MPEG-4 ALS: an Emerging Standard for Lossless Audio Coding

Tilman Liebchen
Technical University of Berlin
Einsteinufer 17, D-10587 Berlin, Germany
E-Mail: liebchen@nue.tu-berlin.de

Yuriy A. Reznik
RealNetworks, Inc.
2601 Elliott Avenue, Seattle, WA 98121
E-mail: yreznik@ieee.org

Abstract

We provide a brief overview of an emerging ISO/IEC standard for lossless audio coding, MPEG-4 ALS. We explain choices of algorithms used in its design, and compare it to current state-of-the-art algorithms for lossless audio compression.

1 Introduction

Lossless compression has many applications in recording and distribution of audio. For instance, bit-exact representation is required (or strongly desirable) in archival systems, studio operations, for collaborative work or music distribution over the Internet, etc. Even wider use of lossless audio compression in the future should be enabled by continuing growth of capacities of storage devices, transition to high-speed Internet connections, and deployment of broadband wireless networks.

Considering these trends, existing application scenarios, and industry's interest in a common format for lossless compressed representation of audio signals, the MPEG Audio group has started work on defining lossless audio coding technology for ISO/IEC 14496-3:2001 (MPEG-4 Audio) standard [11]. The main requirements for this technology, formulated in the call for proposals [12], issued in November 2002, were:

- provide lossless reconstruction of PCM audio signals at sampling rates from 44.1KHz to 192KHz and word lengths of 16, 20, and 24 bits,
- achieve superior compression efficiency to all currently known algorithms. Scalable algorithms (i.e. ones embedding a lossy MPEG audio stream) must achieve lower rate than the rate of a simulcast transmission of the base layer and a losslessly encoded signal,
- provide means for editing, manipulations, and random access to the compressed audio.

In the following period, the MPEG Audio group has evaluated 6 scalable and 3 lossless-only compression technologies [13] submitted in response to the call for proposals, and in March 2003 the group has made a decision to proceed with the standardization of the lossless-only scheme first, while continuing further investigation of scalable proposals. The lossless-only proposal from Technical University of Berlin [18] has been chosen as a reference model, and a coding scheme for prediction residual proposed by RealNetworks [24] has been accepted as a first successful core experiment.

In July 2003, the specification of this algorithm has received a working draft (WD) status [14, 15], and pending a period for additional core experiments it is scheduled to become an international standard ISO/IEC 14496-3:2001/AMD 4, Audio Lossless Coding (abbreviated as MPEG-4 ALS) by the end of 2004.

In this paper we provide a brief overview of this emerging standard and explain the choices of algorithms used in its design.

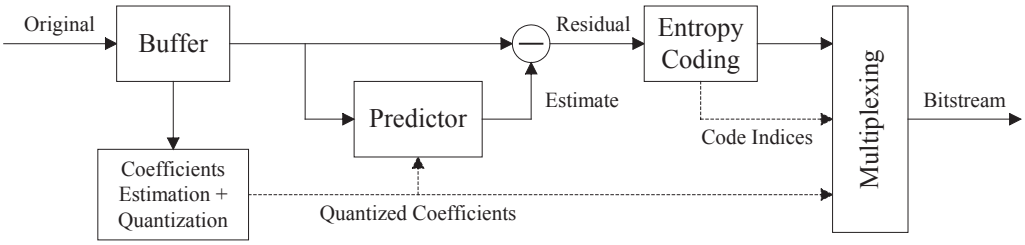

Figure 1: MPEG-4 ALS encoder.

2 The Concept and the Structure of the Algorithm

The idea of the MPEG-4 ALS encoder is based on a well-known and well-studied principle of *predictive coding* [4, 19]. Early examples of predictive coders include delta-modulation, DPCM, and adaptive DPCM schemes [19, 7]. *Linear Predictive Coding (LPC)* is a further development of this idea incorporating linear filter with properly selected coefficients as a predictor. While originally proposed and developed as a technique for speech compression [1, 22], LPC has found many other applications, and in recent years it emerged as a most common and practical technique for lossless audio compression [26, 2, 29, 10].

