Lossless Compression of Speech and Audio Signals, and Its Application

March　２０１７

Noboru Harada

Lossless Compression of Speech and Audio Signals, and Its Application

Graduate School of Systems and Information Engineering

University of Tsukuba

March ２０１７

Noboru Harada

Abstract

Speech and audio coding technologies have broad application, such as speech communication, radio and TV Broadcasting, portable music players, and storage on optical discs.

The first speech coding technologies were introduced in early 1970s and have widely been used in many applications. In order to accommodate more number of end user devices, high-efficient speech coding schemes are applied. Those coding schemes are capable of archiving compression ratios up to 12:1 and higher, by applying a psycho-acoustic model. The information that human ear is insensitive can be dropped without degrading the subjective audio quality. This type of codec is so called "perceptual lossy", or simply called "LOSSY" coding schemes.

On the other hand, some applications require perfect reconstruction. This type of codec is so called "LOSSLESS" coding schemes. Recently, high-resolution audio service is getting popular. High-resolution audio is specified as sampling frequency of 48 kHz or higher, and bit depth is 24 bit. As for the high-resolution audio, LOSSY coding scheme is of no use because high resolution audio signal contains frequency beyond that of human hearing and lossy coding schemes will remove information at the frequency. If any kind of compression is applied, it should come with no loss of information (i.e., no fidelity loss). Thus a LOSSLESS technique must be used.

The goal of this study is to provide efficient lossless coding schemes that can be used in real world application.

For lossless compression of audio signals represented in IEEE 754 floating-point, a new coding scheme, comprising Approximate Common Factor (ApxCF) coding and the Masked Lempel-Ziv (Masked-LZ) compression, is introduced. In the proposed scheme, an input sequence X is decomposed into three parts: a common multiplier A, a multiplicand sequence Y, and a difference sequence Z. Instead of re-inventing a brand new coding tool, proposed scheme makes use of existing efficient encoding tool for integer input sequences. Experimental test results using professional music recording data show that the ApxCF coding can reduce the bit rates considerably, especially when the input values in a frame are constructed by multiplication of the sequence of integer values and a floating-point constant. In addition, the Masked-LZ compression scheme has the potential to reduce bit rates of the difference mantissa. The scheme has been accepted as a part of an ISO/IEC standard, MPEG-4 Audio Lossless Coding (ALS).

For lossless compression of log-companded speech signals, the input target is ITU-T G.711 encoded sequence sampled with 8 kHz, 8 bit, 64 Kbit/s. Plus Minus Zero (PMZ) mapping is proposed for the prediction residual calculation in Mapped Domain Linear Prediction (MDLP) and Escaped-Huffman (E-Huffman) coding combined with adaptive recursive Rice coding is proposed for the prediction residual compression. It is shown that the PMZ mapping improves the compression performance by 0.2% for μ-law input. The E-Huffman coding combined with adaptive recursive Rice coding improves the compression by 0.16% averaged for all test conditions, compare to the conventional Rice coding scheme. Average computational complexity is 1.071 WMOPS for the encoder/decoder pair and the worst-case complexity is 1.667 WMOPS in total. These proposed schemes are approved as a part of ITU-T Recommendation G.711.0. The G.711.0 standard provides more than 50% average compression in service provider environments while keeping low computational complexity for the encoder/decoder pair (1.0 WMOPS average, $<$1.7 WMOPS worst case) and low memory footprint (about 5k octets RAM, 5.7k octets ROM, and 3.6k basic operators).

In addition, in order to apply those proposed coding schemes to long-term preservation and media data exchange, an archival information package format complies with the Open Archival Information System (OAIS) reference model is designed. The proposed archiving format is approved as an ISO/IEC standard: MPEG-A Professional Archival Application Format (PA-AF).

MPEG-A PA-AF is applied to archiving of recorded audio project. Two standard-compliant implementations of the PA-AF packaging and unpackaging tool are introduced. The implementations made use of MPEG-4 ALS for lossless compression of audio files and Gzip for other input files. Proposed implementation 1 is an open-source version and Proposed implementation 2 is optimized for audio archiving applications in terms of processing speed. An optimized MPEG-4 ALS codec library is applied to the implementation 2. Experimental test results show that the proposed archiving tool with the optimized implementation performs much better than widely used archiving tools such as Tar-gz, MacDMG and WinZip. Compression performance of the proposed PA-AF implementation is equivalent to or much better than other tools while keeping processing speed much faster. The devised PA-AF tool is used in commercial archiving systems in music industry.

Another example application is proposed. MPEG-A PA-AF combined with ITU-T G.711.0 is applied to archiving of speech data for telephone customer support system. In the proposed system, speech data is efficiently preserved and easily accessed for improving end user experience therefore customer satisfaction.

By applying the proposed enhancement of lossless coding schemes and the proposed package format, long-term preservation of time domain signals along with related contents and metadata has been made possible.

Abstract in Japanese

音声音響信号の高能率圧縮符号化技術は，1970 年代の電話通信のデジタル化に始まり，限られた電波資源の下で収容端末数を最大化する必要のあった携帯電話での利用など，さまざまな場所で用いられてきた．携帯音楽プレーヤー等でも，限られた記憶容量の中でできるだけ多くの楽曲を持ち歩くために，音響信号の高能率圧縮符号化技術が用いられている．これらの圧縮符号化技術は，人間の聴覚特性を考慮して，なくなっても人間に検知されにくい部分の情報を欠落させることで，主観品質を劣化させることなく音声音響信号を 1/10 ～1/20 に圧縮できる．このような符号化技術は，一旦符号化すると，複号しても完全には元の信号に戻らないことから，非可逆符号化と呼ばれる．

一方で，プロ用途の波形編集素材などでは，編集加工により原音では歪みが聞こえにくかった部分の音圧が持ち上げられて劣化が知覚される場合があるので，原音に忠実な可逆符号化技術を用いる必要がある．また，一部のプロは，自らの作品を原信号のまま近年では，サンプリング周波数 48kHz 量子化ビット数 24 ビット以上のハイレゾ音源による楽曲配信などが話題になっている．このようなハイレゾ音源などは，もともと人間の聴覚では知覚されない範囲の周波数信号を含み，非可逆の符号化技術を用いるとハイレゾを特徴づけている周波数成分が失われて，そもそもハイレゾ音源にする意味がなくなるため，圧縮符号化を行う場合には，復号後に完全に原音に戻ることが保証される可逆符号化技術を用いる必要がある．

