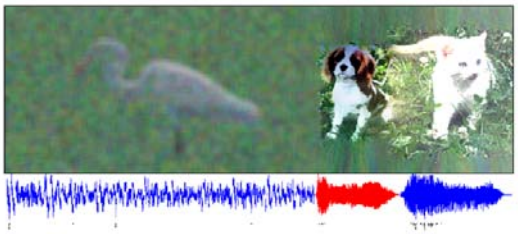

Figure 12. *Filtered bird-dog-cat example with different features.*

Figure 13. *Frog-telephone-rooster example.*

Figure 14. *Bee-Cow-Rooster example.*

Figure 15. *Results with 5 images.*

Figure 16. *Bird-dog-cat example with alternate template images.*

Figure 17. *Oboe-trombone-violin.*

7. LIMITATIONS

There remain some limitations of our system, which may lead to spurious results under certain conditions. The template images of the database partly determine the quality of the final result. The requisites for a good template image include: the color and contrast of the image, the position of important information and whether the image can completely describe the class it represents. Figure 16 is an image result for the same sound but with different template images for the classes in VisualData. In this result, because the most important information (the dog) of the template image of class "dog bark" is very close to the left edge, part of the audio clip is not clearly represented.

The accuracy of visualizing a given sound with time mosaics also depends on the quality of the pre-built database, the accuracy of audio segmentation and classification. If given a sound which contains any audio clip doesn't belong to any class of the database, the system cannot recognize it and does not yet have the ability to notify the user about the accuracy of audio segmentation and classification. However, as noted previously, we are currently augmenting our system to handle this issue.

Figure 18 illustrates a limitation in our audio processing method. The sound file contains a cat-bird-dog sequence of sounds. The bird sound in this file has amplitude that is too low for our segmentation method to detect and results in a time mosaic with only 2 component images. If the amplitude of the bird sound in this file is increased by a factor of 2 then our segmentation method detects the bird sound and accurately produces a time mosaic with 3 component images (Figure 19)

Another possible source of error can occur in situations where sounds overlap. Figure 20 shows an audio signal that consists of a cat's meow and a bird sound in which there is a period of overlap between them. The segmentation failed to separate the 2 sounds, resulting in the sound file being treated as a single sound and classified as a rooster for which a single image was produced.

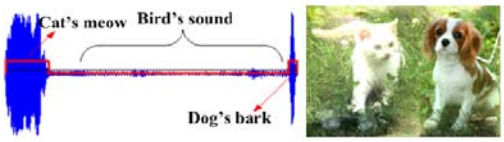

Figure 18. *Three sounds misrepresented as two images.*

Figure 19. *Three sounds correctly represented as three images.*

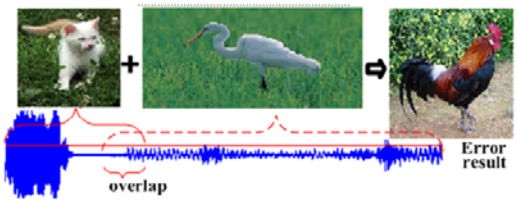

Figure 20. *Overlapping audio signals producing an incorrect result.*

8. CONCLUSIONS AND FUTURE WORK

This paper presents a novel approach to visualizing a given audio file with time mosaics, constructed from a pre-built audio database. The mosaics provides produce a sequence of images seamlessly joined together using the Poisson image editing method. Each filtered image tile in the mosaic represents a single monophonic sound and its features, in the given input audio. We argue that the system would provide a useful means to quickly scan audio query results and navigate an audio database. There are a number of opportunities to further develop the ideas introduced in this paper. For example, the concept of time mosaics could be extended and used as a basis for generating video to represent sound files. Other mapping from audio features to image processing filters could also be explored, including NPR filters.

9. REFERENCES

[1] J. Foote. Visualizing Music and Audio using Self-Similarity. In *International Multimedia Conference*, pp 77-80, 1999.

[2] S. Malinowski. Music Animation Machine. 2007 [cited; Available at: http://www.well.com/user/smalin/mam.html].

[3] S.M. Smith and G.N. Williams. A Visualization of Music. In *Proc. Visualization '97, ACM*, pp 499-503. 1997.

[4] T. Bergstrom, K. Karahalios and J. Hart. Isochords: Visualizing Structure in Music. *Graphics Interface*, pp: 297-304, 2007.

[5] C. Panagiotakis and G. Tziritas. A Speech/Music Discriminator Based on RMS and Zero-Crossings. *IEEE Transactions on Multimedia*, Volume 7, pp 155-166. July, 2004.

[6] G. Tzanetakis and P. Cook. Multifeature Audio Segmentation for Browsing and Annotation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp 103-106, 1999.

[7] M. Spina and V. Zue. Automatic transcription of general audio data: preliminary analyses. In *Spoken Language*, pp 594-597, 1996.

[8] H. Aronowitz. Segmental Modeling for Audio Segmentation. In *Acoustics, Speech and Signal Processing*, pp 393-396, 2007.

[9] B. Ramabhadran, J. Huang, U. Chaudhari, G. Iyengar and H.J. Nock. Impact of audio segmentation and segment clustering on automated transcription accuracy of large spoken archives. In *Proc. EuroSpeech*, pp 2589-2592, 2003.

[10] S. Chen and P. Gopalakrishnan. Speaker Environment and Channel Change Detection and Clustering via The Bayesian Information Criterion, *DARPA speech recognition workshop*, pp 679-682.

[11] H. Meinedo and J. Neto. A Stream-based Audio Segmentation, Classification and Clustering Pre-processing System for Broadcast News using ANN Models. In *INTERSPEECH*, pp 237-240, 2005.

