AUDIO COMPRESSION EXPLOITING REPETITION (ACER): CHALLENGES AND SOLUTIONS

Stuart Cunningham and Vic Grout

Centre for Applied Internet Research, Glyndŵr University, Wrexham, UK
{s.cunningham | v.grout}@glyndwr.ac.uk

ABSTRACT

This paper presents issues relating to information and musical content dealt with as part of the development of an innovative audio compression system, designed to exploit repetition sequences in audio, and particularly, music. The paper briefly introduces and describes how musical content and structure within audio can be exploited to achieve compression. A new system to take advantage of these hypotheses is described. The paper introduces a new file format to deal with high-level data chunks and repetitive structuring.

The practicalities of searching for large blocks of audio data are discussed and evidence provided of the high amounts of complexity involved in performing brute-force searches for self-similar matching. This high complexity is the subject of current work by the authors and research to date so far reveals that search time can be reduced by considering content analysis, this time to reduce complexity rather than data volume.

KEYWORDS

Audio compression, content analysis, complexity analysis, file formats

1. AUDIO COMPRESSION EXPLOITING REPETITION (ACER)

Statistical coding techniques for data reduction exploit repetition which naturally occurs in data. On a higher-level, music often contains sequences which are perceptually repetitive to a listener though this may not be evident upon inspection of the raw sample data. The ACER compression model is a solution to this. The aim of the ACER scheme is to identify sections of audio which are similar and repetitive; in effect a search for perceptual self-similarity. By identifying similar blocks it is necessary to encode only one such block and indicate where it should repeat elsewhere in the overall audio sequence along with any difference between the stored block and the one it replaces.

This data reduction process works by analyzing information within a waveform, repeating sections of it as required and disposing of the perceptually identical replicas. A simple example of repetition in audio would be to consider a piece of music which displays repetitive structure and sequences when examined from a macroscopic level.

For further information on the theory and concepts behind the ACER compression procedure, we refer the reader to our previous works on the topic [1, 2, 3].

2. FILE FORMAT

Given the original and innovative nature of the work presented in this paper, a new file format must be designed to deal with the detailed issues relating to the organisation and sequencing of data blocks, discovered during the ACER search procedure, to allow efficient and effective reconstruction of the original audio.

2.1. File Format Requirements

The following are essential requirements of the ACER format:

1. The ACER compression process must output a single file. The audio blocks extracted should be viewed as mini sound files which have to be stored and organised as part of the ACER procedures.

2. The format needs to be flexible to deal with various audio formats. Therefore, it becomes necessary to encode information about the characteristics of the original audio file to ensure playback of the ACER data will be achieved.

3. The instructions needed to reconstruct audio from a number of blocks in the ACER file are essential. The file must encode the order in which blocks should be structured and repeated. The size of each block is stored to ensure the length of the final audio signal matches the original. The overhead required to reference each block must be minimal.

2.2. File Format Overview

The ACER file format is a new development created as a result of the principles presented earlier in this paper. The objects, of which the file hierarchy consists, are known as chunks. This organisation is common and employed in many file formats, for example Microsoft Resource Interchange File Format (RIFF) [4, 6, 7]. The generic RIFF file structure is illustrated in Figure 1.

ChunkID	ChunkSize	ChunkData
4 bytes	*4 bytes*	*Variable Size*

Figure 1. Generic RIFF Structure

As a result of the ACER compression techniques, an important amount of information and instructions will be output. This data relates to how the audio blocks are used to reconstruct the original piece of digital audio. To this extent, we require to know the sequence in which data blocks taken from the compression process need to be reassembled to produce the required audio output. This set of instructions is stored within the ACER file format as the sequence sub-chunk. Entries in the sequence sub-chunk will effectively contain a vector of index values, where each index value points to the offset of a required block within the audio data, stored later in the ACER file. The data sub-chunk therefore, will contain all of the blocks which have been extracted during the ACER compression procedure along with any remaining unmatched material.