本研究では，音声，音響信号のための可逆圧縮符号化技術とその応用について検討し，浮動小数点表現された時系列信号の可逆符号化や，対数圧伸された音声信号の可逆符号化アルゴリズムについて提案を行った．

浮動小数点表現された音響信号の可逆符号化において，入力信号はもともとは A/D 変換された際には整数信号であったものが，編集の過程で浮動小数点表現に変換されたものである場合が多いことに着目し，入力信号 X を，近似共通因数を用いて，商の整数信号系列 Y と余り系列 Z に分離して符号化する近似共通因数符号化を提案した．提案法では浮動小数点信号のための全く新しい符号化方式を再発明するのではなく，既に存在する整数信号のための高能率な可逆符号化を利用して商の整数信号系列 Y を符号化する．余り系列 Z は，新たに考案した Masked Lempel-Ziv 符号化を用いて符号化する手法を提案した．近似共通因数が存在する場合には，いかなる場合にも近似共通因数を見つけられることを人為的に生成した信号を用いた実験により示した．また，単純に浮動小数点信号を整数信号に割り当てるだけの方法と比較して，近似共通因数符号化と Masked Lempel-Ziv 符号化を組み合わせ用いる提案法が高い圧縮率を得られることを実際にプロによってレコーディングされた音楽プロジェクトを用いた評価実験により示した．同様に，文字列やバイナリデータの可逆符号化に一般的に用いられている ZIP アルゴリズムと比較した場合でも，提案法によって音声音響信号をより効率よく圧縮符号化できることを示した．提案手法は，国際標準 ISO/IEC 14496-3 MPEG-4 Audio Lossless Coding

(ALS) に採用された．

対数圧伸された音声信号の可逆符号化アルゴリズムでは，既にITU-T G.711で符号化されているサンプリング周波数 8 kHz, 8 bit, 64 Kbit/s の信号を入力とする，可逆圧縮符号化方式を提案した．Plus Minus Zero Mapping や写像領域線形予測符号化 (Mapped Domain Linear Predictive Coding)，エスケープ付きハフマン符号化，再帰 Rice 符号化などを新たに提案した．実際に PSTN 網で録音された長時間の音声信号を用いた実験により，従来手法と比較して，提案法により圧縮率が改善されることを示した．これらの提案手法は，ITU-T Recommendation G.711.0 に採用された．G.711.0 を用いることで G.711 信号をおよそ半分のサイズに可逆圧縮することができる．

上記に加え，本研究では，提案したロスレス符号化方式をマスター音源データの長期保存に応用するためのアーカイブパッケージフォーマットの設計・提案と，それらを組み合わせた楽曲アーカイブシステムの提案も行った．実際のレコーディングデータを用いた実験により，データをネットワーク上のサーバに転送するために要する時間や，LTO などの比較的転送速度の遅い記憶装置に保存するために要する時間を，圧縮しない場合と比較して半分程度に短縮できることを示した．また，提案法を用いることにより，Windows や Mac OS などの異なる OS 間で，日本語などの多バイト文字コードを用いた場合に生じる互換性の問題を解決できることも示した．提案したアーカイブパッケージフォーマットは，アーカイブのための標準規格 Open Archival Information System (OAIS) Reference model で定義された情報パッケージ (Information Package) に準拠し，国際標準 ISO/IEC 23000-6 MPEG-A Professional Archival Application Format (PA-AF) に採用されている．また，MPEG-4 ALS と MPEG-A PA-AF とを組み合わせたパッケージフォーマットは，実際に商用のマスター音源データアーカイブシステムで利用されている．さらに，提案した ITU-T G.711.0 と MPEG-A PA-AF とを組み合わせて，コールセンター用のサポートシステムに組み込んだ応用例についても示した．

本研究で提案した可逆符号化方式の拡張とアーカイブ用パッケージフォーマットを用いることにより，これまで実現されていなかった時系列信号の長期保存が可能となる．

Acknowledgements

This work has been done in research projects in NTT Communication Science Laboratories.

First, I would like to express my deepest gratitude to professors at Department of Computer Science and Department of Intelligent Interaction Technologies, Graduate School of Systems and Information Engineering, University of Tsukuba: Dr. Shoji Makino, Dr. Keisuke Kameyama, Dr. Ryuji Tokunaga, Dr. Koichi Mizutani, and Dr. Takeshi Yamada, for withstanding the enduring task for examining the draft dissertation. The advice you committed were invaluable and indispensable.

I would also like to thank Dr. Takehiro Moriya who is an NTT Research Fellow and my supervisor at NTT Communication Science Laboratories. All of the work related to this dissertation is conducted under his expert supervision.

I would like to give my grateful thank to Dr. Tilman Liebchen, Dr. Yuriy Reznik and Dr. Dai Tracy Yang, who are the contributors of ISO/IEC standard, MPEG-4 Audio Lossless Coding. Dr. Tilman Liebchen and Dr. Yuriy Reznik are the main contributors of the MPEG-4 Audio Lossless Coding (ALS) standard other than those of from NTT. Standardization of MPEG-4 ALS took us for several years. We had discussed many issues related to MPEG-4 ALS, such as issues on standardization (e.g., coding performance, source code structure of the reference software, test signals, promotion, etc.). I would also like to thank Dr. Schuyler Quackenbush who is an audio subgroup chair of ISO/IEG SC29 WG11 MPEG. He supported the standardization of MPEG-4 ALS and gave me many important comments.

I would give my thank to Dr. Michael A. Ramalho, Dr. Lei Miao, Dr. Jacek Stachurski, Dr. Hervé Taddei, Dr. Frngyan Qi, Dr. Claude Lamblin, and Dr. Yusuke Hiwasaki who are the contributors of ITU-T Recommendation G.711.0 and ITU-T Q10/SG16/WP3 rapporteurs. Dr. Michael A. Ramalho initiated and lead the standardization work for G.711.0. Without his contribution, G.711.0 could not exist. I would also like to give him a special thank for his kind help for many insightful comments reviewing this dissertation. Dr. Claude Lamblin was supportive to the G.711.0 standardization work. Dr. Yusuke Hiwasaki helped me a lot to finalize the G.711.0 related work as a colleague in NTT and as an ITU-T Q10/16 rapporteur.