In the design of MPEG-4 ALS we have adopted the concept of forward-adaptive linear predictive coding, and tried to define an implementation that is suitable for a wide range of today's computing environments, such as PCs, home theater systems, mobile devices, personal music players, etc.

2.1 Structure of the Encoder

The structure of the MPEG-4 ALS encoder is presented in Fig.1. It consists of the following building blocks:

- *Buffer*: Stores an audio frame. Each frame consists of one or more blocks of PCM samples representing signals in each audio channel.
- *Coefficients Estimation and Quantization*: Estimates and quantizes optimal predictor coefficients for each block.
- *Predictor*: Calculates prediction residual using quantized predictor coefficients.
- *Entropy Coding*. Encodes the residual using Golomb-Rice or higher-efficiency codes.
- *Multiplexer*: Combines the encoded residuals, code indices, and predictor coefficients in the compressed bitstream.

For each audio channel, an optimal predictor order and coefficients are estimated and used to produce the prediction residual. The coefficients are quantized prior to the filtering and transmitted as a side information. The prediction residual is encoded using one of several different entropy codes. The indices of the chosen codes are also transmitted as a side information. Finally, the multiplexing unit combines encoded residual, code indices, predictor coefficients and some additional information to form the compressed bitstream. The encoder also calculates and transmits a CRC checksum, which allows decoder to verify the correctness of the decoded data.

Additional encoder functions include block length switching, random access and joint stereo coding. We will describe them in further details in Section 3.

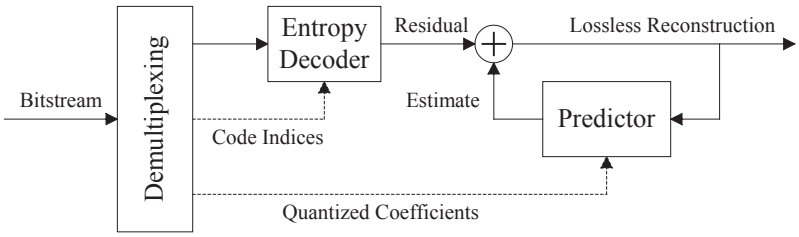

Figure 2: MPEG-4 ALS decoder.

2.2 Structure of the Decoder

The structure of MPEG-4 ALS decoder is presented in Fig.2. Most of its components have the same functions as in the encoder. The demultiplexing unit splits the bitstream in the encoded residual, code indices, and the coefficients. The entropy decoder reconstructs the residual, which is then added to the predicted samples to produce the decoded signal.

3 Description of the Encoding Process

3.1 Linear Prediction

The current sample of an input PCM audio signal x_i is predicted from previous samples x_{i-k} $(k = 1 \dots p)$ using

$$\hat{x}_i = \left\lfloor \left(\sum_{k=1}^{p} a_k \, x_{i-k} + 2^{Q-1} \right) / \, 2^Q \right\rfloor , \tag{1}$$

where p is the order of the predictor, a_k are filter coefficients, and Q is the number bits reserved for fixed-point representation of coefficients. The residual samples r_i represent the difference:

$$r_i = x_i - \hat{x}_i \, . \tag{2}$$

The estimation of linear predictor coefficients a_k can be done using standard techniques, such as autocorrelation or covariance methods [7, 16]. In our case, however, the autocorrelation method employing Levinson-Durbin algorithm [7] has a clear advantage in providing simple means for finding the optimal order of the predictor. Thus, the Levinson-Durbin algorithm estimates parameters of all predictors recursively, in an increasing order of the predictor. For each order a complete set of predictor coefficients is calculated. Moreover, the variance of the corresponding residual can be calculated, resulting in an estimate of the expected bit rate for the residual. Together with the bit rate for the coefficients the total bit rate can be determined in each iteration, i.e. for each predictor order. The optimal order is a point where the total bit rate is minimal.

