Glyndŵr University
Glyndŵr University Research Online

Computing | Computer Science

1-1-2007

Advances in Similarity-Based Audio Compression

Stuart Cunningham
Glyndwr University, s.cunningham@glyndwr.ac.uk

Vic Grout
Glyndwr University, v.grout@glyndwr.ac.uk

Follow this and additional works at: http://epubs.glyndwr.ac.uk/cair

Part of the Computer and Systems Architecture Commons, Digital Communications and Networking Commons, Hardware Systems Commons, and the Systems and Communications Commons

Recommended Citation
Cunningham, S. & Grout, V. (2007), 'Advances in Similarity-Based Audio Compression'. [Paper presented to the Third Collaborative Research Symposium on Security, E-Learning, Internet and Networking 14th-15th June 2007]. Plymouth: Plymouth University

This Conference Paper is brought to you for free and open access by the Computer Science at Glyndŵr University Research Online. It has been accepted for inclusion in Computing by an authorized administrator of Glyndŵr University Research Online. For more information, please contact d.jepson@glyndwr.ac.uk.

Advances in Similarity-Based Audio Compression

Abstract
Existing lossy audio compression techniques such as MP3, WMA and Ogg Vorbis, for example, demonstrate great success in providing compression ratios which successfully reduce the data size from the original sampled audio. These techniques employ psychoacoustic models and traditional statistical coding techniques to achieve data reduction. However, these methods do not take into account the perceived content of the audio, which is often particularly relevant in musical audio. In this paper, we present our research and development work completed to date, in producing a system for audio analysis, which will consider and exploit the repetitive nature of audio and the similarities which frequently occur in audio recordings. We demonstrate the feasibility and scope of the analysis system and consider the techniques and challenges that are employed to achieve data reduction.

Keywords
Compression, Audio, Music, Similarity, Repetition, Psychoacoustics

Disciplines
Computer and Systems Architecture | Digital Communications and Networking | Hardware Systems | Systems and Communications

Comments
This paper was presented at the Third Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN 2007), 3rd International NRG Research Symposium, 14-15 June 2007, which was held by University of Plymouth and the symposium proceedings are available at http://www.cscan.org

This conference paper is available at Glyndŵr University Research Online: http://epubs.glyndwr.ac.uk/cair/54

Advances in Similarity-Based Audio Compression

Stuart Cunningham and Vic Grout

Centre for Applied Internet Research (CAIR), University of Wales, NEWI, Wrexham, North Wales, United Kingdom
s.cunningham@newi.ac.uk | v.grout@newi.ac.uk

Abstract

Existing lossy audio compression techniques such as MP3, WMA and Ogg Vorbis, for example, demonstrate great success in providing compression ratios which successfully reduce the data size from the original sampled audio. These techniques employ psychoacoustic models and traditional statistical coding techniques to achieve data reduction. However, these methods do not take into account the perceived content of the audio, which is often particularly relevant in musical audio. In this paper, we present our research and development work completed to date, in producing a system for audio analysis, which will consider and exploit the repetitive nature of audio and the similarities which frequently occur in audio recordings. We demonstrate the feasibility and scope of the analysis system and consider the techniques and challenges that are employed to achieve data reduction.

Keywords

Compression, Audio, Music, Similarity, Repetition, Psychoacoustics

1. Introduction

Audio compression is a hugely challenging and rewarding area of research, which in the 21st century, has the potential to affect the technologies, lifestyle and standard of life of millions of users worldwide. If proof were needed of the significance of the area of audio compression as a social, as well as research driven, field, one only needs to consider the extensive media coverage and the impact of technologies such as the Apple iPod, MP3 files and the legal and economic issues surrounding the downloading of music through services such as Napster and iTunes.

Audio, particularly music, files tend to be large, however, which can cause problems in transmission across network bottleneck points. The effects are usually data loss or unacceptable latency or jitter. Continual efforts are being made to produce compression techniques for audio which maintain the highest audio quality, or at least *perceptually* high-quality, whilst also achieving high levels of data reduction. The perfect compromise would produce a minimal data representation that produced an inaudible difference, if any, in audio quality to the original sampled waveform. The ideal scenario would be to retain the original CD quality audio. However, it is widely accepted that this format creates large amounts of data (Rumsey, 1996). Without altering the quality, it would be necessary to exploit redundancy in audio data to reduce the amount of data required to store audio of such quality. In effect, this would be a system of lossless compression.

Current compression techniques for audio, although highly successful, both in the user community and from a data reduction perspective, do not consider some fundamental aspects of the content of the audio they process. The main issue identified with these techniques is that they always encode a revised version of the original waveform in its entirety, and do not take into account any musical or other form of excessive repetition, such as extended periods of silence.

The methods we are developing and discuss in this paper are not lossless, but lossy, although we rely on the presence of information in audio which is *perceptually redundant*, and is therefore more rightly termed *irrelevant.* We aim to find this unnecessary information by considering audio, and in particular music, from a top-level, and consider the structure and make-up of audio. Musical audio, in particular, contains inherent amounts of repetition. This could be repeating choruses or verses or simply repeating a small sequence of notes. In particular many musical genres and styles, such as dance, techno, electronic, and hip-hop, rely heavily on frequent repetition of beats, hooks, riffs, and vocal catches.