I would like to thank Dr. Hendry Vincentius Tan, Dr. Houari Sabilin, Dr. Munchurl Kim, and Dr. Wo Chang who are the ad hoc group members of MPEG-A preservation formats. Dr. Hendry Vincentius Tan and Dr. Houari Sabilin helped me a lot by providing excellent contribution to the MPEG-A Professional Application Format standard. Dr. Wo Chang was always supportive to the standardization work.

I would like to thank Dr. Takeshi Norimatsu and Dr. Toshiyuki Nomura who are the active contributors to MPEG standards. I learned a lot from them to survive standardization meetings.

I would like to thank Dr. Hiroshi Sekigawa and Dr. Kiyoshi Shirayanagi. Dr. Hiroshi Sekigawa gave me a primary idea of applying continued fraction for estimating the common factor from values containing truncation error. As a result of the discussion with him, I finally succeeded to formulate the Approximate Common Factor (ApxCF) Coding algorithm. He also has given me tutorial instructions of number theory. Dr. Kiyoshi Shirayanagi helped me a lot as a group leader at the time when I was working on lossless compression of floating-point represented signals. He also gave me important comments on naming of the ApxCF Coding.

I would like to thank Mr. Csaba Kós, Mr. Koichi Sugiura, Mr. Shigeaki Sasaki and Dr. Takeshi Mori, who provided me precious supports for the standardization and for applying the standard into real world solutions. Mr. Csaba Kós and Mr. Koichi Sugiura helped me as excellent programming engineers.

I would like to give special thank to Dr. Akitoshi Kataoka. He has given me continuous encouragement to finalize this study and thesis. Finally his words made me move forward.

I would also thank colleagues in Moriya Research Lab., colleagues and ex-colleagues in other labs., Dr. Yutaka Kamamoto, Dr. Takahito Kawanishi, Dr. Hirokazu Kameoka, Dr. Takashi G. Sato, Dr. Yoshifumi Shiraki, Mr. Ryosuke Sugiura, Dr. Shinji Hayashi, Dr. Kazunori Mano, Dr. Ken'ichi Furuya, Dr. Yoichi Haneda, Mr. Kazunaga Ikeda, and Mr. Hitoshi Ohmuro. Discussions with them gave me many important and useful ideas for this study.

The music source used for performance evaluation in Sections 3.7 and 6.1.4 is originally played by New York Symphonic Ensemble and recorded by Matsushita Electric Industrial Co., Ltd. (“MEI”). Multi-channel sample master data for technology evaluation in Section 6.1.6 was provided by courtesy of Memory-Tech Corporation.

Finally, I must express my very profound gratitude to my family members, my wife Chiho, children Jun and Ryo, my parents Kenichi and Michiko, and my aunt Sadae for providing me with unfailing support and continuous encouragement throughout my years of research. This accomplishment would not have been possible without them. Thank you.

Noboru Harada

March 2017.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

Speech and audio coding technologies have broad application, such as speech communication, radio and TV Broadcasting, portable music players, and storage on optical discs. The first speech coding technologies were introduced in early 1970s and has widely been used in many applications. For example, ITU-T Recommendation G.711 Pulse Code Modulation (PCM) is standardized in 1972 [1] and widely used for narrowband telephony applications including Public Switched Telephone Network (PSTN), General Switched Telephone Network (GSTN) and packet-based network applications such as Voice Over Internet Protocol (VoIP), and has been used for many decades because of its proven voice quality, ubiquity, and utility. In late 1980s and 90s, efficient coding schemes opened the door for exploring the mobile telephone services [2, 3, 4, 5, 6]. In order to accommodate more number of end user devices, high-efficient speech coding schemes are applied. Several audio codec variants are used in portable music players so that users can carry many music files even in limited disc space [7, 8, 9, 10]. Those coding schemes are capable of archiving compression ratios up to 12:1 and higher, by applying a psychoacoustic model. Due to the masking properties of human hearing [11, 10], there is information in the acoustic signal that can be dropped with no loss of subjective audio quality. Thus this information need not be coded. A codec that leverages these human masking properties is called a "perceptually lossy", or simply called "LOSSY" coding schemes.

On the other hand, some applications require perfect reconstruction. This type of codec is so called "LOSSLESS" coding schemes [12, 13, 14, 15]. Lossy audio-compression algorithms have met with strong resistance from the fields of professional studio operations, sound archiving. A lossy compression algorithm – no matter how good – will degrade the fidelity of the signal. Recently, High-Resolution audio service is getting popular. High resolution audio is specified as sampling frequency of 48 kHz or higher, and bit depth is 24 bit. As for the high resolution audio, LOSSY coding scheme is of no use because high resolution audio signal contains frequency beyond that of human hearing and lossy coding schemes will remove information at the frequency. The high resolution folks desire no fidelity degradations other than that are already in the signal (due to imperfect recording and the like). If any kind of compression is applied, it should come with no loss of information (i.e., no fidelity loss). Thus a lossless technique must be used.

According to our interview with professional musicians and sound mixing engineers, total intermediate file size for a three minutes song becomes 2- to 4 G Bytes therefore 400 GB for an album. A recorded music project results in more than thousand files including audio tracks and other non-audio files, such as, plug-in binaries, setting notes, cover-art images and metadata files. They need to have a portable Hard Disc Drive to pass around the audio data among their client, director, or colleges. There are strong needs to compress the size of the file to store, or send them via internet to collaborate other musicians in distance and it must be a lossless compression.

Some of current music production softwares rely on the IEEE 32-bit floating-point data format [16] to process audio data and uses it as the read and write audio data format because sample values do not over-flow when mixing several audio tracks. Precision can be kept during the process. Though the IEEE floating-point representation for audio is getting more important, little work has been done on the lossless compression of IEEE floating-point audio files [17, 18, 19] since most lossless audio-coding algorithms are designed for PCM input sound formats. Yan has proposed a lossless compression schemes for IEEE floating-point audio which separates an input signal into a integer signal and an error signal; and the integer signal is compressed by using lossless compression scheme for integer signal [20, 21]. Ghido independently proposed a similar solution [22] but there are some room for improvement in both schemes.

In the real application scenario for long-term preservation of recorded audio project files, having an efficient lossless compression scheme is not enough. The application requires an information package format which can contain content information along with descriptive metadata. Information Package format that complies with the Open Archival Information System (OAIS) reference model [23] is needed.