3.2 Quantization of Predictor Coefficients

Quantization and encoding of LPC coefficients have already been extensively studied in the design of speech compression algorithms [1, 19, 22]. Among the most efficient techniques are: a) the method based on quantization of roots of the predictor polynomial $A(z) = 1 - \sum_{k=1}^{p} a_k \, z^{-k}$ (LSP method), and b) the method based on quantization of the LAR (Log Area Ratio) function $f(\gamma_i) = \ln \frac{1-\gamma_i}{1+\gamma_i}$ of the *parcor* coefficients γ_i of the predictor.

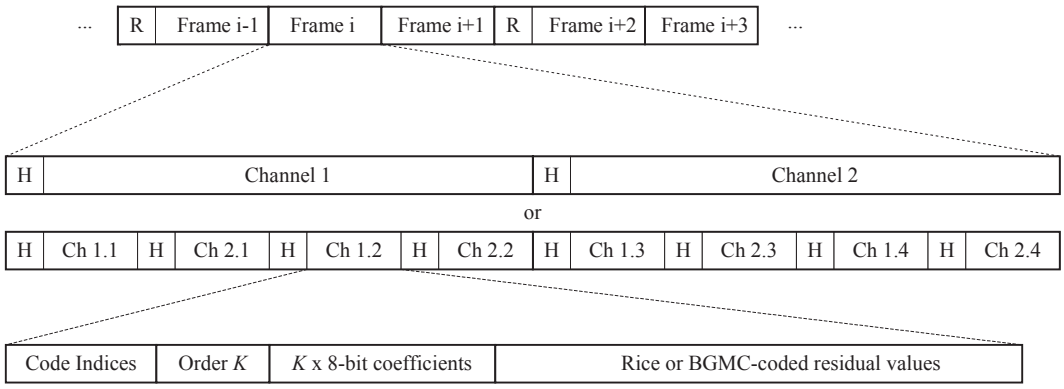

Figure 3: MPEG-4 ALS bitstream (R - random access information, H - block header).

In MPEG-4 ALS we have chosen to quantize parcor coefficients, however, instead of using the standard LAR compander we have decided to use the arcsine function:

$$\alpha_k = \arcsin(\gamma_k) \,, \tag{3}$$

which gives us a restricted to $[-\pi/2, +\pi/2]$ range of values to quantize. This modification allows us to use simple uniform quantization and direct fixed-length codes for transmission of coefficients α_k. The MPEG-4 ALS also specifies an integer-arithmetic function for conversion between quantized values α_k and direct predictor coefficients a_k which insures their identical reconstruction in both encoder and decoder.

3.3 Block Length Switching

The MPEG-4 ALS encoder processes an input audio signal using fixed-length chunks (frames). The length of an audio frame can be selected at the beginning of encoding of a sequence (e.g., based on the sampling rate of the input signal). To allow faster adaptation to transients in audio signals MPEG-4 ALS also includes a mode in which each block in a frame is further sub-divided in four sub-frames (see Fig.3).

3.4 Random Access

In order to provide random access and easy navigation through the compressed bitstream, MPEG-4 ALS allows insertion of special random-access frames (see Fig.3). In such frames, no samples from previous frames are used for the prediction. Each random access frame starts with an field that specifies the distance to the next random access frame, thus enabling standard fast forward, search, and rewind operations in compressed files.

3.5 Joint Stereo Coding

In processing of stereo signals MPEG-4 ALS allows either independent encoding of both left x_i^L and right x_i^R channels, or the use of a difference signal

$$d_i = x_i^L - x_i^R \tag{4}$$

instead of x_i^L or x_i^R. Switching between x_i^L, x_i^R and d_i in each particular frame depends on which of the two signals can be coded most efficiently.

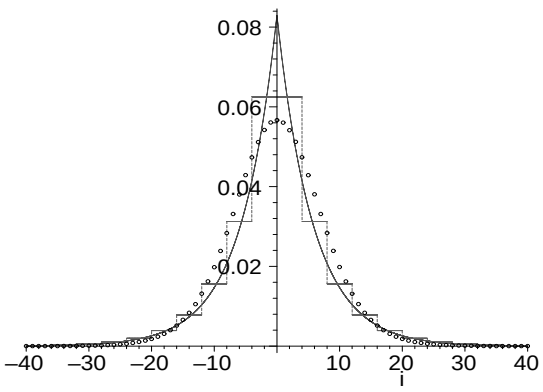

Figure 4: Observed residual distribution (circles), model Laplacian distribution (solid line), and the distribution corresponding to the lengths of Golomb-Rice codes (dashed line).