[12] M.K. Omar, U. Chaudhari and G. Ramaswamy. Blind Change Detection for Audio Segmentation. In *Acoustics, Speech, and Signal Processing*, pp 501-504, 2005.

[13] Z. Cataltepe, Y. Yaslan and A. Sonmez. Music Genre Classification Using MIDI and Audio Features. *EURASIP Journal on Advances in Signal Processing*, pp 150-157, 2006.

[14] S. Lippens et al. A comparison of human and automatic musical genre classification. In *IEEE International Conference on Audio, Speech and Signal Processing*, volume 4, pp 233-236, 2004.

[15] M.F. McKinney and J. Breebaart. Features for audio and music classification. *ISMIR*, pp 151-158, 2003.

[16] Y. Feng, Y. Zhuang and Y. Pan, Popular music retrieval by detecting mood. *SIGIR*, pp 375-376, 2003.

[17] L. Lu, D. Liu and H.-J. Zhang. Automatic Mood Detection and Tracking of Music Audio Signals. *IEEE Transactions On Audio, Speech, And Language Processing*, pp 5-18.

[18] M. Mandel, G. Poliner and D.Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12, pp 3-13.

[19] T. Pohle, E. Pampalk and G. Widmer. Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories. *Proceedings of the 4th International Workshop on Content-Based Multimedia Indexing*, 2005.

[20] S. Essid, G. Richard and B. David. Musical instrument recognition by pairwise classification strategies. *IEEE transactions on audio, speech and language processing*, pp 1401-1412, 2006.

[21] S. Kiranyaz, A. Qureshi and M. Gabbouj. A Generic Audio Classification & Segmentation Approach for Multimedia Indexing & Retrieval. *Audio, Speech and Language Processing*, 14, pp 1062-1081.

[22] L. Lu, H. Jiang and H. Zhang. A robust audio classification and segmentation method. In *9th ACM international Conference on Multimedia, MULTIMEDIA '01*, pages 203-211, 2001.

[23] S.Z. Li. Content-based audio classification and retrieval using the nearest feature line method. *IEEE transactions on speech and audio processing*, volume 8, pp 619-625, 2000.

[24] G. Guo and S.Z. Li. Content-based audio classification and retrieval by support vector machines. In *Neural Networks, IEEE Transactions on*, pp 209-215, 2003.

[25] M. Liu and C. Wan. Feature Selection for Automatic Classification of Musical Instrument Sounds. *ACM*, pp 247-248, 2001.

[26] E. Wold, T. Blum, D. Keislar and J. Wheaton. Content-Based Classification, Search, and Retrieval of Audio. *Multimedia, IEEE*, volume 3, pp 27-36, 1996.

[27] R. Shepard. Circularity in Judgments of Relative Pitch. *The Journal of the Acoustical Society of America*, 36(12), pp 2346-2353.

[28] H.W. Lie, P.E. Dybvik and J. Rygh. SCREAM: screen-based navigation in voice messages. In *Visual Languages*, pp 401-405, 1993.

[29] S. Coren, L.M. Ward and J.T. Enns. Sensation and Perception. 1994: Harcourt Brace College Publisher, Toronto.

[30] P. Perez, M. Gangnet and A. Blake. Poisson Image Editing. In *International Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH 2003)*, pp 313-318.

A TWO PHASE METHOD FOR GENERAL AUDIO SEGMENTATION

Jessie Xin Zhang[1], *Jacqueline Whalley*[1], *Stephen Brooks*[2]

[1] School of Computing and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand {jwhalley, jessie.zhang@aut.ac.nz}
[2] Faculty of Computer Science, Dalhousie University, 6050 University Avenue, Halifax, Nova Scotia, Canada B3H 1W5 {sbrooks@cs.dal.ca}

ABSTRACT

This paper presents a model-free and training-free two-phase method for audio segmentation that separates monophonic heterogeneous audio files into acoustically homogeneous regions where each region contains a single sound. A rough segmentation separates audio input into audio clips based on silence detection in the time domain. Then a self-similarity matrix, based on selected audio features in the frequency domain to discover the level of similarity between frames in the audio clip, is calculated. Subsequently an edge detection method is used to find regions in the similarity image that determine plausible sounds in the audio clip. The results of the two phases are combined to form the final boundaries for the input audio. This two-phase method is evaluated using established methods and a standard non-musical database. The method reported here offers more accurate segmentation results than existing methods for audio segmentation. We propose that this approach could be adapted as an efficient pre-processing stage in other audio processing systems such as audio retrieval, classification, music analysis and summarization.

Index Terms— Audio segmentation, similarity map, edge detection

1. INTRODUCTION

Segmentation plays an important role in audio processing applications, such as content-based audio retrieval recognition and classification, and audio database management. Audio segmentation is a process that divides an audio file into its composite sounds. Each segment or clip should consist of a single sound that is acoustically different from other parts of the audio file. An accurate segmentation process can identify appropriate boundaries for partitioning given audio streams into homogeneous regions.

Although there are many approaches to audio segmentation they are focused on a narrow type of audio such as speech/music separation, speaker recognition and music structure extraction. These methods work well for specific tasks but are not generalisable even, for example, across different genres of music. A method to segment any given audio file that contains different types of sounds remains an open and important problem.

2. RELATED WORK

Several methods have been developed for audio segmentation. Chen identifies two types of segmentation approaches namely, classification-dependent segmentation (CDS) and classification-independent segmentation (CIS) [1]. CDS methods are problematical because it is difficult to control the performance [1].

CIS approaches can be further separated into time-domain and frequency-domain depending upon which audio features they use, or supervised and unsupervised approaches depending on whether the approach requires a training set to learn from prior audio segmentation results. CIS may also be defined as model-based or non-model based methods.