In some cases, lossless compression of yet another input format is desired. ITU-T has standardized a lossless coding technology for G.711 [1] encoded payloads. It is called ITU-T Rec. G.711.0 [24]. The codec may be used for IP phones or conference bridge endpoints. It may also be used as a lossless compression mechanism on any intermediate link (e.g., service provider VoIP backbone links at voice gateways) where G.711 is used by the end systems. Preserving speech data on the over-phone customer support operation is another potential application of it.

1.2 The purpose of the study

The goal of this study is to provide efficient lossless coding schemes that can be used in real world application. For this purpose, applicable input formats of lossless compression needed to be extended to support IEEE 754 floating-point represented audio signal and G.711 encoded signal. Several new technologies had been devised for the purpose.

In addition, in order to apply those coding schemes to a long term preservation of recorded music project and media exchange of recorded speech data, an archival information package format complies with the Open Archival Information System Reference Model [23] is also designed and proposed. Proposed schemes and formats are applied as parts of international standards. Standardization of the proposed technologies is another important aspect of this study as well as improving the compression performance.

1.3 Organization

The organization of this dissertation is outlined in Figure 1.1. Each of those topics are addressed in the following chapters.

Chapter 2 introduces several fundamental technologies. Some of them are the fundamental basis and some others are the carrying vehicles of devised technologies which are newly proposed in this dissertation.

Chapter 3 describes the proposed lossless compression scheme for audio signals in floating point representation. Approximate common factor (ApxCF) coding and Masked-Lempel-Ziv (Masked-LZ) coding are proposed. The proposed schemes are approved as a part of an ISO/IEC standard, MPEG-4 Audio Lossless Coding (ALS).

Chapter 4 provides the proposed lossless compression scheme for log-compounded speech and audio signals. Mapped-domain Linear prediction, Escaped-Huffman coding, and Recursive Rice coding are proposed. The proposed schemes are approved as an ITU-T standard, ITU-T Recommendation G.711.0 lossless compression of G.711 pulse code modulation.

Chapter 5 shows the proposed archival package format for long-term preservation. The designed package format is approved as an ISO/IEC standard, MPEG-A professional archival application format (PA-AF).

Chapter 6 provides example applications of proposed schemes. In Section 6.1, MPEG-A PA-AF and MPEG-4 ALS are applied to archiving of recorded audio projects. In Section 6.2, MPEG-A PA-AF and ITU-T G.711.0 are applied to archiving of speech data for a telephone customer-support system.

Finally, Chapter 7 gives the conclusion of this dissertation.

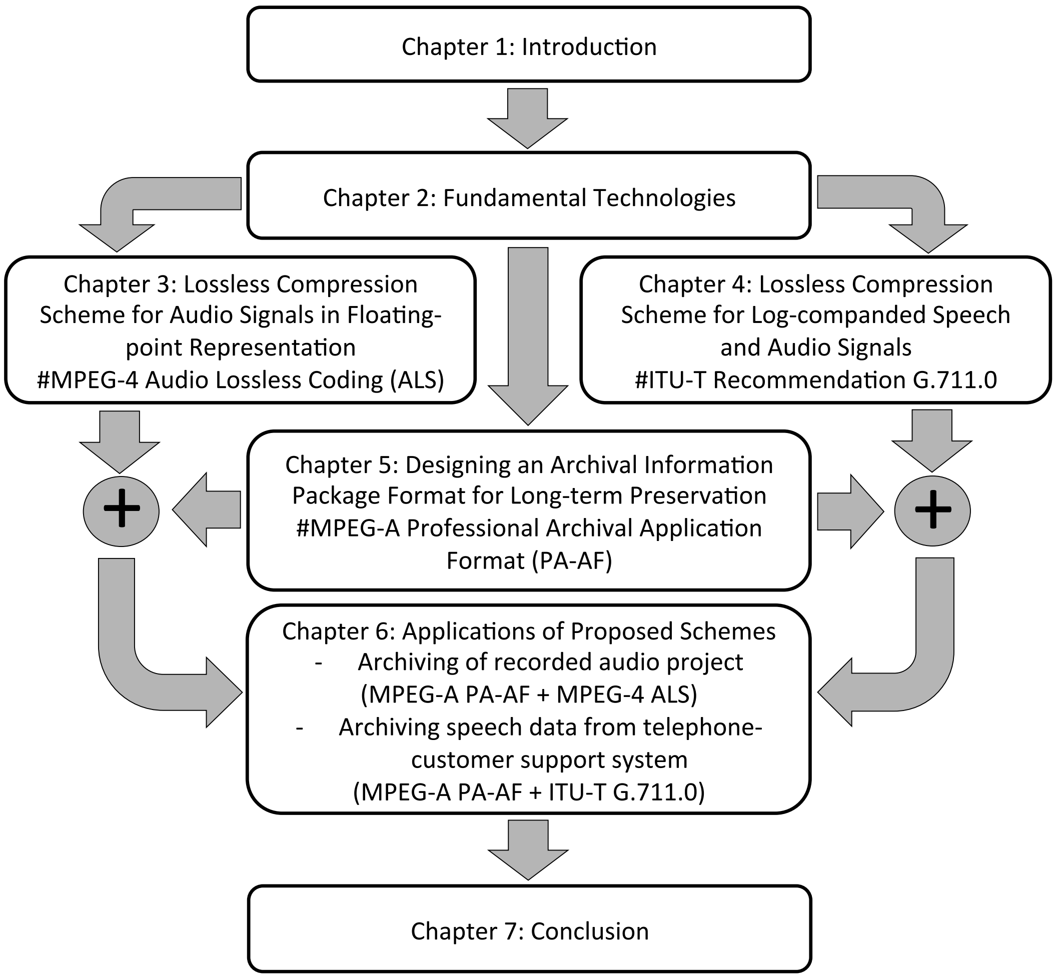

Chapter 1: Introduction
Chapter 2: Fundamental Technologies
Chapter 3: Lossless Compression Scheme for Audio Signals in Floating-point Representation #MPEG-4 Audio Lossless Coding (ALS)
Chapter 4: Lossless Compression Scheme for Log-companded Speech and Audio Signals #ITU-T Recommendation G.711.0
Chapter 5: Designing an Archival Information Package Format for Long-term Preservation #MPEG-A Professional Archival Application Format (PA-AF)
+
+
Chapter 6: Applications of Proposed Schemes
- Archiving of recorded audio project (MPEG-A PA-AF + MPEG-4 ALS)
- Archiving speech data from telephone-customer support system (MPEG-A PA-AF + ITU-T G.711.0)
Chapter 7: Conclusion

Figure 1.1: Organization of this dissertation.