3.6 Coding of Prediction Residual

In its current version, the MPEG-4 ALS specification describes two alternative coding schemes for prediction residual. The first scheme employs simple and fast Golomb-Rice codes [9, 25], while the second one uses a combination of block Gilbert-Moore, Golomb-Rice, and fixed-length codes which offers an improved compression efficiency at the expense of somewhat $(15-25\%)$ increased processing time.

3.6.1 Golomb-Rice Coding

Golomb codes [9] represent a special case of Huffman codes for sources with geometrically distributed symbols: $\Pr\{r_\theta = i\} = (1-\theta)\,\theta^i$ where $\theta \in (0,1)$. The code $G_m(i)$ consists of a series of k ones, where k is the result of a division $k = \lfloor i/m \rfloor$, followed by a 0-bit, and a $\lceil \log_2 m \rceil$-bit[1] representation of the remainder $i \mod m$. It has been shown (see [9, 6]) that an optimal for a source r_θ parameter m can be easily calculated as follows:

$$m = \lfloor -\log(1+\theta) / \log\theta \rfloor . \tag{5}$$

If this quantity is further rounded to the nearest power of 2, so that $m = 2^s$, then the divisions can be replaced by shifts, resulting in an extremely simple and fast encoding. The codes $GR_s(i) := G_{2^s}(i)$ are often called Golomb-Rice or Rice codes [25] with parameter s.

Robinson [26] has already observed that the distribution of the residual signal in LPC-based audio encoders can be closely modelled by a Laplacian (or two-sided geometric) distribution. This means that in order to apply Golomb-Rice codes one simply needs to flip the negative side of the distribution and merge it with the positive one. In MPEG-4 ALS this is accomplished by a mapping:

$$r_i^+ = \begin{cases} 2\,r_i & \text{if } r_i \geqslant 0\,, \\ -2\,r_i - 1 & \text{if } r_i < 0\,. \end{cases} \tag{6}$$

In order to estimate the optimal Golomb-Rice parameter s for a block of residuals $r_1, \ldots, r_n$, the reference implementation of MPEG-4 ALS calculates their absolute mean:

$$\mu_n = \frac{1}{n} \sum_{i=1}^{n} |r_i| \ , \tag{7}$$

[1] More efficient implementations actually use a mix of $\lfloor \log_2 m \rfloor$- and $\lceil \log_2 m \rceil$-bit codes, but it becomes unnecessary in the Golomb-Rice case, i.e. when $m = 2^s$.

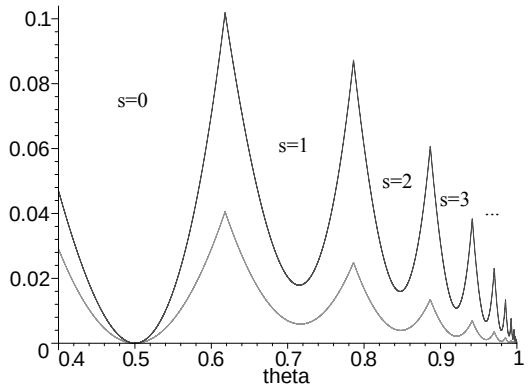

Figure 5: Redundancy $R^{GR}(x_\theta) = E\{|GR_s(x_\theta)|\} - H(x_\theta)$ and normalized redundancy $R^{GR}(x_\theta)/H(x_\theta)$ of Golomb-Rice codes for a geometric source x_θ: $\Pr\{x_\theta = i\} = (1-\theta)\theta^i$.

and then uses:

$$s = \lfloor \log_2 \mu_n + C_1 \rfloor \ , \tag{8}$$

where $C_1 \approx 0.97$ is a constant.