The specification for the ACER file format is a key development. The conception of the file format structure epitomises many of the theories and practicalities of the ACER system.

A graphical representation of the organisational structure of the ACER file format is presented in Figure 2.

Header Chunk	
ChunkID	**ChunkSize**
4 bytes	*4 bytes*
Version	
2 bytes	
Data	
Variable (Contains all sub-chunks)	
Format Sub-Chunk	
ChunkID	**ChunkSize**
4 bytes	*4 bytes*
Original File Type	**Original File Size**
4 bytes	*4 bytes*
Data	
Variable	
Sequence Sub-Chunk	
ChunkID	**ChunkSize**
4 bytes	*4 bytes*
Data	
Variable	
Data Sub-Chunk	
ChunkID	**ChunkSize**
4 bytes	*4 bytes*
Data	
Variable	

Figure 2. Detailed ACER File Format Structure

3. COMPLEXITY CHALLENGES

In this section, the theoretical and practical aspects of implementing an exhaustive search system are briefly described. In previous publications we explained the ACER search procedure in-depth and identified the high levels of computational complexity and processing time involved in performing brute-force searches required to perform the ACER search procedure [2, 3]. To this extent we provide a brief overview of complexity factors here.

3.1. ACER Search Process

Given the length of a file *f*, the file would be iteratively searched using a search block *B*, of size *b*, which would increase (or decrease) in size through the lifetime of the search. Therefore, minimum and maximum sizes of search and target block, b_{min} and b_{max}, are established before searching and a step value, Δ_b, to increment between these limits of block sizes is also set. (The assumption is made that Δ_b divides b_{min} - b_{max}.)

The search block increment across the file, the interval between one search position and the next, is defined as Δ_s, and will effectively allow a fineness or depth facility to the function. Similarly, the target block incremental parameter, Δ_t, provides an offset for the target block where the match for *b* is sought. This is illustrated in Figure 3.
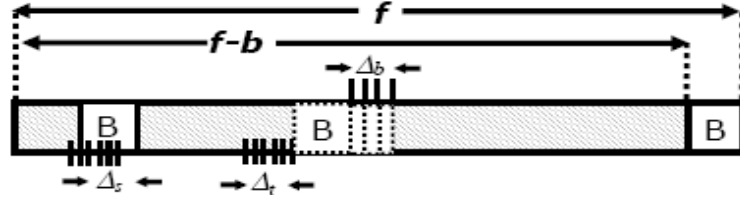
Figure 3. ACER Search Components

The complexity, or number of steps required to match one block of size b, is generically defined 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

$$C(f, b_{\min}, b_{\max}, \Delta_b, \Delta_s, \Delta_t) = \sum_{b=\frac{b_{\min}}{\Delta_b}}^{\frac{b_{\max}}{\Delta_b}} \sum_{s=1}^{\frac{f-\Delta_b b}{\Delta_s}} \sum_{t=s+b}^{\frac{f-\Delta_b b}{\Delta_t}} X(\Delta_b b). \quad (1)$$

Since the length of the file is considered to be static at the start of the compression process, the five parameters (b_{min}, b_{max}, Δ_b, Δ_s, and Δ_t) are all, in effect, measures of quality or effectiveness of the search process and could be defined at the beginning of any search. To provide simplification, these parameters are consolidated into one value: $\Delta = \Delta_b = \Delta_s = \Delta_t$. To provide additional generalisation, the assumption is made that $b_{min} = \Delta_b = \Delta$ and $b_{max} = f/2$, since, it is unnecessary to search for a block greater than half the length of the file (since there can be no repetition). Finally, if $X(b)$ can be approximated by an invariant term X, not dependent on b, and assume $f >> b$, so that all $(f - \Delta b)/\Delta$ terms reduce to f this reduces the expression to

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

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 analysis and matching techniques employed.