The work of Panagiotakis and Tziritas [2] is a typical time-domain approach and uses the root-mean-square (RMS) and zero crossing rate (ZCR) to discriminate speech from music. Tzanetakis and Cook [3] presented a general methodology for temporal segmentation based on multiple features. In model-based approaches, Gaussian mixture models (GMM) [4], [5], Hidden Markov Models (HMM) [6], Bayesian [7], and Artificial Neural Networks (ANN) [8] have all been applied to the task of segmentation. Examples of an unsupervised audio segmentation approach can be found in [7] and [9]. These unsupervised approaches test the likelihood ratio between two hypotheses of change and no change for a given observation sequence. On the other hand, the systems developed by Ramabhadran et al. [6] and Spina, and Zue [4] must be trained before segmentation. These existing methods are limited because they deal with limited and narrow classes such as speech/music/noise/ silence.

Audio segmentation methods based on a similarity matrix, have been employed for broadcasting news, which is relatively acoustic dissimilar, and for music to extract structures or music summarizations. The accuracy evaluation of these methods was undertaken with specific input audios and has not been previously reported for use with audio files in a non-music/non-speech database. This

paper introduces a two phase unsupervised model-free segmentation method that works for general audio files. In this paper, we discuss the process by which we developed and evaluated an efficient CIS method that can determine segment boundaries without being supplied with any information other than the audio file itself.

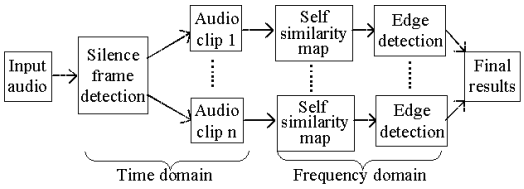

Fig. 1. Frame work for audio segmentation method.

3. THE TWO PHASE SEGMENTATION METHOD

The first stage of the segmentation method roughly separates audio by using silence detection in the time domain. After this initial segmentation, further segmentation in the frequency domain is performed. This is done by taking advantage of the fact that sounds tend to be homogeneous, in terms of a sounds audio features. We make the assumption that any abrupt change in the audio features indicates the start of a new sound clip. This more sophisticated approach is performed on segments from initial segmentation to detect subtle changes based on more complex acoustic features until each audio clip contains a single sound.

3.1. Time Domain Silence Segmentation

The first phase of our segmentation finds the start and end of audio clips based silence periods. Root Mean Square (RMS) [2] and Spectral Power (SP) [10] are widely used audio features in silence detection. If the signal in a frame is less than a pre-determined threshold [10], it is regarded as a silence frame. We have found that neither RMS nor SP are suitable when the sound's duration is less than 10ms, (e.g. hand claps in Fig.2).

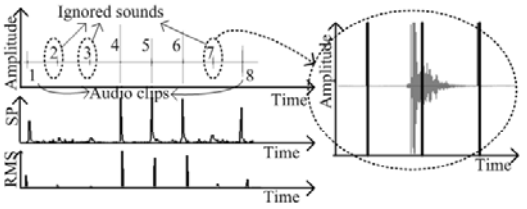

Fig. 2. (L) Wave shape of audio file "Hands clapping" and its SP, RMS; (R) Enlarged wave shape of a sound in "Hands clapping".

The signal in Fig 2 contains 8 hand claps, each lasting ~ 11ms. Using both RMS and SP, only 5 are detected. The remaining 3 clearly audible sounds that are visually identifiable in the time-amplitude signal shapes are not detected. When sound 1 and sound 7 are compared (the top image in Fig.2 (L)) the SP values of the two sounds are similar if we manually mark their boundaries. However sound 7 is not identified as signal in the input audio. This is because when the input audio is split into frames, the Sound 7 is cut into two frames (Fig.2 (R)) which contain short limited sample points, resulting in two frames being detected as silence. Neither RMS nor SP works for short duration signals such as this hand clapping example.

In order to cope with very short sounds we propose a new segmentation method as follows. The maximum amplitude value of each frame is used as a parameter to label silence and signal frames with an empirical threshold T. The threshold T adapts to each audio file, as follows:

$$T = \begin{cases} 0.015 \times \max(A), \textit{if } mean(A) > 0.075 \times \max(A) \\ 0.05 \times \max(A), \textit{ if } mean(A) \le 0.075 \times \max(A) \end{cases} \quad (1)$$

where A is the absolute amplitude of the signal. Adapting the threshold using the mean value of the absolute amplitudes of an input sound is necessary to minimize the effects of noise signals within audio signals of variable SP. This results in higher thresholds that eliminate most of the noise in a signal but may also result in the removal of the edges of a fading signal. This tradeoff between loss of signal edges and elimination of noise is necessary to achieve a good segmentation.

We separate the given audio into frames, each lasting 2.5ms. Such a short frame size is not appropriate for accurate audio feature extraction. But in our method audio features are not required in this phase. We have found that a 2.5ms frame leads to accurate silence detection for signal boundaries using the maximum amplitude.

Once all the frames have been marked, the start and end points for each audio clip are detected. Given frame f_i if f_{i-1} and f_{i-2} are silent then f_i is a clip start frame f_s and if $f_{i+1}$and f_{i+2} are silent then f_i is a clip end frame f_e. The clip is then defined as $C \in \{f_s, f_{s+1}, \ldots f_e\}$. If the silence between two audio clips C_i and C_{i+1} is short (less than 0.01 × *input_audio_duration* for general sound) these two audio clips are combined. If an audio clip C_i is very short (less than 5 ms in duration), it is hard to perceive audibly and therefore does not need to be regarded in further analysis and is discarded as noise. If a result sound clip lasts more than 5ms but less than 16ms, the frame in front of it and the frame after it are added to it to form a signal clip. The result of the first phase is a set of sound clips (signal frames).