This estimation technique has already been used in several compression algorithms such as SHORTEN [26] or LOCO-1 [31], and it is based on the fact that sample absolute mean μ_n converges (with large n) to the first absolute moment $E\{|r|\}$, which, in turn can be converted in a parameter θ of the corresponding geometric model. Obtained in such a way parameter s is transmitted along with the encoded residuals $GR_s\left(r_1^+\right), \ldots, GR_s\left(r_n^+\right)$ (see Fig.3).

In Fig.4 we show an observed distribution of the residual, its approximation by a Laplacian distribution, and the distribution corresponding to the lengths of the resulting Golomb-Rice codes. It is clear that there is a divergence between the observed distribution and the encoded one. However, after an extensive experimental study [23] using MPEG audio sequences [12] we have found that the estimated average redundancy of Golomb-Rice encoding represents only 1.6452..% of the bitrate occupied by the residuals. In other words, given the simplicity of this coding scheme, it works remarkably well in this application.

3.7 Improved Residual Coding

The main motivations for including another residual coding scheme in MPEG-4 ALS were: a) the requirement to achieve a better compression than what is claimed by today's state of the art algorithms [12], b) realization of the fact that the efficiency of encoding of residuals is critical for the performance of the entire algorithm, and c) the fact that Golomb-Rice codes have fundamental performance limits (see [6, 30], also Fig.5). Thus, further progress cannot be achieved without using a more efficient technique.

On the other hand, looking for alternative schemes we also had to consider their complexity, so our final goal was to design an algorithm that substantially reduces the redundancy of encoding of prediction residuals while making the whole encoding/decoding process only slightly more complex [24].

In order to achieve this goal several techniques have been incorporated:

- *Higher-resolution representation of the code parameter s.*
  Similar to the previous scheme, at the beginning of each block we transmit parameter s

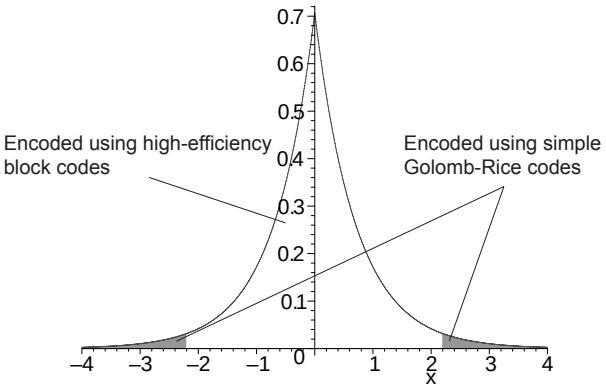

Figure 6: Partition of the residual distribution.

describing the probability distribution of the residuals. We use the same technique for estimation of this parameter, but we quantize and transmit it using higher precision.

- *Partition of the residual distribution.*
  To restrict memory usage we only construct high-efficiency block codes for a central region $[-r_{\max}, r_{\max}]$ of the residual distribution (see Fig.6). The tails are still encoded using Golomb-Rice codes as described in the previous section.
- *The use of block Gilbert-Moore codes for nested ordered distributions.*
  To encode residual values in the central region we have chosen to use block Gilbert-Moore codes [8]. The key advantages of these codes[2] come from the facts that:
  - their average redundancy rate is *decreasing with the length of an encoded block* (cf. [8, 17, 30]):
    $$R^{GM}(n) = \frac{3/2+\delta(n)}{n} < \frac{2}{n}, \tag{9}$$
    where n is a block size, and $\delta(n)$ is an oscillating function $|\delta(n)| < 1/2$.
  - they can be approximately constructed using memory-efficient arithmetic encoders [32]. The added redundancy of such an implementation with m-symbols alphabet, t-bit code registers, and τ-bit probability representations is bounded by [28]:
    $$R^{AC}(t,\tau) < m\,(\tau + \log e)\; 2^{-(t-2)} \tag{10}$$
    and can be easily controlled by properly selecting t and τ.
  - the construction of these codes is based on cumulative probabilities of symbols, making it possible not only to encode a source with a given distribution, but also *all reduced-resolution versions of this source*.