This data reduction process will work by analysing existing information within a waveform, repeating sections of it as required, and disposing of the perceptually identical replicas contained in the file. By doing so, the data is repeated or looped rather than storing two versions of the same thing. A simple example of repetition in music would be to consider a piece of music which displays repetitive structure and sequences when examined from a macroscopic level. Audio blocks could then be repeated where necessary, rather than being coded multiple times. Such an example is illustrated in Figure 1.

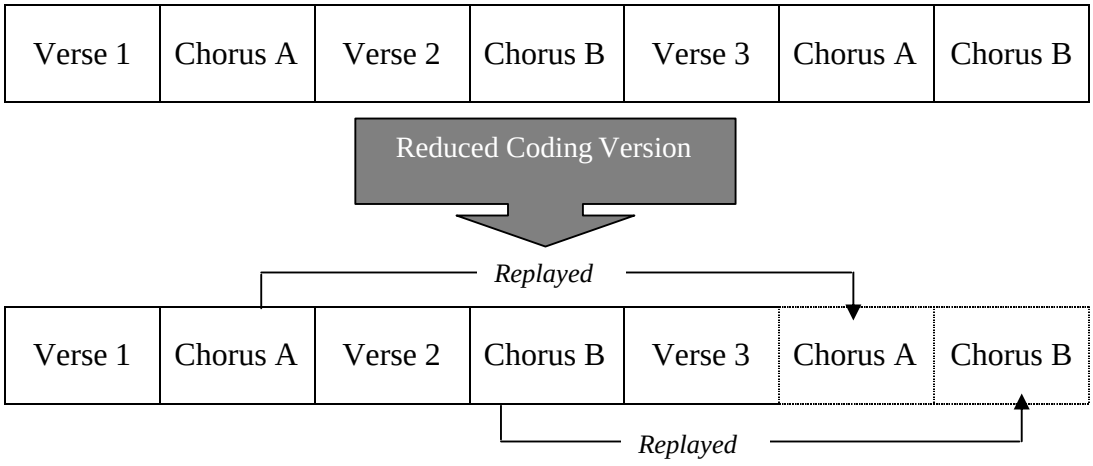

Figure 1: Top-Level Example of Musical Structure

In this simple example, it is clear that, provided the Choruses A are similar and the Choruses B are similar, or at least perceptually similar, it is only necessary to encode and store one version and repeat that sequence at the appropriate point later in the song. In this example, this would instantly present a saving of approximately 29% over the original piece of music through data redundancy (if samples were identical) or data irrelevancy (if the difference cannot be perceived). It is also important to recognise that such similarity will exist in audio, not only on such a macroscopic level, but also on a much smaller scale, such as individual musical notes or sections of audio which may be less than a second long.

This can be demonstrated more specifically for audio, this time on a much smaller scale, by considering the images in Figures 2 and 3, where we show the contents of two wave samples from the song, *Another Brick in the Wall (part 2)* by Pink Floyd, by graphing the sample values. Each sample window is approximately 2.4 seconds long and has been extracted from two separate time points within the overall song.

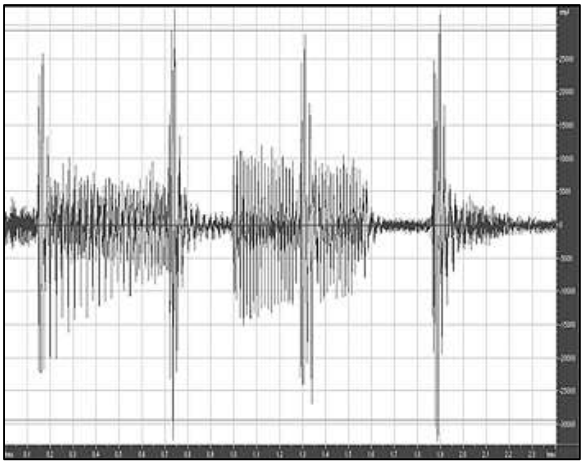

Figure 2: Audio Sample at 0m 14s

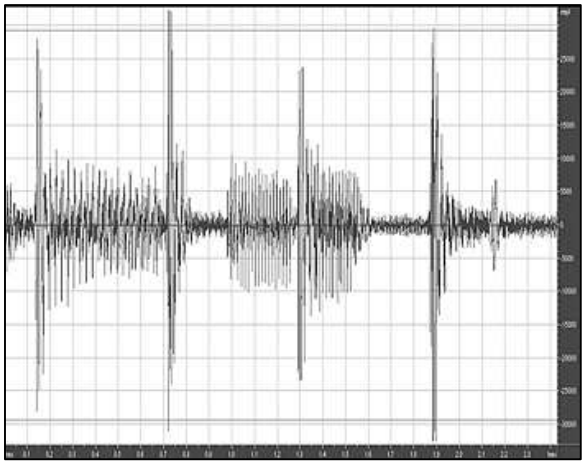

Figure 3: Audio Sample at 1m 27s

From observation of the two waveforms shown in Figures 2 and 3, it is easy to ascertain that there are distinct similarities between the two samples presented. Indeed, if one listens to each sample they are perceptually very similar and each represents part of sequences which are frequently repeated throughout the song. Though initial inspection suggests that the images are identical, closer attention to detail, and the contours of the waveform graph, reveals that there are some subtle differences. In practice, this would be noticed by listening more closely or by listening a number of times when comparing the samples. However, this does not remove the notion that, although they may not be identical, these samples are very similar. To demonstrate this slightly more scientifically, we could consider analysing the sample values contained in each of the sections. However, since each sound file, in this case, contains 19,236 samples, it is easier, for the purposes of this paper, to note the fact that the samples in Figures 2 and 3 have different mean and standard deviation values.

