Audio object coding for distributed audio data management applications

AUDIO OBJECT CODING FOR DISTRIBUTED AUDIO DATA MANAGEMENT APPLICATIONS

Kathy Melih[1], Ruben Gonzalez[2]

School of Information Technology, Griffith University, PMB 50, Gold Coast Mail Centre, QLD, 9726
[1]K.Melih@gu.edu.au, [2]R.Gonzalez@gu.edu.au

Abstract - The now ubiquitous WWW provides ready access to a plethora of MM data. However, the MM information management facilities of the WWW are little better than those that are several decades old. The audio domain in particular has received little attention in this area. The authors have addressed this oversight by developing a structured audio representation based on "audio objects", allowing more useful access than traditional methods. The representation has also been designed for the demands of a networked world where efficient transmission capabilities are a priority. This paper describes how the decomposition of audio into objects can be exploited for compression gain.

Keywords – Audio coding; hyperaudio; structured audio.

I. INTRODUCTION

In 1945 Vannevar Bush published his vision for the ideal information storage, retrieval and management system [1]. Today, it is apparent that the WWW has finally made the vision a reality. However, in its current form, the WWW falls well short of Bush's vision. Rather it is a "multimedia enhanced" hypertext system. This is largely due to continued dependence on unstructured multimedia data representations.

Unstructured representations result from the need for compression. However, this comes at a cost: ready access to vast volumes of information that are disorganised and difficult to access in a random fashion. Indeed, modern access mechanisms for audio have not progressed beyond the simple functions offered by the most basic audio tape players: play, rewind, etc and modern audio information management techniques are still either impractically complex to implement or too low level to be useful for the general user.

By contrast, audio cognition is highly structured, with often complex mixtures, separated into individual acoustic 'events' or 'objects'[7]. Further, when recalling a long audio 'scenario' (e.g. piece of music) it is most natural to be able to recall individual events (e.g. key change) than a temporal location relative to the start. Currently, the latter access method is essentially the only one available.

Existing content-based access schemes do not satisfactorily solve this problem as they are also unstructured. Two general classes exist: statistical and transcription schemes. Statistical methods provide little relevant information to general users (e.g. musicians won't know the FFT centroid of a cello tone). Transcription methods (e.g. speech recognition or melody retrieval) may be useful in constrained environments. However, they do not reflect the natural organisational structure in the data. These schemes are akin to a book with no chapters, headings or paragraphs. An index may be useful but, with no obvious data organisation, it is difficult to gain a conceptual view of information presented.

Clearly, a duality exists between providing compression of audio data and meaningful content-based access and retrieval mechanisms. Presently, attempts to resolve one exacerbates the other. The only feasible solution is a compact audio representation that reflects useful information about the underlying data *and* its organisation.

The authors previously described the low level aspects of a structured representation designed specifically to support audio object extraction [3] and the compression possibilities for the lower levels of this representation [4]. They then detailed audio object extraction [5]. Now the issue of compression is revisited with respect to audio objects.

II. ELEMENTARY AUDIO OBJECT FORMATION

A. Definition of "Audio Objects".

An audio object is a single semantically relevant unit that is individually decodable and randomly accessible[6]. Audio objects exist in a hierarchical framework of elements that each display these characteristics. Figure 1 shows the hierarchical decomposition of an audio stream.

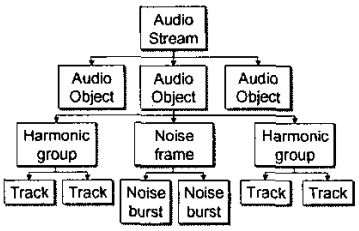

Figure 1: Structured audio representation

The lowest level contains the perceptually relevant atomic features of an audio signal. These reflect the features extracted by the lowest levels of the auditory system. Essentially two main feature classes are extracted: track and noise burst. Tracks can be classified as either 'tone', 'sweep' or 'formant'. Track formation is detailed in[3].