For example, given symbols $a_1, \ldots, a_m$ with non-increasing probabilities $p_1 \geqslant \ldots \geqslant p_m$, the code construction requires a table with m quantities $s_i = \sum_{j=i}^{m} p_j$. If we now quantize this source, for instance, by skipping the last bit, then the new symbols $\hat{a}_1, \ldots, \hat{a}_{\lfloor m/2 \rfloor}$ will have (also non-increasing) probabilities $\hat{p}_i = p_{2i} + p_{2i+1}$. It is clear that the new cumulative probabilities $\hat{s}_i = s_{2i}$ simply represent a *sub-set of the same table* $s_1, \ldots, s_m$. As a general rule, if Δ represents the number of skipped bits, then the encoding of such symbols can be accomplished by using probabilities $\hat{s}_i = s_{i\,2^{\Delta}}$.

[2]Sometimes these codes are also called Elias-Shannon-Fano codes [3].

- *Block-size adaptive reduction of the alphabet size.*
  In order to speed up both encoding and decoding and to further reduce memory usage we split and transmit k least significant bits (LSBs) of the residuals in central region directly[3]. The number of directly transmitted bits k is selected using:

$$k = \begin{cases} 0, & \text{if } s \leqslant B\,, \\ s - B, & \text{if } s > B\,, \end{cases} \tag{11}$$

  where s is the code parameter (8), and B is a quantity depending on the block size n:

$$B = \lfloor (\log n - 3)\,/2 \rfloor\,. \tag{12}$$

  The associated parameter Δ (the number of missing bits) used in our Gilbert-Moore encoder is obtained using:

$$\Delta = 5 - s + k\,, \tag{13}$$

  where 5 is the maximum value of B, given that MPEG4 ALS's block size $n \leqslant 2^{13}$.

It can be shown that the described choice of the parameter k limits the redundancy due to direct transmission of LSBs to approximately $0.5/n$. Choices of other parameters in our algorithm, such as $r_{\max}$, t, and τ, are also based on similar constrains, leading to only minor increase in the total redundancy of this scheme.

Overall, our implementation of the improved residual coding scheme uses only 2K words of memory to store probability tables, and involves 2 multiplications per encoded or decoded sample. Given the fact that the order of the prediction filter in MPEG-4 ALS typically fluctuates in the range of $8 - 30$, these two extra multiplications have only minor (limited to $10 - 25\%$) effect on the overall performance.

4 Experimental Results

In this section we provide the results of comparisons of the current test model for MPEG-4 ALS standard [20] with the following popular programs for lossless audio compression:

- *Shorten* [26] – current most widely used format for distribution of lossless audio files on the Internet,
- *FLAC* [5] – a gaining momentum open-source codec, and
- *Monkey's Audio* [21] – a program recognized in MPEG-4 call for proposals as current state-of-the art algorithm for lossless audio compression [12].

We used most recent versions of these programs and we run them with options providing maximum compression (`shortn32` – no flags, `flac -8`, and `mac -c4000`, correspondingly). The tests were conducted on a 1.2GHz Pentium III-M PC, with 512MB of memory. As a test material we use the standard set of audio sequences for MPEG-4 Lossless Coding [12]. It comprises of nearly 1 GB of stereo waveform files with sampling rates of 48, 96, and 192 kHz, and resolutions of 16, 20, and 24 bits. Unfortunately, not all of these formats were supported by Shorten and FLAC, which resulted in a few empty boxes in our comparison tables.

[3] This idea is similar to a more elaborate "alphabet grouping" technique [27], but our approach is much simpler because we use groups of equal size.

Format	Shorten	FLAC	Monkey's	ALS/Rice	ALS/BGMC
48 kHz / 16-bit	50.9	48.6	45.3	46.5	46.0
48 kHz / 24-bit	–	68.4	63.2	64.0	63.6
96 kHz / 16-bit	38.7	36.2	30.9	31.1	30.4
96 kHz / 24-bit	–	56.7	48.1	47.1	46.7
192 kHz / 16-bit	26.7	–	22.2	21.9	21.1
192 kHz / 24-bit	–	–	39.1	38.2	37.8
Total			**41.3**	**41.1**	**40.6**

Table 1: Average compression ratios for different audio formats.