CHAPTER 2

Fundamental Technologies

2.1 Introduction

This Chapter introduces several fundamental technologies that are tightly related to this study. Some of them are the fundamental basis and some others are the carrying vehicles of devised technologies which are newly proposed in this dissertation.

Section 2.2 introduces ITU-T Recommendation G.711 [1]. G.711 is the benchmark standard for narrowband telephony. It has been successful for many decades because of its proven voice quality, ubiquity, and utility. The G.711 payload is the coding target of G.711.0 [24] described in Section 2.5.

In Section 2.3, some of lossless data compression schemes are overviewed. These coding schemes are the conventional reference of the proposed technologies. At the same time, new inventions under this study have been made on top of these coding schemes as the basis. Golomb-Rice code [25, 26, 27] and Huffman code [28] are variable-length codes for entropy coding of symbols. Those codes are often used as a part of other lossless compression schemes. Lempel-Ziv-Welch (LZW) [29, 30, 31] and Deflate [32] are dictionary based compression schemes that are essential for text and binary data compression. The performance of the devised technologies newly introduced in this dissertation are compared to those conventional technologies.

In Sections 2.4 and 2.5, two international standards, MPEG-4 Audio Lossless Coding (ALS) [14, 19, 33, 34] and ITU-T Recommendation ITU-T G.711.0 (Lossless compression of G.711 pulse code modulation) are introduced. These standards are the carrying vehicle of this study. The devised technologies described in this dissertation were adopted as parts of coding tools in these two standards. Henceforward, the term tool implies a methodology, procedure or algorithm that accomplishes a specific mathematical or coding task. Many of the compression standards use this term.

Finally, in Section 5.2, an overview of the Open Archival Information System (OAIS) reference model is provided. This model is defined in another international standard, ISO 14721 [23]. It provides an abstract definition of the information package which is used in long term preservation. This standard reference model is the basis of the information package format newly designed and proposed in this dissertation.

2.2 ITU-T G.711 pulse code modulation

2.2.1 Background

In the early 1960's, an interest was expressed in encoding the analog signals in telephone networks, mainly to reduce costs in switching and multiplexing equipments and to allow the integration of communication and computing, increasing the efficiency in operation and maintenance [35]. In 1972, then the CCITT published the Recommendation G.711 that constitutes the principal reference as far as transmission systems are concerned [1, 36]. The basic principle of the algorithm is to code speech using 8 bits per sample, the input voiceband signal is sampled at 8 kHz with the telephony bandwidth of 300 to 3400 Hz. This configuration results in 64 kbit/s per each voice channel.

2.2.2 G.711 encoding

The ITU-T Recommendation G.711 [1] coding is a form of a non-linear quantization whereby individual uniform (linear) PCM samples of 13 or 14 bit precision are compressed to 8 bits using one of two logarithmic conversion laws (A-law and μ-law).

A-law:

$$c(x) = \begin{cases} \frac{A|x|}{1+ln(A)} sgn(x), & for \;\; 0 \leq \frac{|x|}{x_{max}} \leq \frac{1}{A} \\ x_{max} \frac{1+ln(A|x|/x_{max})}{1+ln(A)} sgn(x), & for \;\; \frac{1}{A} \leq \frac{|x|}{x_{max}} \leq 1 \end{cases} \tag{2.1}$$

μ-law:

$$c(x) = x_{max} \frac{ln(1+\mu|x|/x_{max})}{ln(1+\mu)} sgn(x) \tag{2.2}$$

where values $A = 87.56$ and $\mu = 255$ are used in G.711. x_{max} is the maximum value allowed as the uniform PCM input that defines the applicable value ranges of linear signal.

Those characteristics behave as linear for small amplitude signals, but are logarithmic for large signals. Followings are the expected SNRs for large signals:

$$SNR_A = 6.02B - 10.1 = 38.06dB \tag{2.3}$$

$$SNR_\mu = 6.02B - 9.99 = 38.17dB. \tag{2.4}$$

A linear-piecewise approximation for 8-bit samples is used. The approximation uses bit 1 (bit 1 is the MSB of an 8-bit samples) for sign (1 for positive, 0 for negative), bits 2 - 4 to indicate a segment, and bits 5 - 8 for level (bit 8 is the LSB of 8-bit samples). The sample format of G.711 is shown in Figure

2.1. Within each segment, the quantization is linear (4 bits or 16 levels), having 13 segments of distinct slopes for A-law and 15 segments for μ-law. For A-law, the G.711 encoded signals are obtained by inverting the even bits of the 8-bit samples. When G.711 encoded signals are transmitted serially, i.e. consecutively on one physical medium, bit No. 1 (the most significant bit) is transmitted first and No. 8 (the least significant bit) last.

The mapping between Log PCM and Linear PCM is shown in Figure 2.2. The A-law works with signals in the range from -4096 to 4096, implying in a range of 13 bits. As for μ-law, the linear signals are accepted in the range -8159 to 8159, which is represented by 14 bits. Besides this, in the dynamic range sense, A- and μ-laws are equivalent to 12 and 13 bit linear quantization, respectively.

The conversion rule for A-/μ-law from/to linear is described in terms of tables in the ITU-T Recommendation G.711 [1, 36] because there is no closed form for the compression of linear samples (although it is possible to find a closed formulae for the expansion algorithm).

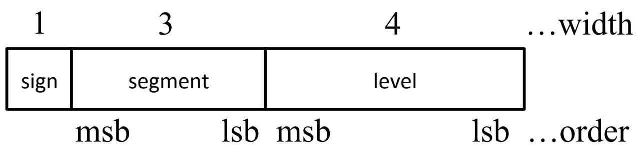


Figure 2.1: The 8-bit sample format of G.711.