This provides a suitable insight into the challenges and scope of the work required to achieve such a system for audio compression. Clearly, the main points to be considered are the procedures and processes needed to comprehensively search for repetition and structure in audio as well as the investigation into the best techniques to identify suitable, perceptually similar, matching audio blocks.

2. Previous and Related Work

There has been a wealth of work carried out, especially in the field of musicology, which concentrates on analysing and determining repetition, structure and patterns within music. However, to date, the majority of this research has focussed on using musical notation or similar descriptive musical information, such as MusicXML, MIDI (Musical Instrument Digital Interface), or similar formats, as the data for analysis, not the actual waveform audio representation of the music. However, it is worthwhile presenting a brief overview of some of this work, as it further demonstrates the value of investigation of musical content and

repetition and provides indications of techniques which could be applied to a waveform-based analysis system.

Dannenberg & Hu (2002_a & 2002_b) have shown positive results in detecting similar sequences of melody in monophonic music and have experimented in the polyphonic domain, primarily using pitch detection and chroma quantisation. Another technique considered by Mazzoni and Dannenberg (2001) use a pitch contour matching technique against a database of modified MIDI songs, which had assigned pitch contours, alleviating problems introduced by quantisation. However promising, these methods rely on pitch detection, normally based on frequency analysis, or using musical notation and MIDI. Although their systems provide indicative evidence of repetition, they lack the granularity that would be necessary to achieve acceptable compression of the kind we propose in this paper. In the system proposed in this paper, it is necessary to account for not only the pitch, but the entire polyphonic spread of sounds. After all, though the proposed system is intended to be used in music compression, there is no reason why it should be in any way restricted in its use in the audio domain.

Chai (2003) and Chai and Vercoe (2003) present work more considerate towards polyphonic audio. Their research demonstrates excellent and highly-positive results in detection of musical structure. The defining differences between the system proposed here and that of Chai is that their system tends to look for high level music structure (e.g. ABABA, where A an B are typically verses or choruses) whereas our system will detect structure at as small a level as the user wishes to define. The other main difference is that Chai uses single-channel audio, low sample rates and low bit-depth. Our system aspires to operate at any sample rate, bit-depth, and across multiple channels, provided the search algorithms are scaled correctly.

Peeters *et al*. (2002) have also considered the issue of detecting patterns and structure in polyphonic music using signal analysis, for the purposes of music summarisation and employ a similarity matrix approach which can be visually depicted. Their work also presents the results of such analysis, and the visual indicators for several very successful tests across a varied genre of musical compositions. Particularly, their system is focused on genres of music where repetition plays a major role in the piece.

Of significant note is the work of Kirovski and Landau (2004 and 2005). They employ psychoacoustic, audio analysis techniques, based upon the principles of the Lempel-Ziv algorithm, to detect the existence of suitable redundant data to allow entropy coding. In their work, it is useful to note the functions used to analyse the audio, which, in this case, are mainly those of the frequency analysis transform, the Modified Discrete Cosine Transform (MDCT). As is expected from our own investigations, the results are variable depending on the musical material being processed, but reductions in excess of 90% have been demonstrated for electronic musical pieces that contain high-levels of repetition. Kirovski and Landau's work demonstrates further the worth and concept of our own investigations and development efforts and also provides material against which to benchmark our own techniques.

In other work by the authors of this paper, we also chose to examine the musical structure and sequences present in music (Cunningham_a *et al*. 2005). However, rather than looking for similarity of sequences and structure within a single piece of music, we searched for similarity between perceptibly different genres of music. For example, we compared music from the heavy metal genre to classical music. This work was useful as it helped gain more

understanding of the logical and procedural approaches needed when analysing musical content, although it too relied on musical notation as the source for analysis.

As part of the initial developments in this area of research, a framework for the audio compression technique was devised. This provides a useful overview of how the proposed techniques will be perceived from the top-level or user perspective. This is presented in Figure 2.

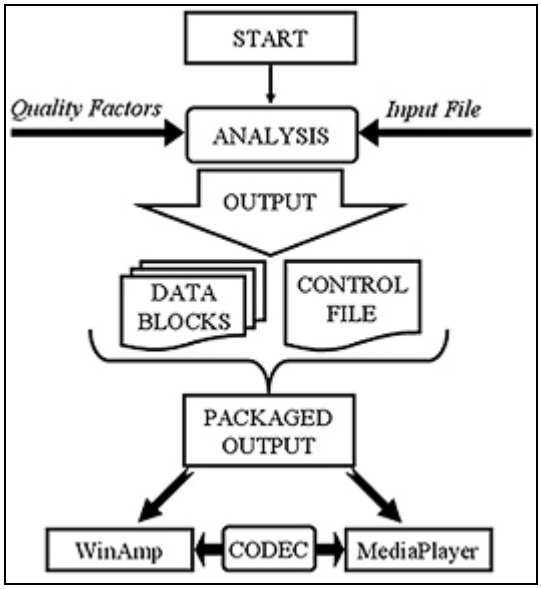

Figure 2: Overview of Compression System (Cunningham$_b$ *et al.* 2005)

As part of the initial research, one of the main areas of investigation was into the feasibility of reusing audio segments from the point of view of the audio or music listener. A series of small tests were devised to attempt to gauge the extent to which a listener was able to differentiate between similar audio samples. To attempt to engage users in their usual listening environments, this test was placed online and users were able to take the tests at their convenience. Further testing in controlled environments will be heavily employed later in the development but at this early investigative stage, this was an appropriate method (Cunningham, 2005).