Fig. 3 gives an example of segmentation results based on silence detection. There are 2 signals in the input audio. The first audio is a telephone's touch tones, (marked in gray) and the second is a violin (marked in black). Using our adaptive threshold based on amplitude the silence periods between the telephone touch tones are correctly detected.

But if 2 sounds are seamlessly connected with no silence or very short silence in between, such as clip 7 in the telephone/violin audio, silence detection does not work. It is necessary to undertake a second segmentation refinement phase to deal with contiguous adjacent signals.

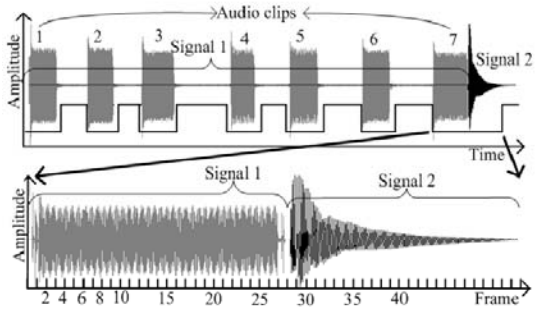

Fig. 3. Audio segmentation result for a mixed sound audio file.

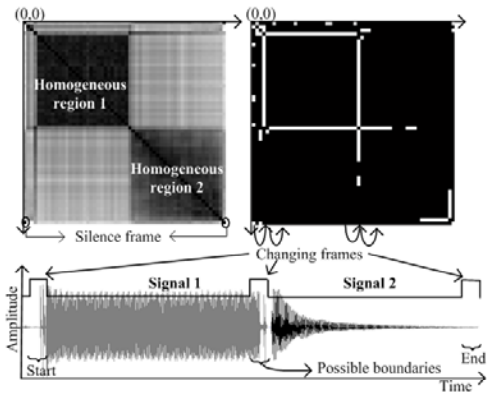

Fig. 4. Top: (L) Similarity map image; (R) Edge detection of similarity map image; Bottom: Segmentation result based on similarity map image.

3.2. Frequency Domain Similarity Segmentation

For any resultant audio clips from the first phase, if it needs further segmentation, a sophisticated segmentation approach is used to detect if there are subtle changes in acoustic features.

We adapt Foote's similarity map [11] into a general sound segmentation procedure. Foote first introduced the similarity map to identify significant changes in music. A checkerboard kernel is used to calculate cross-similarity values between different regions along the diagonal of the similarity map. Kernel size limits the accuracy, a small kernel tends to be sensitive and on a short time scale is capable of finding detailed changes. On the other hand, a large kernel is suitable for those audio files that have few segments but each segment contains several sounds. In a similar vein, Cooper [12] used a 2D similarity matrix for music summarization. Because it is impossible to find a kernel size that suits all kinds of sounds, we developed an edge detection method using image processing techniques, instead of a kernel comparison, based on a similarity map that finds changes in audio properties.

Generally speaking, accurate audio feature extraction requires sounds of sufficient duration (such as the standard frame size 16ms in [13]). We perform further segmentation on an audio clip if it is longer than 160ms (10 frames). For the audio clips shorter than 160ms, there is little possibility of them containing more than one distinguishable sound. In the second phase we process any audio clips with duration of 160ms or longer that were generated in phase 1 and separate them into 16ms non-overlapping frames.

An audio feature vector is then extracted for each frame. The audio features are: Total Spectrum Power, Bandwidth, Brightness, Pitch and an 8 order Mel Frequency Cepstral Coefficient (MFCC). This feature set was selected as it has been shown to give the most accurate classification result and can therefore best represent the difference between classes. Details of the computation of these features can be found in [13].

A similarity matrix of these audio vectors is then calculated, where each element represents a distance of two frames. This similarity matrix is normalized and is represented by a gray scale similarity map (image) for the audio clip. If there is a single sound in the audio clip, the similarity map is homogeneous. Otherwise a similarity map illustrates the differences between the audio clips' frames. By applying Sobel edge detection [14] on the similarity map, a heterogeneous image can be separated into homogeneous regions. The edge pixels of the regions represent the boundaries in the audio clip. In this way, the audio clip is then separated into homogeneous sound regions.

For a homogeneous audio clip, the similarity distances are similar before normalization, but are extended in the similarity map. To represent the audio clip correctly, two silence frames are added at the front and the end of the audio clip. After adding a silence frame, the distances between the signal frames in the audio clip are compared with silence frames. Homogeneous audio clips will not be extended. The similarity is measured and compared using 5 different distance metrics, as discussed in section 4.

We use the audio clip in Fig. 3 (lower image) as an example; its similarity map is the left image of the top row in Fig. 4. The similarity map has zeros along its diagonal because the distance is zero from each frame to itself. There are 2 homogeneous blocks in the similarity map that indicate two homogeneous sounds in the audio clip. Fig. 4Top(R) gives the resultant image from performing Sobel edge detection on the self-similarity map (Fig. 4Top(L)). The changing frames are clearly evident from the result edge detection image (Fig. 4Top(R)). The final segmentation result for the phone and violin mixed audio clip is illustrated in Fig. 4 (bottom image). We can see that the edges of similarity map highly illustrate the boundaries of the sounds.