0-7803-7510-6/02/$17.00 ©2002 IEEE

Harmonic groups and noise frames appear at the next level up. Harmonic groups consist of tracks, co-located in time, bearing some resemblance to one another. Generally, the frequency and amplitude contours will be scaled versions of a single 'prototype'. This property can be exploited to achieve compression gains for harmonic groups. Noise frames consist of temporally adjacent noise bursts with similar characteristics (RMS power, spectral envelope, etc). [5] detailed the formation of these mid-level groups. This paper describes the compact representation of harmonic groups.

To provide content-based random access, every element of the structure is randomly accessible and individually decodable. However, from a user's perspective, the lowest levels of the hierarchy are unlikely to have any significance. Hence, it is much more likely that a combination of at least two mid-level elements is required to create an inversion that is both intelligible and of acceptable quality. Thus, from both encoding and retrieval aspects, coding harmonic groups as individual objects is preferable to encoding individual tracks.

B. Generation and Inversion

The lowest level of the structure presented in the previous section consists of a series of time-frequency-amplitude trajectories that are non-uniformly sampled in each dimension. The trajectories are composed of amplitude peaks in short-time spectra. These peaks are 'tracked' through time according to low-level psychoacoustic principles.

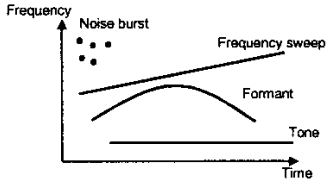

Figure 2: Basic perceptual elements of auditory data

The basic process used to obtain the elements in Figure 2 was detailed in [3] and is summarised in Figure 3. Incoming audio is analysed to produce a TFD. The peaks in amplitude of this distribution are found and tracked in time and frequency. Finally, the tracks are classified and grouped according to psychoacoustic principles and encoded in these groups.

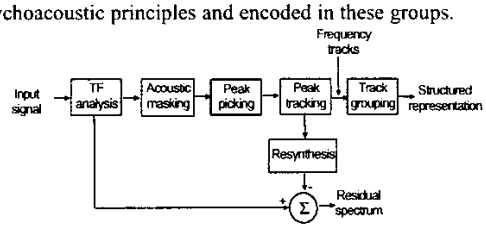

Figure 3 Coding algorithm

These elements may be inverted using the procedure illustrated in Figure 4. This inversion technique makes it possible to invert individual tracks or groups of tracks. This is consistent with the aim of maintaining access to individually decodeable, randomly accessible units.

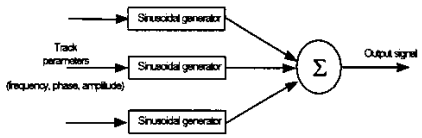

Figure 4: Track inversion

III. ENCODING THE OBJECT PRIMITIVES

In providing for compact storage it is important to note that the structure revealed in both individual tracks and harmonic groups cannot be obfuscated. This contrasts with traditional coding schemes that exploit statistical redundancies in the data at the expense of structural information. Hence the selection of coding techniques is somewhat limited.

A. File structure

The proposed file structure consists of a 24 byte file type and version code followed by a list of the position of each harmonic group in the file. This list is terminated by a null code. Each harmonic group is preceded by it's own header information described in section III.C.3). Since each harmonic group is effectively self contained, it is simple for some high level process to assign a new organisation by creating a list of pointers to individual groups. Further, if access to a single track is required, the groups themselves are encoded in a structured manner such that it is a relatively straightforward matter to define a link to even a single track.

B. Harmonic Group Description

The information required for each harmonic group is:

(a) Number of tracks (harmonics)
(b) Model Track class: {TONE, SWEEP, FORMANT}
(c) Model Frequency v's time contour
(d) Harmonic number and freq offset of each track
(e) Model Amplitude v's time contour
(f) Amplitude v's harmonic scale factor for each track
(g) Model Phase v's time contour
(h) Phase v's harmonic phase offset for each track
(i) Start and stop time of group model
(j) Start and stop of all tracks (relative to group model).

The necessity of items a), c)-e), g) and i) should be obvious. Item b) provides inbuilt indexing as the track shape provides information useful to determine the audio source. For example, 'formant' shaped tracks indicate speech. A frequency offset from the exact harmonic frequency must be recorded as harmonics of many natural sounds are not exact.