The test was divided into two sections. The first required the subjects to listen to five pairs of sounds and state whether or not they thought they were different. The listener was only permitted to hear each pair of sounds once. In the second section, the same sounds were used but listeners were free to listen to each pair of sounds as many times as they wished. The tests consisted of audio samples of pieces of music and others of single instruments or sounds. The pairs of sounds were all different in some way (with the exception of the sounds in question 3, which were the same - a 'placebo'), either by extracting similar, but not identical, samples from a piece of audio, or by taking identical pairs and processing or filtering one of them. The samples used were constructed to range across the spectrum of being obviously different to more subtle. The results of these tests are shown in Figures 3 and 4.

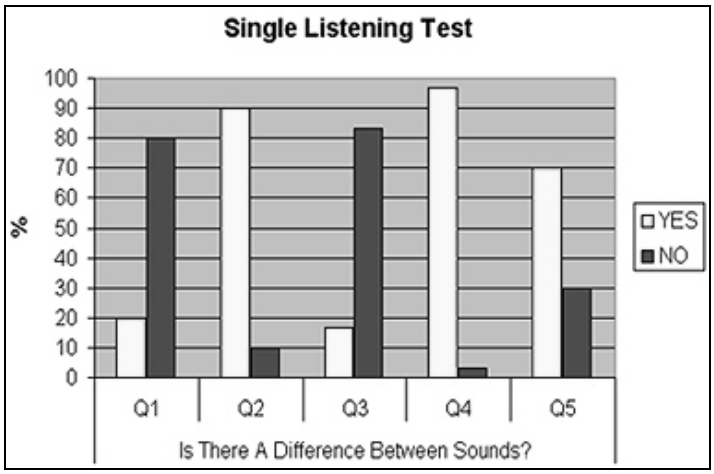

Figure 3: Single Listen Audio Comparison Test

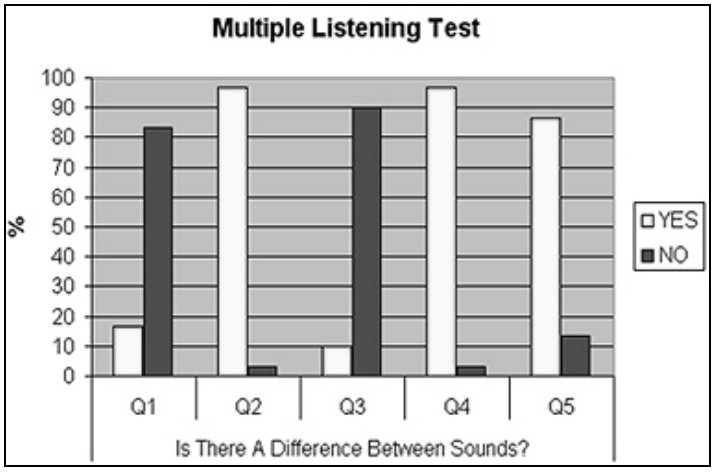

Figure 4: Multiple Listen Audio Comparison Test

Overall, the results were generally as expected and indicate that listeners had difficulty determining well constructed, repetitive audio as distinct segments. The results also reveal that pieces of audio which were different, but had many qualities in common, could also require more careful listening to determine the difference. The outcomes from question 3 are useful as they allow us to consider the inclusion of an error or deviation factor into the results, which in this case would be an average of ±13.4%. Another interesting feature emerged from considering how quickly listeners made their decision about the similarity between two sounds. In section two, where subjects were permitted to listen as many times as they wished, the average number of plays of each pair of sounds was smaller than expected - an overall average of 1.62. This indicates that in such scenarios listeners quickly assess the content and importance of the audio being presented and make a firm assessment of it. This is also a well-founded assumption as we compare results between sections one and two of the test. We see that similar trends are presented and the average difference between responses in sections one and two is only 6.7%.

Hacker (2000) defines a set of criteria, which any compression format hoping to compete with, or supersede, the MP3 format would have to demonstrate. The criteria Hacker defines, which would be required of a compression format to surpass MP3, are as follows:

- Smaller file sizes
- Superior audio quality
- Free and unprotected format

Although it could be argued that an improvement in any of these areas, with no corresponding degradation in the other two, could still constitute progress, it is worthwhile considering and reflecting upon these factors, either individually or in combination, when considering the scope and aims of this work.

3. Comprehensive Similarity Searching

To start, it is necessary to establish a suitable searching system to analyse audio files. We consider such factors as the number of steps or complexity as well as the practicalities and implications of carrying out this highly linear process. This produces mechanisms which are essentially brute-force attempts to ensure that every possibility of finding a match is procedurally exposed. Where we employ an actual block comparison function in the tests performed in this section, we rely on comparing a search and target block using an exact array matching technique. This is the most logical (although least likely to yield the best results) method to begin the investigation, particularly when we are at the stage of developing and refining the comprehensive search procedures.