Format	Shorten	FLAC	Monkey's	ALS/Rice	ALS/BGMC
48 kHz / 16-bit	22.3	117.2	61.6	88.6	107.2
48 kHz / 24-bit	–	197.3	79.5	120.6	112.2
96 kHz / 16-bit	66.3	226.6	136.9	158.0	206.8
96 kHz / 24-bit	–	387.7	163.3	218.4	228.8
192 kHz / 16-bit	39.8	–	97.9	116.6	148.0
192 kHz / 24-bit	–	–	108.2	177.4	195.6
Total			**647.4**	**879.6**	**998.6**

Table 2: Compression time (in seconds) measured for different audio formats.

Format	Shorten	FLAC	Monkey's	ALS/Rice	ALS/BGMC
48 kHz / 16-bit	8.4	10.3	68.4	22.6	26.5
48 kHz / 24-bit	–	21.4	72.2	23.6	26.6
96 kHz / 16-bit	20.9	26.9	134.2	32.2	45.5
96 kHz / 24-bit	–	43.9	144.2	45.7	52.8
192 kHz / 16-bit	47.0	–	104.4	24.8	33.7
192 kHz / 24-bit	–	–	113.8	51.6	53.1
Total			**637.2**	**200.5**	**238.2**

Table 3: Decompression time (in seconds) measured for different audio formats.

In the following the compression ratio is defined as

$$C = \frac{\text{Compressed File Size}}{\text{Original File Size}} \cdot 100\% \,. \tag{14}$$

The results for the examined audio formats are shown in Table 1. It is clear, that among considered programs only Monkey's Audio can compete with MPEG-ALS in terms of compression efficiency. Still, MPEG-4 ALS delivers better overall results. This gap is even wider when the improved residual coding technique (BGMC) is used.

We present measurements of the total compression and decompression times in Table 2, and Table 3 correspondingly. It can be seen that MPEG-4 ALS encoder is somewhat slower than Monkey's Audio, but one should take into account the fact that our reference implementation of the MPEG-4 ALS has not been optimized for speed. At the same time, even without any optimizations, our MPEG-4 ALS decoder is nearly 3 times(!) faster than Monkey's Audio.

References

[1] B. S. Atal and S. L. Hanauer, Speech Analysis and Synthesys by Linear Prediction of the Speech Wave, *J. Acoust. Soc. Am.* **50** (2) (1971) 637–655.

[2] A.A.M.L. Bruekers, A.W.J. Oomen, and R.J. van der Vleuten, Lossless Coding for DVD Audio, *AES 101 st Convention* (Los Angeles, November 1996).

[3] T. M. Cover and J. M. Thomas, *Elements of Information Theory* (Wiley, New York, 1991).

[4] P. Elias, Predictive coding. Part I and Part II. *IRE Trans. Inform. Theory*, **1** (1) (1955) 16–33.

[5] FLAC open-source audio compression program, `http://flac.sourceforge.net/`.

[6] R. G. Gallager, and D.C. Van Voorhis, Optimal Source Codes for Geometrically Distributed Integer Alphabets, *IEEE Trans. Inform. Theory*, **21** (3) (1975) 228–230.

[7] A. Gersho, and R. M. Gray, *Vector Quantization and Signal Compression* (Kluwer, Boston, 1992).

[8] E. N. Gilbert and E. F. Moore, Variable-Length Binary Encodings, *Bell Syst. Tech. J.*, 7 (1959) 932–967.

[9] S. W. Golomb, Run-Length Encodings, *IEEE Trans. Inform. Theory*, **12** (7) (1966) 399-401.

[10] M. C. Hans and R. W. Schafer, Lossless Compression of Digital Audio, *IEEE Signal Processing Magazine* (July 2001).

[11] ISO/IEC 14496-3:2001, *Information technology - Coding of audio-visual objects - Part 3: Audio* (International Standard, 2001).