C. Coding Techniques

Coding techniques for individual tracks were suggested in [4]. Given the harmonic grouping now achieved it is possible

to exploit redundancy within a group improved compression as we now only need to store detailed information about a single track per group with a small amount of overhead to accommodate individual tracks in the group. To access the details for an individual track, simple scaling or offset of the corresponding parameters for the group model is all that is required. Hence, we can now achieve a much greater coding gain while maintaining direct access to the individual elements of the representation.

1) Amplitude and Frequency Contours

The amplitude and frequency contours for the tracks vary slowly in time. Hence, DPCM is an ideal way to encode them. To further increase the coding gain, variable length codewords can be used.

The first step is to find the contours' numerical gradients:

$$df(t_i) = f(t_{i+1}) - f(t_i) \text{ and } dA(t_i) = A(t_{i+1}) - A(t_i)$$

where $f(t_i)$ and $A(t_i)$ represent the frequency and amplitude values respectively of the model track at time t_i. Huffman coding assigns variable length codewords.

Thus for each model contour the following quantities must be recorded: an initial value, the Huffman codebook and the Huffman encoded values. Table 1 summarises the bit allocation for the amplitude and frequency contours.

Table 1: Summary of bit allocation for frequency and amplitude contours

Aspect	Number of bits
Initial value	7 bits (amp) 10 bits (freq)
Code book	# symbols * 7 + sum of symbol lengths
Contour (approximately)	# of frames * entropy of gradient function

Since the values of the amplitude contour vary from 0 to 90 dB SPL, with a resolution of 1 dB SPL is, 7 bits are used for the initial value. Similarly, the frequency range is 0 to 1 kHz with 1 Hz resolution, so 10 bits are used. Each entry in the codebook follows the pattern: value – 4 bits; codeword length – 3 bits followed by the codeword – 7 bits max.

2) Phase Contour

In order to achieve acceptable inversion, the phase contour must be recorded more accurately than the amplitude and frequency contours [4]. Further, the phase contour generally has a higher entropy than the amplitude and frequency contours thus a greater number of bits per frame will be required. Indeed, the entropy value is sufficiently large that Huffman coding is considered unjustified.

3) Group-Specific Overhead

By grouping the tracks together, a small amount of overhead information is required. This overhead is represented by items (a); (d); (f); (h); and (i) listed in section III.B. However, this group overhead is significantly less then the gain achieved by grouping the tracks.

Table 2 shows the bit allocation for each of the overhead elements mentioned above.

Table 2: Bit allocation for track group overhead information

	# of Bits	Description
(a)	5	Number of tracks
(b)	2	Track type (TONE/SWEEP/FORMANT)
(d)	6 bpt*	3 bits harmonic number, 3 bit offset
(f)	3 bpt	Amplitude scale factor per harmonic
(h)	4 bpt	Phase offset per harmonic
(i)	16 + 8	Group start and duration relative to start of file (or block)
(j)	(8+8) bpt	Track start and duration relative to group

* bpt = bits per track

A conservative estimate, based on observation of naturally occurring sounds and the frequency response of the human auditory system, for the maximum number of harmonics in a group is 30. Hence, 5 bits are allocated for this purpose.

If the harmonics are ordered within the group by their harmonic numbers, the harmonic number can be differentially coded using 3 bits per track. This assumes that at most 7 consecutive harmonics will be 'missing'. This is reasonable as even harmonically related tracks separated by a greater distance would be segregated into separate groups according to the perceptually-based grouping rules used.

Tracks further than a frequency dependent threshold (with a maximum value of 20Hz) away from the expected harmonic value, are considered to not be harmonics. Hence, we would expect the values for the frequency offset to fall within the range–20 to + 20 Hz. Also the offset, if present, tends to increase with harmonic number. Again, ordering the tracks by harmonic number and differentially encoding the offsets ensures that 3 bits is sufficient to record this information.

The start time of the tracks is the number of analysis frames since the beginning of the file. There are 125 frames per second. With 16 bits allows for 65536 frames or over 8.5 minutes of audio data. This is easily extended by including a special field in the file header to define blocks of 8 m long each. In this case the start field for a group is relative to the beginning a block. The stop time is encoded relative to the start time. By the definition of a harmonic group, it is extremely unlikely that one would exceed 1-2 seconds in length, hence, 1 byte is used for this value. However, this field is extensible with a flag in the group header signifying either a 1 or 2 byte length field.