3.2. Practical Search Time

To translate complexity into meaningful real-world data, sample searches were run. The results for small tests, for a file length of 10000 samples (approximately a quarter of a second at CD quality) are given in Table 1.

Table 1. ACER Sample Search Results (f=10000)

b_{min}	b_{max}	Δ_b	Δ_s	Δ_t	Time (minutes)
2	1000	1	1	1	5892.80
2	1000	10	1	1	591.75
2	1000	100	1	1	88.28

Computational overheads for compression systems rarely operate in real-time and typically moderate amounts of delay are acceptable. In archival compression, the acceptable level of delay is increased. ACER feasibly falls into the later category, used to gain extra compression at a higher cost of time. Additionally, the decompression of ACER data will incur almost no overhead whatsoever.

The amount of time required to carry out the ACER search procedure is excessive and, for large amounts of data, performing searches of this magnitude is impractical and unacceptable for the majority of application areas. Even though useful compression might be achieved, a technique which takes literally days to operate would be unfit for all but the most low-priority of data archiving applications. Clearly, there is a need to optimise the search. The size of the file is the single biggest factor which contributes to the overall complexity of the exhaustive search process. However, we consider file size to be fixed prior to the search. To improve the search we consider the *content* or *information* being represented by the data.

4. CONTENT ANALYSIS SOLUTIONS

Search time can be reduced by considering the musical content contained within the audio. It is proposed that by deriving suitable information about the information contained in an audio file, the search parameters can be configured to suit the particular data content and therefore reduce the need for a brute-force approach.

The search parameters employed are temporal measures. That is to say, block sizes and increments, though measured in samples. Therefore, to optimise the search, the evident domain in which to look is time. Based on the assumption that musical content is contained in audio, there are a number of musical properties which can be exploited for this purpose.

4.1. Beat Detection

One suitable translation of the time domain occurs in audio signals that contain music. Music is heavily controlled by the *tempo*. Tempo provides a regimented timing and commonly stays static for the duration of the song (although this is not always the case). Tempo is measured in Beats per Minute (BPM). As the BPM acronym infers another musical term is the *beat*.

Knowledge of the tempo is valuable to the ACER search. If we consider the search processes, the block size b and block step increments Δ_b, Δ_s and Δ_t play a major role in complexity of the search. Since one of the main concepts of ACER compression is to exploit repetition within musical audio, a suitable size for blocks and increments should be related to the repetitive content. Beat duration and tempo are highly suitable starting points in determining block size and increments since repetitive sequences or loops will generally start and end on a beat, or a multiple of a beat.

We experimented with several filter banks when using Scheirer's beat detection algorithm on several test songs [8]. The songs, and human estimated tempos are shown in Table 2 and comparison between the estimate and several filters is shown in Figure 9.