4. SEGMENTATION METHOD EVALUATION

To fully evaluate this novel segmentation method we employ a widely used standard non-musical sound database called MuscleFish [13]. MuscleFish contains 16 classes and

410 audio files. When contrasted with previously reported segmentation evaluations this database introduces some significant segmentation issues. Some classes in MuscleFish contain many audio files that are very similar in acoustic audio features. Moreover, the database has a broad range of classes including, music, speech, animal sounds and everyday sounds (e.g. thunderstorm and hand clapping).

For segmentation evaluation, we generated 10 groups, of 100 audio files. Each file is generated by conjoining two randomly selected files from the MuscleFish database. As the audio files in MuscleFish contains up to 7 sounds, the generated audio files may contain up to 14 segments. To evaluate how our segmentation method works, we compared the two phase method with a Euclidean similarity map, a cosine angle and silence detection (Table 1). To fully evaluate the use of Euclidean distance in our method, we also tested the two phase method using a cosine angle in the second phase. After extracting the audio features for all the frames, for each feature, the values of all audio frames are normalized to [0, 1]. This ensures each feature has the same weight in the Euclidean distance calculation.

A tolerance frame before and after the true boundary is integrated for evaluation. This small tolerance is introduced to allow frames for what contains more than one signal. If two signals belong to the same class, we regard any segmentation result as "correct" because they are similar in acoustic features.

Table 1 shows the segmentation accuracy using different methods. Our two-phase method with Euclidean distance in a similarity map gives the most accurate segmentation result with an average of 91.9%. The result using silence detection gives the worst average result (81%).

When comparing the result of using the cosine angle and the Euclidean distance, the latter gives better accuracy regardless of the approach (two-phase method or the similarity map alone). Compared with using a similarity map alone, our two-phase method gives improved accuracy (see Fig. 5).

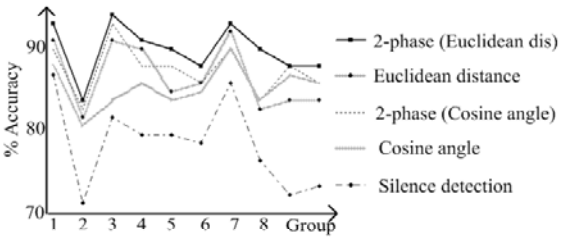

Fig. 5. Segmentation accuracy using various methods.

5. CONCLUSION

This paper provides a model-free and training-free two-phase method for general audio segmentation. From evaluation, because there are a wide variety of classes in the MuscleFish database, we believe our two-phase segmentation method provides a good solution to general audio segmentation problems. As a power segmentation tool, this method outperforms typical CDS methods or model-based approaches reported in the literature. Moreover, by judicious design of the feature set, this method can be tailored for other applications such as music structure analysis and summarization in music field and speaker detection in speech analysis as well as audio retrieval and classification for general sounds.

Table 1. Accuracy of different segmentation methods.

Group	Average accuracy
2-phase Euclidean	91.9%
2-phase Cosine	89.1%
Euclidean	89.6%
Cosine distance	87.6%
Silence detection	81.0%

6. REFERENCES

[1] G. Chen, H. Tan, and X. Chen, "Audio Segmentation via the Similarity Measure of Audio Feature Vectors", Wuhan University Journal of Natural Sciences, Vol. 10, No. 5, pp.833-837, 2005.

[2] C. Panagiotakis, and G. Tziritas, "A Speech/Music Discriminator Based on RMS and Zero-Crossings", IEEE Transactions on Multimedia, pp. 155-166, 2005.

[3] G. Tzanetakis, and P. Cook, "A Framework for Audio Analysis Based on Classification and Temporal Segmentation", EUROMICRO, pp. 2061- 2067, 1999.

[4] M.S. Spina, and V.W. Zue, "Automatic Transcription of General Audio Data: Preliminary Analyses", pp. 594-597, ICSLP 96, 1996.

[5] H. Aronowitz, "Segmental Modeling for Audio Segmentation", Acoustics, Speech and Signal Processing, pp. 393-396, 2007.

[6] B. Ramabhadran, J. Huang, U. Chaudhari, G.Iyengar, and H.J. Nock, "Impact of Audio Segmentation and Segment Clustering on Automated Transcription Accuracy of Large Spoken Archives", Proc. EuroSpeech, pp. 2589-2592, 2003.

[7] S.S. Chen, and P.S. Gopalakrishnan, "Speaker Environment and Channel Change Detection and Clustering via The Bayesian Information Criterion", in *DARPA* speech recognition workshop, pp. 127-132, 1998.

[8] H. Meinedo, and J. Neto, "A Stream-based Audio Segmentation, Classification and Clustering Pre-processing System for Broadcast News using ANN Models", INTERSPEECH, pp. 237-240, 2005.

[9] M.K. Omar, U. Chaudhari, and G. Ramaswamy, "Blind Change Detection for Audio Segmentation", Acoustics, Speech, and Signal Processing, pp. 501-504, 2005.

[10] A. Ganapathiraju, L. Webster, J. Trimble, K. Bsush, and P.Kornman, "Comparison of Energy-based Endpoint Detectors for Speech Signal Processing", Southeastcon, pp. 500-503, 1996.

[11] J. Foote, "Visualizing Music and Audio Using Self-Similarity", *ACM Multimedia*, pp. 77–84, 1999.

[12] M. Cooper, and J. Foote, "Automatic Music Summarization via Similarity Analysis", Proc. IRCAM, pp. 81-85, 2002.

[13] S.Z. Li, "Content-based audio classification and retrieval using the nearest feature line method", IEEE Trans. Speech and Audio Proc., pp. 619-625, 2000.

[14] R. Duda, and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons Inc, 1973.