3.1 Static Block Searches

With this technique, the block size is static throughout the search procedure and is defined by the user before execution. The target block is incrementally passed over the length of the file until one block's length from the end. The same motion is applied to the target block. This has the effect of providing the most comprehensive coverage possible of the data in the file.

Let *f* represent the length of the file and *b* the size of the search and target blocks. Assuming that the stepped value for the search and target block offset increments is always 1, the block size is static over the course of this search and that the cost of the matching process can be defined generically as some function, *X(b)*, then the complexity for the search is defined as

$$C(f) = \sum_{g=0}^{f-b} \sum_{i=0}^{f-b} X(b). \tag{1}$$

To be more precise, it is also useful to include the steps required in the iterative process of reading the byte data into the target and search blocks. Since most audio files have large sample rates, even a block consisting of a tenth of a second's worth of audio can generate a large number of data bytes. This gives a revised form of

$$C(f) = \sum_{g=0}^{f-b} \sum_{i=0}^{f-b} b^2 X(b). \tag{2}$$

A disadvantage of this particular searching method is its exhaustive nature. In the process of this search, any block which is being searched for in the file will detect itself as a valid match. This is one drawback in functionality of this method and means that the minimum number of

matches in any file will be given by f / b. Another drawback also skews the number of matches due to the retrospective nature of the technique. Since the target block will always look for a match from the start of the file, this means that matches that have been made earlier in the search will be made again. For example, if a block at offset 20 matches a block at offset 60, then block 60 will also match block 20. This provides the result of two matches being found, when we would only consider this to be a single match.

However, although inconvenient when attempting to determine how many true matches will be contained in a file, this also increases the amount of time required to carry out the search. We therefore consider a procedure of searching that does not match a block against itself (self-matching) or retrospectively search within a file (backwards-matching). The offset is defined by the block size, declared at execution. This eliminates the detection of false matches which occur when the search block and target block overlap, resulting in a true match result. This also reduces the overall complexity/number of steps involved in carrying out the search, since the number of steps needed to progress the target block is reduced. The complexity for this search can now be defined as

$$C(f) = \sum_{g=0}^{f-b} \sum_{i=g+b}^{f-b} b^2 X(b). \tag{3}$$

To demonstrate the efficiencies gained by making these simple reductions, it is useful for us to compare the number of steps required between expressions (2) and (3). This is shown in Table 1.

Sample Values	Expression (2)	Expression (3)	Difference	Saving
f=3,000, b=5	224,250,625	111,863,400	112,387,225	49.88%
f=3,000, b=10	894,010,000	444,467,100	449,542,900	49.72%
f=3,000, b=100	84,100,000,000	39,242,010,000	44,857,990,000	46.66%
f=12,000, b=5	3,597,000,625	1,797,450,900	1,799,549,725	49.97%
f=12,000, b=10	14,376,010,000	7,177,817,100	7,198,192,900	49.93%
f=12,000, b=100	1,416,100,000,000	696,377,010,000	719,722,990,000	49.18%

Table 1: Evaluating Expressions (2) and (3)

The results presented in Table 1 clearly show that there is a significant saving made by reducing the number of steps involved in the search and by removing essentially redundant and irrelevant comparisons. Across all tested search parameter values (f and b) we can see that a saving of approximately 50% (average is 49.2%) of the steps required is made. This optimisation is significant.

The cost of the search for (2) can be approximated as On^2, or more specifically by the considering the file and block size values.

$$C(f) \approx fb^2. \tag{4}$$

This is in contrast to the complexity of carrying out the search as evaluated in expression 3, which is approximate to On^2-$n/2$, or more precisely.

$$C(f) \approx \frac{fb^2 - fb}{2}. \tag{5}$$

To translate the notion of the number of steps required to meaningful real-world data, some sample comparison tests were run. The aim here is to measure the actual time involved in carrying out such searches. This is a crucial stage in the testing and development of these numerical matching techniques. In the tests conducted, all of the files used a sample rate of 6 kHz and a bit-depth of 8-bits. The results of these operations are detailed in Table 2.

Sample Values	***f* Length (s)**	**Expression (3)**	**Time (s)**	**Time (m)**
f=3,000, b=5	0.5	111,863,400	50.719	<1
f=3,000, b=10	0.5	444,467,100	39.360	<1
f=3,000, b=100	0.5	39,242,010,000	21.016	<1
f=12,000, b=5	2	1,797,450,900	736.125	~12
f=12,000, b=10	2	7,177,817,100	594.359	~10
f=12,000, b=100	2	696,377,010,000	427.625	~7

Table 2: Complexity Calculations and Actual Running Times

When performing these tests, it must be recognised that, when measuring time, there will be additional costs to carry out the particular matching function *X(b)* with each step. To ensure that this was not dramatically altering the actual time taken, we measured the time costs of performing the numerical block (array) comparisons. This is summarised in Figure 5, which shows that the comparison times are negligible, given the magnitude of the total search complexity from Table 2.