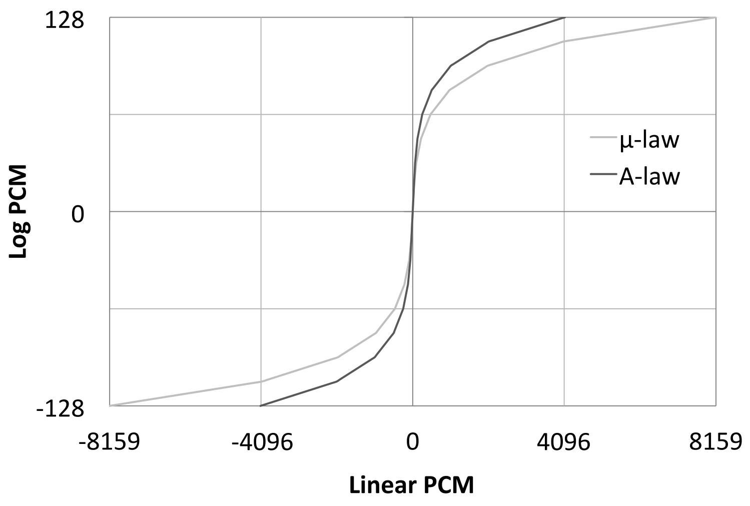


Figure 2.2: Mapping between Log PCM and Linear PCM defined in G.711.

2.3 Lossless data compression schemes

2.3.1 Background

Variable length coder (VLC) is used for statistical model [12, 37, 38]. Dictionary based coders are efficient for lossless compression of text and binary data. In the history of data compression, many Lempel-Ziv variants have been proposed [12, 29, 31, 37, 39, 40, 41, 42]. In this section, Golomb-Rice code and Huffman code are described as the basis of VLC. As for the examples of dictionary based coding schemes, Lempel-Ziv-Welch (LZW) and Deflate (LZ77 combined with Huffman code) are described. All of them are tightly related to this study.

2.3.2 Rice coding

Rice code (also known as Golomb-Rice code) [25, 26, 38] is widely used in several audio lossless compression schemes, such as MPEG-4 Audio Lossless Coding (ALS), FLAC, Shorten, etc. [14, 19, 43, 44].

The length of the Rice code of the integer $n \geq 0$ with the Rice parameter (separation parameter) S is $1 + S + \lfloor n/2^S \rfloor$ bits. This code is suitable for data where the integer $n \geq 0$ appears with a probability of $P(n)$ that satisfies $log_2 P(n) = -(1+S+n/2^S)$ or $P(N) \propto 2^{-n}$, an exponential distribution, such as the Laplace distribution [27, 37, 38, 39]. Actually, Rice code is the optimal code for the Laplace distribution. It is known that the linear prediction residual of any audio signal often follows the Laplace distribution [13].

With a given separation parameter S, a Rice code for $n \geq 0$ is computed as following steps: (1) Separate the S least significant bits (LSBs). These bits become the LSBs of the Rice code. (2) Code the remaining $k = \lfloor n/2^S \rfloor$ bits as whether k zeros followed by a 1 or k $1'$s followed by a 0 (similar to the unary code). These bits become the most significant bits (MSBs) of the Rice code. These coding steps require just a few logical operations. The computational complexity requirement of Rice coding and decoding process is rater small compared to that of other VLC algorithms such as Huffman coding, Arithmetic coding, etc. This feature is especially important for the decoder, which has to be simple and fast. Another important advantage of Rice code is that Rice coding does not require any code table because the code can be generated by mathematical calculations. Therefore, Rice codes can be generated for any large value n.

Some example Rice codes for $0 \leq n \leq 8$ and $-4 \leq n' \leq 4$ with several Rice parameters ($S = 0, 1, 2, 3$) are shown in Table 2.1. The Rice parameter S has to be signaled separately. When the Rice parameter $S = 0$, the resulting Rice code is identical to Unary code. Note that Rice code represents only integer values $n \geq 0$. Therefore, if the valuer range n' contains negative integer values, the value range has to be mapped to the integer $n \geq 0$. Alternatively, another bit can be assigned as a sign bit.

Table 2.1: Example Rice codes for $n \geq 0$ and for n' which contains negative values.

n	n'	$S = 0$	$S = 1$	$S = 2$	$S = 3$	$S = 1$ with a sign bit for n'
0	0	1	1|0	1|00	1|000	0|1|0
1	1	01	1|1	1|01	1|001	0|1|1
2	-1	001	01|0	1|10	1|010	1|1|1
3	2	0001	01|1	1|11	1|011	0|01|0
4	-2	00001	001|0	01|00	1|100	1|01|0
5	3	000001	001|1	01|01	1|101	0|01|1
6	-3	0000001	0001|0	01|10	1|110	1|01|1
7	4	00000001	0001|1	01|11	1|111	0|001|0
8	-4	000000001	00001|0	001|00	01|000	1|001|1

2.3.3 Huffman coding

Huffman coding [28, 37, 38] is a popular method for compressing data with variable-length codes. Given a set of data symbols and their frequencies of occurrence (or, equivalently, their probabilities), the method constructs a set of variable-length codewords with the shortest average length for the symbols.

A few years after Shannon and Fano published their approaches [45, 46], a student at MIT by the name of David Huffman devised a general-purpose mechanism for determining minimum-redundancy codes. The basic idea of the method is, with hindsight, extremely simple. Rather than the top-down approach of the Shannon-Fano technique, a bottom-up mechanism is employed. To begin, every symbol is assigned a codeword that is zero bits long. All of these individual symbols are considered to be in packages initially of size one; and the weights of each package is taken to be the sum of the probability weights of the symbols in the package.

At each stage of the algorithm, the two lowest probability weight packages are combined into one, and a selector bit prefixed to the codewords of all of the symbols involved: a "0" to the codes for the symbols in one of the two packages, and a "1" to the codes for the symbols in the other package. The process is then repeated, using the modified set of packages and package weights. The process terminates when the number of package becomes one. Codewords for each of the symbols in the alphabet have then been constructed.

2.3.4 Lempel-Ziv-Welch (LZW)

LZW is proposed by Terry Welch in 1984 [29, 30, 31, 39]. It is an improved coding scheme based on LZ78 [41]. In LZ78 algorithm, the coder registers encountered character strings into a dictionary and outputs a dictionary index that provides the longest matching string and a non-matching character. In contrast, LZW algorithm outputs only a dictionary index of the longest matching string. LZW initially registers all the unit characters (alphabets) into the dictionary before start coding so that LZW does not have to send any non-matching character in the code. In general, the unit character is represented in 8 bits therefore the dictionary is initialized with values of 0 to 255. The dictionary index is represented in variable number of 9 to 15 bits. Values 256 and 257 are used for a couple of special codes FLUSH_CODE and FREEZE_CODE. FLUSH_CODE is used to signal re-initialization of the dictionary. The encoder sends this code when the dictionary has become full. FREEZE_CODE is used to signal to stop adding new entries to the dictionary. There is another code called BUMP_CODE. This code is sent to increment the number of bits for the dictionary index code by one. The largest value represented in the current number of code bits are used as BUMP_CODEs (e.g., the value $2^9 - 1$ for 9 bits).