AUDIO CLASSIFICATION BASED ON ADAPTIVE PARTITIONING

Jessie Xin Zhang[1], *Stephen Brooks*[2], *Jacqueline L. Whalley*[1],

[1] School of Computing and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand {jessie.zhang, jwhalley,@aut.ac.nz}

[2] Faculty of Computer Science, Dalhousie University, 6050 University Avenvue, Halifax, Nova Scotia, Canada B3H 1W5 {sbrooks@cs.dal.ca}

ABSTRACT

This paper presents an audio classification system that provides improved accuracy, robustness and flexibility over reported content-based audio classification methods. The system reads an input audio file, performs segmentation and classification of the composite sounds contained within the file and, for each sound clip, determines the most plausible matching class of audio in the database. Improvements in the accuracy of audio classification are largely due to the partitioning of the input audio file into homogeneous segments while the incorporation of new class detection offers greater flexibility of use.

Index Terms— Audio segmentation, classification, new class detection

1. INTRODUCTION

Digital audio files are widely used in a variety of fields including film, television, computer gaming, radio, website design and audio book production. Extensive databases have been developed to store these digital files for professional use and allow the retrieval of desired sound files through some defined interface. Effectively managing the numerous records in an audio database is challenging and relies on the accurate classification and retrieval of audio files. Traditional text-based querying is not feasible for audio management because an audio file is usually treated as an opaque collection of bytes with primitive fields attached [1]. Furthermore, the subjective nature of assigned keywords and the tedium of manual annotation motivate researchers to find more efficient and automated methods.

Providing a given audio file, as an example, to find similar audio files in a database is one way of performing an audio query, based on similarity analysis. Although many approaches have been developed in this field, existing research is based on the use of rich training datasets where it is unlikely for a query candidate to fall outside the ontology of the database. Dealing with a situation where the input sound is very different from those in the training dataset remains an unsolved problem.

2. RELATED WORK

Automatic classification offers a way to efficiently navigate and provide control for search and retrieval within audio databases. This task can be understood as a process where a previously unknown input signal is assigned to a class $C \in \{C_1, C_2, ..., C_n\}$ in an audio database. Such an assignment is made by establishing and applying a decision rule over a designed feature space producing a class (or an audio file in a class) that is deemed closest to the new audio [2].

A number of studies have attempted to classify audio files [4] [5]. Comparative studies of the accuracy of various classification methods have also been undertaken [5]. The most accurate method known for content-based audio classification and retrieval uses various perceptual sound features to form a feature vector for each audio file. This method has been reported to give a 90.28% accuracy using a Nearest Feature Line (NFL) classifier, based on the PercCeps8 feature set and the MuscleFish database [3].

The feature line in the NFL algorithm is a linear interpolation and extrapolation between any two prototype points within a class. The distances between the query sample and all the feature lines are calculated and the query sample belongs to the class of the nearest feature line.

Existing applications for audio classification can only determine the most likely class in the database the query candidate may belong to. They are limited because they can not process audio files that do not belong in an existing class. Instead they classify the file incorrectly into the most similar class. Our work addresses this limitation by introducing a novel 'new class detection' method. This method could be incorporated into other audio systems to prevent misclassification of sounds.

Additionally, to improve the efficiency of the audio classification process, a novel pre-feature extraction, pre-classification, segmentation module is integrated into our system. Our segmentation process differs from approaches

used in existing systems which are limited to broad classes (e.g. speech/music/ silence [6]) and use segmentation as a post-process to improve the classification results [7] [8].

3. THE ADAPTIVE AUDIO CLASSIFICATION SYSTEM

Fig. 1 shows the framework for our audio classification system. The segmentation stage (Fig. 1) takes a single audio file, that may contain one or more sounds, and separates the input audio into clips (segments) where each clip contains a single sound. Subsequently, a set of audio features are extracted from each audio clip. These extracted features are used to determine if the audio file can be classified into the existing ontology or not.

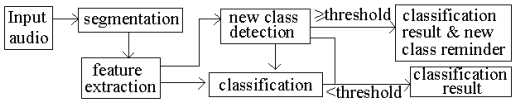

Fig. 1. Frame work for audio classification system.

If the file cannot be classified, a new class is required in the database, and our system automatically builds a new class for the input, prompting the user to name the new class. An 'uncertain' result means the system leaves the classification decision to the user. If the user classifies this file to an existing class the file is still marked as an 'uncertain'. When there are more audio files marked as an 'uncertain', users can make further decisions about whether new classes should be built for these uncertain audio files.

3.1. Pre-Processing modules

The segmentation module in our system aims to separate heterogeneous input audio into acoustically homogeneous regions. Initial segmentation is achieved in the time domain based on silence detection. Then further segmentation is undertaken using similarity autocorrelation of the frames to detect abrupt changes in acoustic audio features. Details can be found in [11].

The subsequent feature extraction module selects the audio feature set that will be used to determine which class an audio file is most similar to. The best feature set may vary according to the training set. We have adapted the best audio feature set as reported by Li [3] for computing efficiency. Our audio feature set is composed of the mean and standard deviation of the Total Spectrum Power, Subband Power, Brightness, Bandwidth, Pitch and ZCR; as well as X order Mel Frequency Cepstral Coefficients (MFCCs) with X=5, 8, 10 and 15. Details on the computation of these features can be found in [3] and [12]. Pitched ratio and Silence ratio are two features used in [3] but not employed here because after segmentation, these two features always have the value 1 (all frames are signal frames).

For the benchmark testing of our system we adopted the MuscleFish database which contains 16 classes, 410 audio files and has been widely used as a standard non-musical sound database for the development of audio matching and classification algorithms [1] [3]. Table 1 shows that for a leave-one-out test, based on the MuscleFish database as a training set, our system of segmentation and classification achieves the best accuracy using the feature set PercCeps8+ZCR (without pitched ratio and silence ratio) at 90.2%.