The start time of each track is recorded relative to the start of its group. The length of the track is also recorded. In the case of an extended group length, the track length is also extensible. The first bit in the 8-bit length field is taken as a flag. If it is set to OFF, that one byte comprises the length field. If the first bit is set to ON, then a second byte follows.

In addition to these essential features, 1 byte is allocated as a group identifier, a further byte is reserved for flags and 4 bytes have been allocated for use in indexing, or other structuring mechanisms[6]. Given this bit allocation, the format for each group is shown in Figure 5.

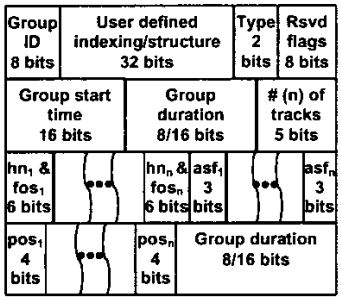

Figure 5: Header format for each track group

IV. RESULTS

To validate the technique, a simple yet representative test file is analysed[5]. The file a combination of speech and music. The speech is a female speaker pronouncing the vowel 'U' while the music is the piano note b2. The two signals were artificially mixed such that the spoken utterance occurred simultaneously with the centre of the note. The input is a 16-bit PCM WAV sampled at 32kHz.

Figure 6 shows the results of the track formation and grouping algorithm illustrated in Figure 3. It may be surprising to see that the tonal tracks (from the solo piano tone) have been assigned to two separate harmonic groups. This is because the higher frequency harmonics of the piano tone do indeed 'mistuned'. (cfFigure 7).

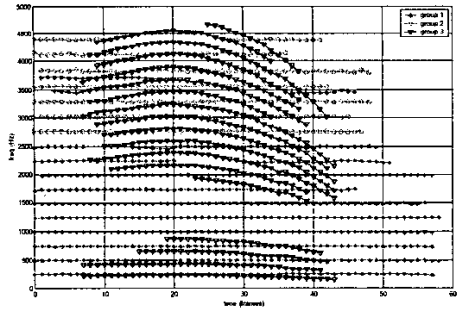

Figure 6: Tracks and groups for mixed source data

To determine the amount of data required to encode each of these groups, the parameters mentioned in section 4 need to be derived. We must then perform a statistical analysis of each trajectory used to represent the model tracks and determine the remaining header information that pertains to each individual harmonic. The basic group duration and membership information is summarised in Table 3.

Table 3: Group duration and number of constituent tracks for each group in the mixed data set

Group	Start frame	Length	# of tracks
1	0	59	16
2	0	49	8
3	7	37	18

Figure 7 and Figure 8 show the frequency and frequency gradient functions respectively. Clearly, the data are highly correlated.

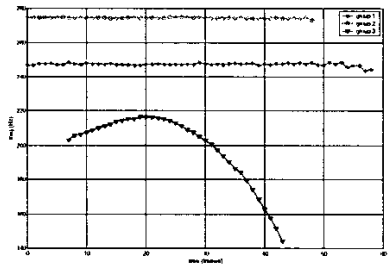

Figure 7: Frequency contours of models derived for track groups

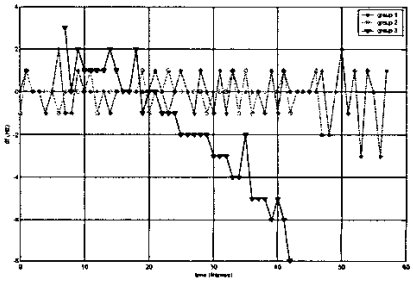

Figure 8: Frequency gradient functions

Table 4 summarises the bits required to represent the frequency contours of each group. The codebook size and total bits required to code each gradient function were determined by generating Huffman codes. The total figure includes the 10 bits used to code the initial frequency value.

Table 4: Summary of bit allocation for the frequency contour

	Grp 1	Grp 2	Grp 3
Number of symbols	6	4	11
Entropy	1.875	1.385	3.274
Codebook size in bits	62	38	120
Total bits for gradient function	92	82	113
Total bits for freq contour	**164**	**130**	**243**

Table 5 summarises the number of bits required to code the amplitude contours of the model tracks.