2.3.5 Deflate

Deflate is a lossless compression algorithm that makes use of a combination of LZ77 [40] and Huffman coding [28, 37, 38]. Deflate is originally designed by Philip Katz and implemented in his PKZIP software. Since then, the method is referenced in International Engineering Task Force (IETF) Informational Request For Comment (RFC) 1951 [32] and applied to ZLIB [47, 48] and GZIP [49, 50, 51]. Application of the method includes the hyper text transfer protocol (HTTP) [52, 53], the PPP compression control protocol [54, 55], and Adobe's Portable Document File (PDF).

Deflate works on a series of data blocks. Each block is compressed by using LZ77 and Huffman coding. There are two coding modes, one is fixed Huffman coding mode and the other one is dynamic Huffman coding mode.

In the fixed Huffman coding mode, three types of values, literals, offsets, and length are represented using a pair of Huffman code trees. One Huffman code tree is used for literals and lengths, and the other Huffman code tree is used for offsets. Literals in ranges of (0..255) and lengths in ranges of (3..258) are mapped into a range of values (0..287). Each value is encoded with a corresponding Huffman code. The value $0 \leq v \leq 255$ represents literals and the value 256 indicates end-of-block. Literal values of $0 \leq v \leq 143$ are encoded with a corresponding Huffman code in 8 bits, and $144 \leq v \leq 255$ are encoded in 9 bits. Length values are represented in a Huffman code plus 0 to 5 extra bits. Values $256 \leq v \leq 279$ are encoded with a corresponding Huffman code in 7 bits, and $280 \leq v \leq 287$ are encoded in 8 bits.

Offset values ranging $(1..32,768)$ are represented in 5 bits code (0..31) plus 0 to 13 extra bits. This offset range covers 32 Kbytes which is the maximum backward buffer size.

The dynamic Huffman coding mode further compress the code length of the fixed Huffman coding mode using a Huffman code.

2.4 MPEG-4 audio lossless coding (ALS)

2.4.1 Background

MPEG-4 Audio Lossless Coding (ALS) is an extension of the MPEG-4 audio coding family [14, 19, 33, 34, 56, 57, 58, 59].

In July 2002, MPEG committee issued a call for proposal for a lossless audio coding algorithm [60, 61]. By December 2002, seven organizations have submitted codecs that met the basic requirements. Finally, several organizations such as the Technical University of Berlin, Germany; RealNetworks, USA; Institute for Infocomm Research (I2R), Singapore; and Nippon Telegraph and Telephone Corporation (NTT), Japan have contributed to the standard throughout the standardization process. As the result of the final ballot in December 2005, the specifications of the lossless coding was officially established as "ISO/IEC 14496-3:2005/Amd. 2:2006, Information technology – Coding of audio-visual objects – Part 3: Audio, 3rd Edition Amendment 2: Audio Lossless Coding (ALS), new audio profiles and BSAC extensions" [14].

MPEG-4 ALS defines a simple architecture of efficient and fast lossless audio compression techniques for both professional and consumer applications. Examples for the use of lossless audio coding in general and MPEG-4 ALS in particular include both professional and consumer applications:

- Archival systems (audio archives in broadcasting, sound studios, record labels, and libraries)
- Studio operations (storage, collaborative working, digital transfer)
- High-resolution disc formats (CD, DVD, and future formats)
- Internet distribution of audio files and streaming
- Online music stores (downloading purchased music)
- Portable music players (an especially popular application)
- Lossless audio with 4K video for TV broadcasting

The MPEG-4 ALS core codec is based on forward-adaptive linear prediction, which offers remarkable compression together with low complexity. Additional features include long-term prediction, multi-channel coding, and compression of floating-point audio material [18, 19, 20, 21]. MPEG-4 ALS also offers much flexibility in terms of the compression-complexity trade-off, ranging from very low-complexity implementations to maximum compression modes, and thus adaptability to different requirements. Input formats supported by MPEG-4 ALS are shown in Table 2.2.

Table 2.2: Input formats supported by MPEG-4 ALS.

Sampling frequency	Currently tested up to 192 kHz (Higher frequency, such as 384 kHz, can be handled with the current specification).
Bit depth	PCM (up to 32 bit), IEEE754 32-bit floating-point
Channels	Up to 65536
File formats	Raw, WAVE, AIFF, BWF, BWF with RF64, Sony Wave64

2.4.2 Overview of MPEG-4 ALS encoder and decoder

The high-level block diagram of the MPEG-4 ALS encoder and decoder is shown in 2.3.

The input audio data is partitioned into frames. Within a frame, each channel can be further subdivided into blocks of audio samples for further processing (block length switching). For each block, a prediction residual is calculated using forward adaptive prediction. Linear predictive coding is used as a short-term prediction. Long-term prediction can be further applied. Inter-channel redundancy can be removed by joint channel coding, wither using difference coding of channel pairs or multi-channel coding. The remaining prediction residual is finally entropy coded. The encoder generate bitstream information allowing for random access at intervals of several frames. The encoder can also provide a CRC checksum, which the decoder may use to verify the decoded data.

2.4.3 Linear predictive coding

Linear prediction is used in many applications for speech and audio signal processing. The current sample of a time-discrete signal $x(n)$ can be approximately predicted from previous samples $x(n-i)$. The prediction is given by