[12] ISO/IEC JTC1 /SC29/WG11 N5040, *Call for Proposals on MPEG-4 Lossless Audio Coding* (Klagenfurt, AT, July 2002)

[13] ISO/IEC JTC1/SC29/WG11 N5576, *Analysis of Audio Lossless Coding Performance, Complexity and Architectures* (64th MPEG Meet-ing, Pattaya, Thailand, March 2003).

[14] ISO/IEC JTC1/SC29/WG11 N5718, *Working Draft of ISO/IEC 14496-3:2001/AMD 4, Audio Lossless Coding (ALS)* (65th MPEG Meeting, Trondheim, Norway, July 2003).

[15] ISO/IEC JTC1/SC29/WG11 N6012, *Working Draft 1.0 of ISO/IEC 14496-3:2001/AMD 4, Audio Lossless Coding (ALS)* (66th MPEG Meeting, Brisbane, Australia, October 2003).

[16] N. S. Jayant and P. Noll, *Digital Coding of Waveforms* (Prentice-Hall, Englewood Cliffs, NJ, 1984)

[17] R. E. Krichevsky, Universal Data Compression and Retrieval, (Kluwer, Norwell, MA, 1993).

[18] T. Liebchen, *Detailed technical description of the TUB 'lossless only' proposal for MPEG-4 Audio Lossless Coding*, ISO/IEC JTC1/SC29/WG11 M9781 (64th MPEG Meeting, Awaji, Japan 2003).

[19] T. J. Lynch, *Data Compression Techniques and Applications* (LLP, Belmont, CA 1985).

[20] MPEG4 ALS source code, `ftp://ftlabsrv.nue.tu-berlin.de/mp4lossless/`

[21] Monkey's Audio compression program, `http://www.monkeysaudio.com/`

[22] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals* (Prentice-Hall, Englewood Cliffs, NJ, 1978).

[23] Yu. A. Reznik, *Performance Limits of Codes for Prediction Residual*, ISO/IEC JTC1/SC29/WG11 M9890 (65th MPEG Meeting, Trondheim, Norway, July 2003).

[24] Yu. A. Reznik, *Proposed Core Experiment for Improving Coding of Prediction Residual*, ISO/IEC JTC1/SC29/WG11 M9893 (65th MPEG Meeting, Trondheim, Norway, July 2003).

[25] R. F. Rice, *Some practical universal noiseless coding techniques - Parts I-III*, JPL Tech. Reps. JPL-79-22 (1979), JPL-83-17 (1983), and JPL-91-3 (1991).

[26] T. Robinson, *SHORTEN: Simple Lossless and Near-Lossless Waveform Compression*, Tech. Rep. CUED/F-INFENG/TR.156 (Cambridge University, UK, December 1994).

[27] B. Ya. Ryabko and J. Astola, Fast Codes for Large Alphabet Sources and its Application to Block Encoding, *IEEE Intl. Symp. Inform. Theory* (Yokohama, Japan, 2003) 112.

[28] B. Ya. Ryabko and A. N. Fionov, Efficient Method of Adaptive Arithmetic Coding for Sources with Large Alphabets, *Probl. Inform. Transm.* **35** (1999) 95–108 (in Russian).

[29] J.R. Stuart, P.G. Craven, M.A. Gerzon, M.J. Law, and R.J. Wilson, MLP Lossless Compression, *AES 9th Regional Convention* (Tokyo, Japan, 1998).

[30] W. Szpankowski, Asymptotic Average Redundancy of Huffman (and Other) Block Codes, *IEEE Trans. Inform. Theory*, **46** (7) (2000) 2434–2443.

[31] M. J. Weinberger, G. Seroussi, and G. Sapiro, LOCO-1: A Low Complexity, Context-Based, Lossless Image Compression Algorithm, *Data Compression Conf.* (Snowbird, UT 1996) 140–149.

[32] I. H. Witten, R. Neal, J.G. Cleary, Arithmetic Coding for Data Compression, *Comm. ACM.*, 30 (6) (1987) 520–541.