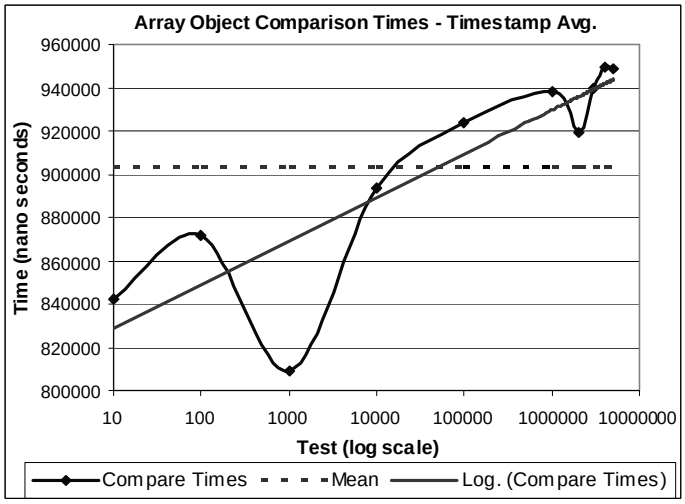

Figure 5: Analysis of Numerical Array Comparison Times

The major limitation with all procedures discussed so far is that of the static search and target blocks. To use a block of static size is extremely limiting and does not provide a complete and comprehensive search for all available matches. Therefore, this must be addressed.

3.2 Dynamic Block Searches

To further refine this search method and ensure that maximum scope is given to detect matching audio sequences, we employ a dynamic search and target block. Given the length of a file *f*, the file would be iteratively searched using a search block *b*, which would increase (or decrease) in size through the lifetime of the search process. Therefore, minimum and maximum sizes of search (and target) block , b_{min} and b_{max}, are established before searching and an increment value, Δb, between these limits of block sizes is also set. (We Assume Δb divides b_{min} and b_{max}.) A search increment across the file - the interval between one search position and the next - is also defined as Δs, and will effectively allow a fineness or depth facility to the function. Similarly, a final incremental parameter, Δt, provides an offset for the target block where the match for b is sought. This is illustrated in Figure 6.

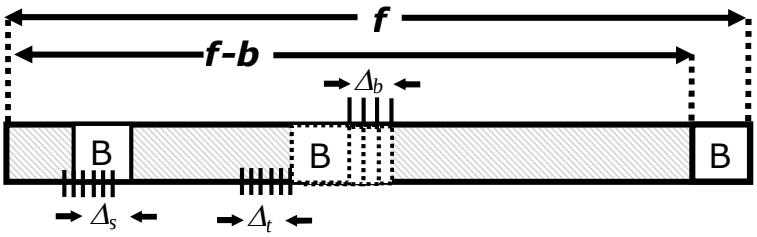

Figure 6: Assignment of Search Components

Again, we generically define the complexity, or number of steps required to match one block of size *b*, as the function *X(b)*. Then for a given set of values of file size, maximum and minimum block size, search block and target block start position increments, the complexity of an exhaustive search routine, considering all possible matches of *b* ($b_{min} < b < b_{max}$) is given by

$$C(f, b_{\min}, b_{\max}, \Delta_b, \Delta_s, \Delta_t) = \sum_{b=\frac{b_{\min}}{\Delta_b}}^{\frac{b_{\max}}{\Delta_b}} \sum_{s=0}^{\frac{f-\Delta_b b}{\Delta_s}} \sum_{t=0}^{\frac{f-\Delta_b b}{\Delta_t}} X(\Delta_b b). \tag{6}$$

Since the length of the file is effectively static at the start of the compression process, the five parameters (b_{min}, b_{max}, Δb , Δs , and Δt) are all, in effect, measures of quality or effectiveness of the search process and could be defined at the beginning of any search. However, the end user of such a system is unlikely to want to specify five different values each time they save a file. It would be much better to give an overall, single 'quality' rating factor to the end user, through which the quality value of the final compressed file could be specified. Internally however, it is expected that the setting of this value would scale the previously defined quality parameters by the required factors to facilitate an overall quality factor.

To provide simplification, these parameters are consolidated into one value: $\Delta = \Delta b = \Delta s = \Delta t$. To provide additional simplification, the assumption is made that $b_{min} = \Delta b = \Delta$ and $b_{max} = f / 2$, since, in most cases, it would be impractical to search for a block greater than half the length of the file. Given these values, the expression in (6) then simplifies to

$$C(f,\Delta) = \sum_{b=1}^{\frac{f}{2\Delta}} \sum_{s=0}^{\frac{f-\Delta b}{\Delta}} \sum_{t=0}^{\frac{f-\Delta b}{\Delta}} X(\Delta b). \tag{7}$$

Finally, if we approximate $X(b)$ by an invariant term X, which is not dependent on b, and assume $f >> b$, so that all $(f- \Delta b)/ \Delta$ terms reduce to f then this reduces the expression to:

$$C(f,\Delta) \approx \frac{f^3 X}{2\Delta^3}. \tag{8}$$

From these expressions, it is clear that performing such a search would be linearly dependant on the cost or complexity of the search itself (X) and the time required for the search process will depend on the particular pattern discovery techniques being employed. Although we use a simple array matching technique in our initial tests, it is far more practical and likely that future methods will involve more audio-specific transforms and functions that will provide much more suitable mechanisms for comparing audio.