Table 2. Song & Tempo Estimation

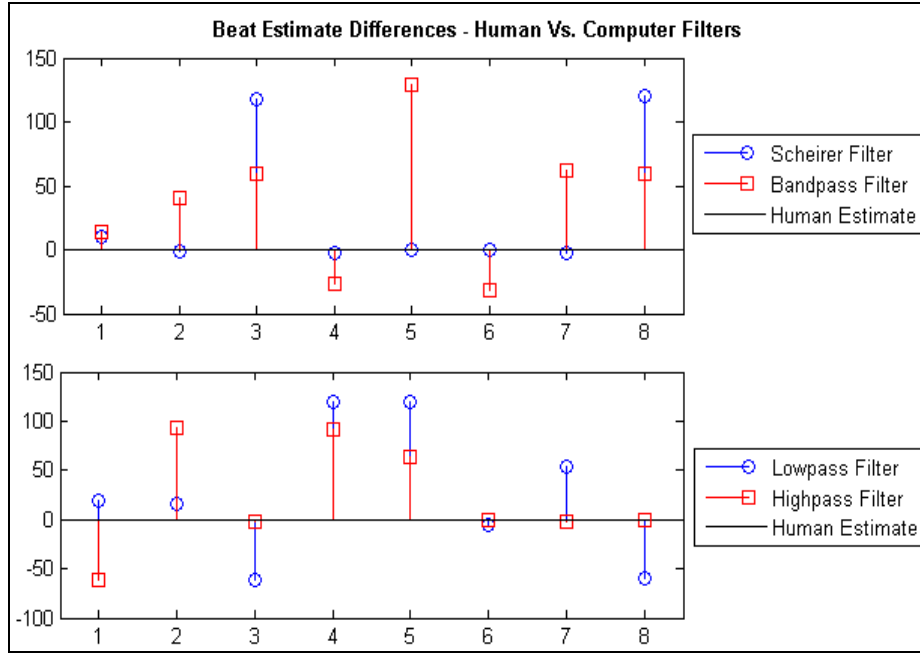| Artist & Song	#	Estimate
Daft Punk – *One More Time (Radio Edit)*	1	130
Fun Lovin' Criminals – *Love Unlimited*	2	95
Hot Chip – *Over and Over*	3	122
Metallica – *Harvester of Sorrow*	4	94
Pink Floyd – *Comfortably Numb*	5	64
Sugababes – *Push The Button*	6	126
The Prodigy - *Breathe*	7	132
ZZ Top – *Gimme All Your Lovin'*	8	120

Figure 4. Differences in Tempo Estimation

Table 3 presents a comparison between the computational steps required in a basic ACER search against several searches with tempo-informed parameters. The reduction is significant.

Table 3. Standard and Tempo Informed Searches (f=132000)

Standard ACER Search					
b_{min}	b_{max}	Δ_b	Δ_s	Δ_t	Complexity
2	66000	1	1	1	191653110186999
ACER Searches with Tempo Informed Parameters					
b_{min}	b_{max}	Δ_b	Δ_s	Δt	Complexity
4020	16080	4020	1	1	25205047800
16080	32160	4020	1	1	17862761400
16080	64320	4020	1	1	23189361000
16080	64320	16080	1	1	7910887200

4.2. Identification of Musical Structure

The aim of gaining an understanding of the structure of the audio is to eliminate time spent searching areas of the file initially identified as significantly dissimilar to the current search block.

By constructing an Audio Similarity Matrix (ASM) an overview of the musical structure is gained which presents the self-similarity within the audio [5]. Whilst the ASM does not measure similarity in exhaustive detail, this is countered by relatively quick processing time. The similarity measurement is achieved by taking the Euclidean distance between two windows. The ASM is presented for human viewing, where level of similarity is represented in an image as a colour or using luminance in a greyscale image. Figure 5 shows an example ASM. Structure appears as a block and chequerboard pattern. In Figure 5, dark colours (blues in the spectrum) are used to indicate areas of high similarity and the inverse applies for bright colours (reds in the spectrum), which represent distant sections.
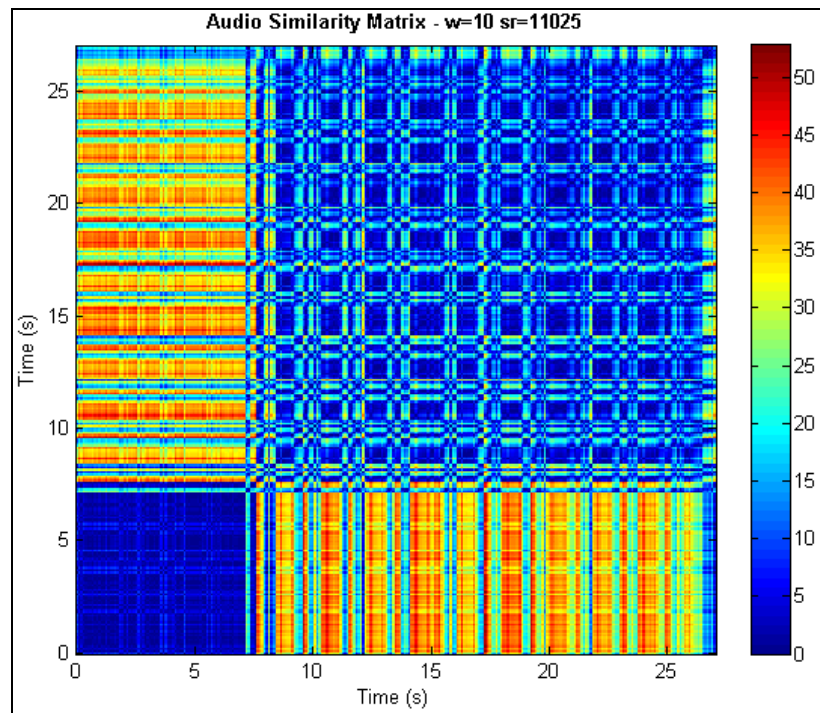