Table 5: Summary of bit allocation for the amplitude contours of the group models

	Grp 1	Grp 2	Grp 3
Number of symbols	6	6	8
Entropy	2.04	2.00	2.51
Codebook size in bits	62	64	85
Total bits for gradient function	121	95	92
Total bits for freq contour	**190**	**166**	**184**

Figure 9 shows the phase contours of the group models. It is clear that, unlike the frequency and amplitude data, the phase data are highly uncorrelated. Hence fixed length codewords are used with the codeword length determined by the range of phase values for the track. The first 4 bits of the phase description indicates this length.

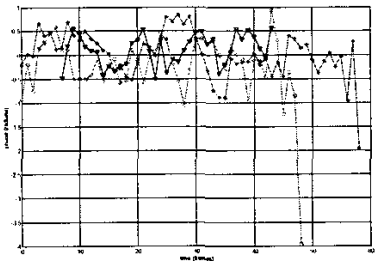

Figure 9: Phase 'contour' of each group model

Table 6: Summary of bit allocation for phase contours of group model

	Min Value	Max Value	Bits per frame	Total bits for track
Group 1	-1.95	0.85	9	535
Group 2	-1.23	2.31	9	445
Group 3	-0.49	0.57	7	263

Table 7: Summary of bit allocation for all track elements in the mixed data file

Item	Group1	Group 2	Group 3
(a)	5	5	5
(b)	2	2	2
(c)	164	130	243
(d)	96	48	108
(e)	190	166	184
(f)	48	24	54
(g)	535	445	263
(h)	64	32	72
(i)	24	24	24
(j)	16	16	16
head	50	50	50
Total	**1194**	**944**	**1024**

Table 7 shows that a total of 3215 bits is required to describe the track data. In addition to this the file contains a 24 byte general file header and an index header requiring of 32 bits/group plus an additional 32 bits to terminate the header. This gives a total file size of 3439 bits = 430 bytes. The original file size was 34,860 bytes hence a compression ratio of 81:1 has been achieved. However, this analysis has neglected one of the classes required in our structured data model: noise. It can be expected that the addition of the noise class will approximately half this value to give a compression ratio of approximately 40:1.

V. DISCUSSION AND CONCLUSIONS

The authors have presented a review of the development of an object-based structured audio representation designed specifically to support advanced applications such as hypermedia. This review was followed by an analysis of the representation' s suitability for compressed data storage and transmission by suggesting suitable coding techniques for the objects.

Having successfully extracted harmonic groups from the low-level track information, has facilitated the exploitation of redundancy in these harmonic groups. By doing so we have greatly improved upon the compression ratio of 9:1 achievable when each individual track is encoded separately. While formal listening tests are yet to be conducted, subjective audio quality of each reconstructed group is sufficient to be recognised as the original source. Further, each group is generally free from artefacts of the other sources in the original mix, with any evidence of the original mix being apparent only in ' noise' groups.

REFERENCES

[1] Bush, V., "As We May Think", *Atlantic Monthly*, July 1945, 101-108.

[2] ISO, "MPEG-7 Requirements Document V.9", N2859, Vancouver Canada, July 1999.

[3] Melih, K. and Gonzalez, R., "Audio Retrieval Using Perceptually Based Structures", *Proc. IEEE International Conf. On Multimedia Computing and Systems '98*, Austin, Texas, 28 June - 1 July 1998, 338-347.

[4] Melih, K. and Gonzalez, R., "Structured Coding for Content Based Interactive Audio", *Proc. IEEE International Conf. On Multimedia Computing and Systems '99*, Florence, Italy.

[5] Melih, K. and Gonzalez, R., "Source segmentation for Structured Audio", *Proceedings of the IEEE International Conference on Multimedia and Expo* (ICME), New York, 2000.

[6] Melih, K. and Gonzalez, R., "Towards True Hyper-Audio", ISPA 2001.

[7] Bregman, A.S., Auditory Scene Analysis: the Perceptual Organisation of Sound, MIT Press, 1990