4. Audio Analysis Techniques in Similarity Searching

Although array matching is currently used, this has been inefficient in yielding suitable matches, and it is far more appropriate to consider solutions which take into account the spectral audio content of the sound being analysed. For example, if we consider the samples presented in Figures 2 and 3, we see that the wave shapes were similar at a macroscopic level. However, we also notice subtle differences. When considering the audio blocks on a sample-by-sample basis, there will clearly be radical, almost random, differences in a purely numerical sense, although there is potentially scope for simple correlation or range pattern matching functions to be applied. However, even these methods would be further enhanced by some kind of transform being applied.

Frequency/time related transforms, such as the Fast Fourier Transform or Discrete Cosine Transform, are commonly used in signal processing and analysis. The effectiveness of using such techniques is presented simply in Figure 7, which graphs the frequency analysis (acquired using a FFT) of the two samples we believed to be similar in Figures 2 and 3. Notice in particular, the similarity in frequency content and trends between the two samples.

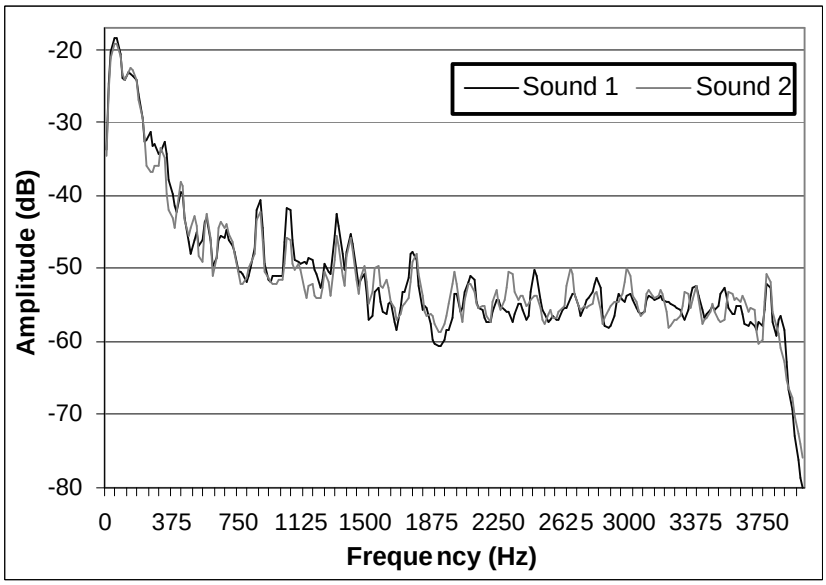

Figure 7: Frequency Analysis of Similar Audio Samples

Working with the data produced by the FFT is now much simpler and faster than searching through large amounts of numerical information. However, we must accept that the FFT applied on these audio samples provides an average of the frequency content within the sample. Therefore, although this might be suitable for samples of short length, it is too inaccurate to be used across large blocks of audio. A more suitable solution is to apply the transform (in this case an FFT) as a sliding window across the audio file. To demonstrate this principle, we apply a Short-Term Fourier Transform (STFT) to the two previous samples to demonstrate the greater granularity and detail allowed. This is shown in Figures 8 and 9.

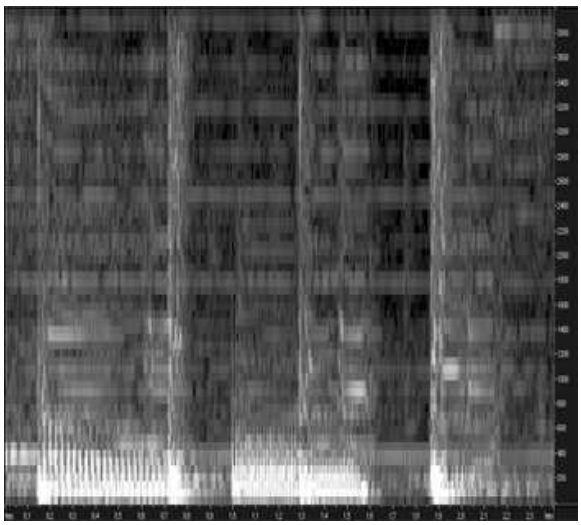

Figure 8: Spectrogram - Sound 1 (0m 14s)

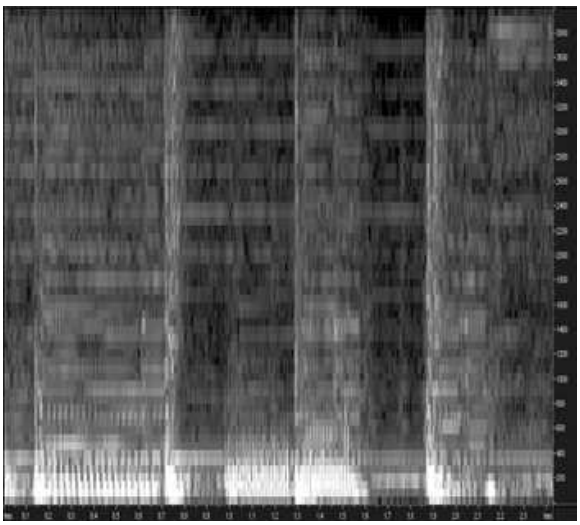

Figure 9: Spectrogram - Sound 2 (1m 27s)

Figures 8 and 9 show the results of the STFT in the form of a spectrogram. The scale uses brightness in the image represents the high power and darkness indicates low power at given frequencies. Again, observing the two images immediately identifies large amounts of similarity, although closer inspection reveals some differences. However, most relevant is

that, with this data, we can identify *where* and to *what extent* there are differences between blocks much more accurately than can be achieved using purely sample values.