Figure 5. ASM of Head Over Heels by ABBA (28 seconds)

From the ASM, various sizes of similar areas are identifiable. This allows zoning of the audio to be compressed. By identifying similar zones, dissimilar zones need not be considered by the search and target blocks, therefore reducing the overall complexity and time required. The effect of this process effectively reduces the file length required for each given iteration of the ACER search process. Applicability of this technique is subjective, as it depends upon the individual file being processed. Certain musical genres will have higher levels of repetition and structure than others.

5. CONCLUSIONS & FUTURE WORK

The savings of complexity and running time detailed by using beat detection are valuable. However, there are parameters which can be more finely tuned to produce more reduction in computation time. For example, it is generally more desirable and efficient to discover longer sections of repetition, such as that of one or more musical bars. A bar consists of a logical grouping of beats, governed by the time signature of the music. We are currently investigating time signature detection.

Down-sampling provides a simple technique by which reductions in complexity can be achieved. In signal processing it is common to encounter processing techniques which reduce the sample rate of the audio prior to any computationally intense processing. Here, analysis could be performed on a down-sampled file and the similarity results scaled back as indices to the high-quality version to prevent information loss.

Finally, distributed processing is being investigated. The ACER process segments are not dependant upon each other during searching. Iterations of the ACER search are independent. This means processing can be parallel.

REFERENCES

[1]   Cunningham, S. 2005. Waveform Analysis for High-Quality Loop-Based Audio Distribution, Proceedings of ISCA 20th International Conference on Computers and Their Applications, New Orleans, USA, 16th-18th March.

[2]   Cunningham, S. & Grout, V. 2007. Advances in Similarity-Based Audio Compression, Proceedings of the Third Collaborative Research Symposium on Security, E-Learning, Internet and Networking (eds. U.G. Bleimann, P.S. Dowland & S.M. Furnell), ISNRG, Plymouth.

[3]   Cunningham, S., Grout, V., & 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, Lisbon, Portugal, 19th-22nd October.

[4]   EBU. 2001. BWF – A format for audio data files in broadcasting. Version 1, TECH 3285 Technical Specification, European Broadcasting Union, Geneva, Switzerland.

[5]   Foote, J. 1999. Visualizing music and audio using self-similarity. Proceedings of the seventh ACM international conference on Multimedia (Part 1), 77 – 80, Orlando, USA.

[6]   Grande, C. 1997. The Notation Interchange File Format: A Windows Compliant Approach, Beyond MIDI: The Handbook of Musical Codes, 491 – 512, MIT Press, ed. Selfridge-Field, E.

[7]   Plichta, B., Kornbluh, M. 2001. Digitizing Speech Recordings for Archival Purposes. Working Paper, Matrix, Michigan State University, USA.

[8]   Scheirer, E.D. 1998. Tempo and beat analysis of acoustic musical signals. Journal of the Acoustical Society of America, (1) 103, 588-600