Table 1. Number of incorrect classifications for 32 audio feature sets using 410 audio files.

Perc			ZCR	CepsX			
Pitch	Pitch/Silence ratio			X=5	X=8	X=10	X=15
√	√	√		54	47	46	53
√	√	√	√	49	47	44	53
	√	√		50	44	42	48
	√	√	√	50	42	43	44
√				47	43	46	51
√			√	46	**40**	46	54
				47	46	44	48
			√	48	44	45	49

3.3. New class detection

Existing audio classification systems assume high quality training sets that map cleanly to the target audio provided by the user. However, for more general use, the system should be able to process input sounds that are very different from those in the training set. The New Class Detection module addresses this limitation, by flagging a particular sound input as potentially belonging to a new class.

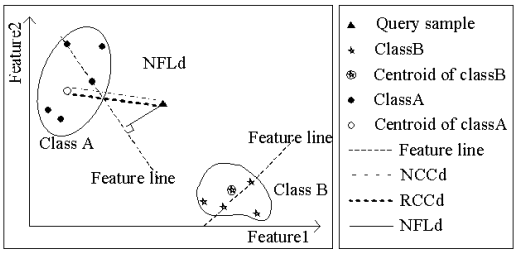

Fig. 2. Parameters used in 'new class definition' experiments.

Because the PercCeps8+ZCR feature set offers the best classification results this feature set is used for the calculation of the new class detection parameters. The criterion for detecting a new class was tested using 4 parameters (Fig. 2):

- NFL distance (***NFLd***), Euclidean distance from the query sample to its NFL.

- Real Center Class distance (***RCCd***), Euclidean distance between the query sample and the class centroid.
- Normalized Centroid Class distance (***NCCd***), with normalization carried out in the same manner as for PercCepsX [3].
- Class Feature Range Number (***Fnum***). When we test Audio *a* and determine its class A, regardless of the accuracy of the result, we find the maximum values and minimum values of each audio feature in the class. ***Fnum*** is defined as how many features of *a* are in the feature ranges of that class.

If an audio file $a \in A$ but is incorrectly classified as $a \in B$ then the parameters for "new class detection" are extracted from *B*. These parameters can be used individually or as a parameter set (Table 2), depending on the training set.

Table 2. Number of files incorrectly classified for the LOFO and LOCO experiments using various parameter sets.

Exp #	Parameter set				LOFO n=370	LOCO n=410	Total n = 780
	NFL*d*	NCC*d*	RCC*d*	Fnum			
1	√				132	63	195
2		√			223	69	292
3			√		283	34	317
4				√	111	101	212
5	√	√			131	63	194
6	√	√		√	131	63	194

In order to evaluate the effectiveness of our new class detection criterion two leave-one-out experiments were developed. The two experiments 'Leave-One-File-Out' (LOFO) and 'Leave-One-Class-Out' (LOCO) are detailed in Fig. 3.

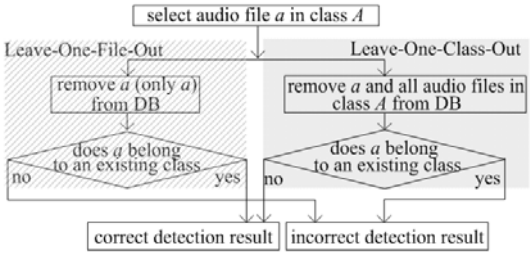

Fig. 3. LOFO and LOCO experiments.

In LOFO, when a sound is used as the query, it is not used as a prototype so the prototype set consists of the entire database minus the query. 370 audio files were tested in the LOFO experiment. These files are those that can be correctly classified into an existing class when classifying the file without new class detection (see Table 1). If one of these audio files is classified as a 'new class', it is incorrect because we know that it actually belongs to an existing class in the database. In LOCO, the prototype set only includes the files that do not belong to the class of the query candidate. In this case if the query candidate is classified as belonging to an existing class in the database, the classification result is incorrect. All 410 audio files in the MuscleFish were tested in the LOCO experiment.

The parameter set and threshold for each parameter were also determined during the LOFO and LOCO experiments. The values for the 4 parameters (*NFLd*, *RCCd*, *NCCd*, and *Fnum*) were calculated for each query candidate. For *NFLd*, *RCCd* and *NCCd* if the result values were less than a threshold, the query candidate had a higher likelihood of belonging to the calculated class. And if the value of *Fnum* was greater than its threshold, the query candidate was close to the result class. So we define that for a query candidate *a*, if any of the parameters in a parameter set does not pass the threshold test, *a* is classified as a candidate for a 'new class'. Because the correct classification and new class definition results are known for these files, the best parameter set and thresholds are the combination where it achieves the most accurate new class detection result. The 3 steps for threshold determination, employing the database as a training set, are as follows:

1. Calculate parameter vectors
2. Determine the threshold range for each parameter.
3. Calculate the best threshold value

For the LOFO and LOCO experiments four parameters are extracted (*NFLd*, *RCCd*, *NCCd*, and *Fnum*). For each parameter *i* a vector v_i is calculated. The threshold for parameter *i* must be greater than the minimum value and less than the maximum value (the range of valid threshold values) for the vector v_i. This range is divided equally into 100 possible threshold values. Classification (with new class detection) is then undertaken using LOFO and LOCO using this set of 100 thresholds. The threshold that results in the lowest error ratio is then used as the threshold for parameter *i* for the database. The result of classification based on this automatic parameter threshold determination is given in Table 2. We conclude from these threshold determination experiments that the most effective parameter of the four is *NFLd* with a comparable accuracy of parameter set *NFLd* and *NCCd*. Additionally, *Fnum* offers no benefit.