5. Conclusions & Future Work

There is clearly a significant amount of work to be carried out in the development of this analysis and compression technique. Key to this is determining the most suitable method(s) for audio comparison, and crucially, those which provide the most acceptable results to a listener. The results of research and investigation so far indicate that this will most likely be a frequency/time based transform, as this is where the bulk of similar audio analysis work has succeeded. However, this should not rule out the opportunity to investigate other methods which may have been overlooked. This presents the next highly significant stage of the research beyond the development of the technique, namely the methods and procedures used to assess and evaluate the suitability of any compression mechanisms developed.

It is expected that testing will initially take a *quantitative* form in order to assess as wide and diverse a range of subjects as possible. This is particularly essential given the potentially vast range of end-users of such a system. This will provide a suitable top-level evaluation in the first instance, to determine the acceptability of the technique. This is likely to be a very iterative process and will be used to test many parameters - some of which have been mentioned previously in this paper - and the effects these have on the resultant audio. A key aspect of this will be to investigate the limits and thresholds of what is acceptable, in particular to what extent the technique can be employed before the quality of the audio is unacceptable. At this stage, more *quantitative* methods may become useful to identify more precisely the traits and aspects of the audio which are particularly of note in determining the effectiveness, and ultimately the quality, of the compressed file.

Key to achieving implementation and practical usage of the techniques described in this work is the optimisation of the search procedures. As we have shown, there us a high level of complexity involved in the linear, exhaustive search methods we have described. However, even from small-scale testing it is apparent that these methods are inappropriate and take excessive lengths of time to execute. Clearly, there is a need to consider *heuristic* and *non-linear* search routines when processing the audio data. For example, we are currently investigating the effects of altering the block sizes used b_{max} and b_{min}, search and target block increment values Δt and Δs, and the increment of the block size Δb. It may also be more reasonable to search for large blocks first and then blocks of decreasing size, or vice-versa. This is a crucial area, which is a priority for current and future investigation.

There are many other potential uses for this system and the resultant data produced. For example, content-based searching is a field where the use of small, searchable, clips of audio could prove tremendously useful, saving search time, and diversifying content. The waveform analysis and comparison techniques could also be used in legal situations, to settle differences in how similar one piece of music was to another.

Another function for this system in copyright might be to detect the use of music samples within audio compositions or collages. Blocks extracted from waveforms could be compiled into a central library, which could aid such applications as audio searching by humming or whistling – again, content-based searches. Such libraries could also be used for composition,

providing a system of 'Object-Oriented Music Composition' for computer music students and musicians alike.

6. References

Chai, W. (2003), *Structural Analysis of Musical Signals via Pattern Matching*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Hong Kong, China.

Chai, W. and Vercoe, B. (2003), *Music Thumbnailing via Structural Analysis*, Proceedings of the 11th ACM International Conference on Multimedia, Berkeley, California, USA.

Cunningham, S. (2005), *Waveform Analysis for High-Quality Loop-Based Audio Distribution*, Proceedings of ISCA 20th International Conference on Computers and their Applications (CATA-2005), 16th – 18th March, New Orleans, Louisiana, USA.

Cunningham$_a$, S., Grout, V. and Bergen, H. (2005), *Mozart to Metallica: A Comparison of Musical Sequences and Similarities*, Proceedings of ISCA 18th International Conference on Computer Applications in Industry and Engineering (CAINE-2005), Hawaii, USA.

Cunningham$_b$, S., Grout, V. and McGinn, J. (2005), *Play it Again, Babbage! – A Framework to Exploit Musical Repetition for High-Quality Audio Compression*, Proceedings of IADIS International Conference on WWW/Internet, 19th – 22nd October, Lisbon, Portugal.

Dannenberg$_a$, R. B. and Hu, N. (2002a), *Pattern Discovery Techniques for Music Audio*, Proceedings of ISMIR 2002 Conference on Music Information Retrieval, IRCAM, Paris, France.

Dannenberg$_b$, R. B. and Hu, N. (2002b), *Discovering Musical Structure in Audio Recordings*, Proceedings of Music and Artificial Intelligence: Second International Conference, ICMAI, Edinburgh, Scotland, UK.

Hacker, S. (2000), *MP3: The Definitive Guide*, O'Reilly, UK.

Kirovski, D. and Landau, Z. (2004), *Generalized Lempel-Ziv Compression for Audio*, IEEE 6th Workshop on Multimedia Signal Processing.

Kirovski, D. and Landau, Z. (2005), *Parameter analysis for the Generalized LZ Compression of Audio*, Proceedings of Data Compression Conference (DCC 2005), 29th – 31st March, Snowbird, UT, USA.

Mazzoni, D. and Dannenberg, R. B. (2001), *Melody Matching Directly from Audio*, Proceedings of ISMIR 2001 Conference on Music Information Retrieval, Indiana, USA.

Peeters, G., La Burthe, A. and Rodet, X. (2002), *Toward Automatic Music Audio Summary Generation from Signal Analysis*, Proceedings of ISMIR 2002 Conference on Music Information Retrieval, IRCAM, Paris, France.

Rumsey, F. (1996), *The Audio Workstation Handbook*, Focal Press, Oxford, UK.