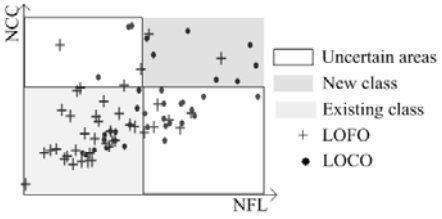

Fig. 4. Audio file classification using 'uncertain criterion'.

In previous experiments, we separated the audio files into 2 categories: the files were either classified into an existing class in the database or labeled as a new class. For an audio file, if all of its parameters are consistent it is easy to make a definitive classification. However, some audio

files have conflicting parameter values. One parameter may indicate that the file belongs to an existing class, but another may indicate a new class is required. These audio files are hard to classify and so an 'uncertain' criterion was introduced. Figure 4 shows the results of testing audio classification using the uncertain criterion and Table 3 provides the results of testing the accuracy of audio classification using the uncertain criterion.

Following the results of Table 2, *NCCd* and *NFLd* form the parameter set. As shown in Fig. 4, for a query candidate, when both parameters are less than corresponding threshold, it is classified to an existing class (the lower left quadrant area). When both the parameters are greater than thresholds (the upper right quadrant), the query candidate belongs to a new class. The rest of the query candidates, which have one parameter greater than threshold and another less than threshold, are marked as 'uncertain'.

Reducing the number of incorrect files can lead to an increase in the number of 'uncertain' files. The threshold is therefore optimized to minimize the number of incorrect files while simultaneously maximizing the number of correct files. With the optimized thresholds, we tested 370 audio files (which are correctly classified without new class detection) in LOFO and all the 410 audio files in LOCO (Table 3).

Table 3. Results for LOFO and LOCO using 'uncertain' criterion.

Incorrect classified (40) files	LOFO exp.	Existing		8
		New		22
		Uncertain		10
Correct classified (370) files	LOFO exp.	Existing	Correct	131
		New	Incorrect	117
		Uncertain		122
410 audio files	LOCO exp.	Existing	Incorrect	40
		New	Correct	318
		Uncertain		52

In the two experiments, the number of incorrectly classified audio files is 157 (117 in LOFO and 40 in LOCO) of 780 query candidates. It shows that our system can deal with audio files that do not belong to existing classes of the training set, and the correct classification ratio with new class detection is ≈ 80%. This is a significant result as it is comparable with the results of classification accuracy without new class detection using other classifiers (such as 5-NN) reported in [3]. Furthermore, the system allows users to fine-tune any threshold to obtain desired results. For example, by increasing the thresholds to maximize the existing class region.

Moreover, the 40 incorrectly classified audio files were also tested (top row, Table 3). Only 8 files of the 40 are classified into an existing class directly. The system prompts the users to consider classifying them manually. Our system reduces the risk of misclassification. In a system without new class detection, for example [3], a LOCO experiment would result in all 410 files being incorrectly classified. In our system only (40/410) 10% are incorrectly classified.

Future work includes investigating alternative parameters for new class detection, such as the distance from the test sample to the closest prototype.

5. CONCLUSION

Our system is a powerful tool for audio segmentation, classification and new class prediction. We achieve the same accuracy as Li [3] using our system. Integrating a pre-classification segmentation enables the system to process not only single sounds, as in previous systems, but also multiple-sound audio files. In addition, the system explores a novel approach to the detection of new classes from the input query samples during classification. The system is geared to predict whether a query sample belongs to an existing class, or a new class. Those query samples that potentially belong to a new class are marked as 'uncertain'. They can be re-evaluated, with user input, when additional 'uncertain' audio files appear, and subsequently clustered to new classes. In general, by incorporating new class detection and segmentation, our system breaks through the limitations of current classification systems, offers greater flexibility and robustness in general audio classification.

6. REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-Based Classification, Search, and Retrieval of Audio", IEEE Multimedia, Vol. 3, No. 3, pp. 27-36, 1996.

[2] S. Theodoridis, and K. Koutroumbas, *Pattern Recognition*, Third Edition, Academic Press, 2006.

[3] S.Z. Li, "Content-based audio classification and retrieval using the nearest feature line method", IEEE Trans. Speech and Audio Proc., pp. 619-625, 2000.

[4] M.I. Mandel, G.E. Poliner, D.P.W. Ellis, "Support vector machine active learning for music retrieval", Multimedia Systems, Vol. 12, No. 1, pp. 3-13, 2006.

[5] G. Guo, S.Z. Li, "Content-based audio classification and retrieval by support vector machines", IEEE Trans. Neural Networks, Vol. 14, I(1), pp. 209-215, 2003.

[6] A. Omar, "Audio segmentation and classification", Masters Thesis, Technical University of Denmark, 2005.

[7] D. Kimber and L. Wilcox, "*Acoustic Segmentation for Audio Browsers*", Proc. Interface Conference, pp. 60-68, 1996.

[8] L. Lu, H. Jiang and H. Zhang, "A Robust Audio Classification and Segmentation Method", ACM Multimedia, pp 203-211, 2001.

[9] J. Foote, "Visualizing music and audio using self-similarity", Proc. 7th ACM Multimedia, pp. 77-80, 1999.

[10] R. Duda, and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons Inc, 1973.

[11] J.Zhang, J. Whalley, and S. Brooks, "Time Mosaics – An image processing approach to audio visualization", pp. 273-280, DAFx_08, 2008.

[12] C.H. Chen, *Signal processing handbook*, CRC Press, New York, 1988.