$$\hat{x}(n) = -\sum_{i=1}^{P} a_i \cdot x(n-i), \tag{2.5}$$

where P is the order of the predictor. If the prediction residual

$$r(n) = x(n) - \hat{x}(n) \tag{2.6}$$

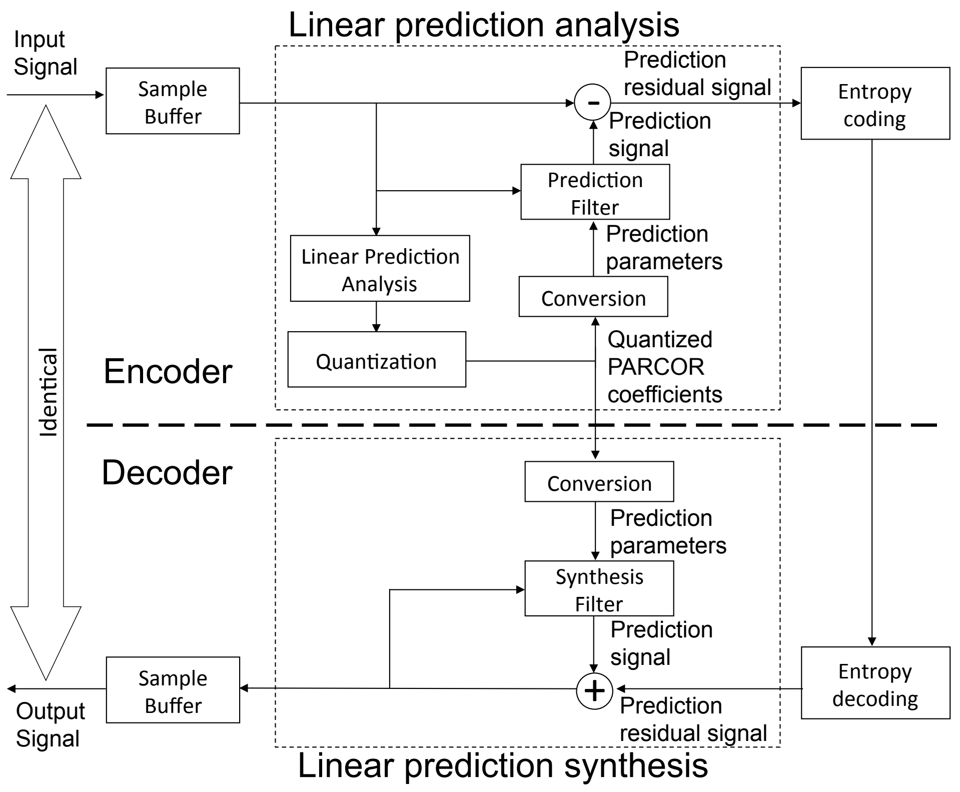


Figure 2.3: High-level block diagram of the MPEG-4 ALS encoder and decoder.

has a smaller variance than $x(n)$ itself, $r(n)$ can be encoded more efficiently. For speech or audio signals in general, total number of bit required to represent $r(n)$ with prediction coefficients a_i is smaller than that of required for $x(n)$. The linear prediction coefficients are converted into PARCOR coefficients and quantized with the form.

2.4.4 Entropy coding of prediction residual

It is known that a probability density function of the linear prediction residual signal roughly follows exponential distribution. Linear prediction residual signal $r(n)$ described in Section 2.4.3 is compressed by using one of two entropy coding schemes, Rice code (also known as Golomb-Rice code) [25, 26] described in Section 2.3.2 or the block Gilbert-Moore code (BGMC) [56, 57, 58, 62]. For each block, either all values can be encoded using the same Rice parameter (separation parameter), or the block can be further divided into four parts, each encoded with a different Rice parameter.

When a Rice parameter (separation parameter) S is given, Rice code of the residual value $r(n)$ is calculated as follows:

If $S = 0$, after $k(n)$ 0s, one 1 is presented.

$$k(n) = \begin{cases} 2r(n) & if \;\; r(n) \geq 0 \\ -2r(n) - 1 & if \;\; r(n) < 0 \end{cases} \tag{2.7}$$

For cases $S > 0$, after $k(n)$ 0s, one 1 appears. Then remainder $j(n)$ follows in S-bit representation.

$$k(n) = \begin{cases} \lfloor 2^{-(S-1)} r(n) \rfloor & if \;\; r(n) \geq 0 \\ \lfloor 2^{-(S-1)} \left(-r(n) - 1\right) \rfloor & if \;\; r(n) < 0 \end{cases} \tag{2.8}$$

$$j(n) = \begin{cases} r(n) \circledast \left(2^{(S-1)} - 1\right) + 2^{(S-1)} & if \;\; r(n) \geq 0 \\ (-r(n) - 1) \circledast \left(2^{(S-1)} - 1\right) & if \;\; r(n) < 0 \end{cases} \tag{2.9}$$

where $\circledast$ denotes an AND bit-operator.

2.4.5 Other features

In this dissertation, some additional coding tools for supporting IEEE 754 floating-point represented audio signal are newly proposed and integrated into the MPEG-4 ALS standard.

2.5 ITU-T G.711.0: lossless compression of G.711 pulse code modulation

2.5.1 Background

The ITU-T Recommendation G.711 [1] is the benchmark coding standard for narrowband telephony for many decades. Owing to its proven voice quality, ubiquity and utility, G.711 continues to enjoy widespread use in today's newest packet-based networks, e.g., Voice over Internet Protocol (VoIP) – even when neither endpoint interfaces to a telephony network. The ITU-T has recently established a lossless coding standard for G.711 payloads typically used in VoIP applications. This standard is ITU-T Recommendation G.711.0 [24]. The G.711.0 codec may be used as a traditional codec and its use negotiated (end-to-end) by the end terminals (IP phones, conference bridge endpoints, etc.). Additionally, owing to its lossless and stateless design, G.711.0 may also be used as a lossless compression mechanism on any intermediate link (e.g., service provider VoIP backbone links at voice gateways) where G.711 is used by the end systems. G.711.0 employed in these transcoding applications provides bandwidth savings with no degradation in audio quality relative to G.711 since it is a lossless algorithm. For these gateway applications, low computational complexity is desired. A Figure of Merit (FoM), defined in the G.711.0 Terms of Reference (ToR), was used to assess the tradeoff between complexity and signal compression during the design phase and the G.711.0 selection process [63].

2.5.2 Overview of G.711.0 encoding

The G.711.0 codec accommodates both G.711 encoding laws (A-law and μ-law) and losslessly compresses frames consisting of 40, 80, 160, 240 or 320 G.711 samples. G.711.0 is lossless for all possible G.711 payloads and is most effective when compressing zero mean acoustic signals such as speech. Owing to its stateless and self-describing design (all information needed to reconstruct an original G.711 frame is contained in the G.711.0 compressed frame), the input frame lengths may be changed on-the-fly at the encoder. The algorithmic delay of G.711.0 is defined by the input frame length and is therefore, 5, 10, 20, 30, and 40 ms (assuming the usual 8 kHz sampling).

G.711.0 is a variable bit rate compression algorithm; the size of the (compressed) output frame depends on the input signal characteristics. The minimum size of an encoded frame is one byte. The maximum size of an encoded frame is the input frame size plus one byte which occurs when the input frame cannot be compressed by any of the available encoding tools.

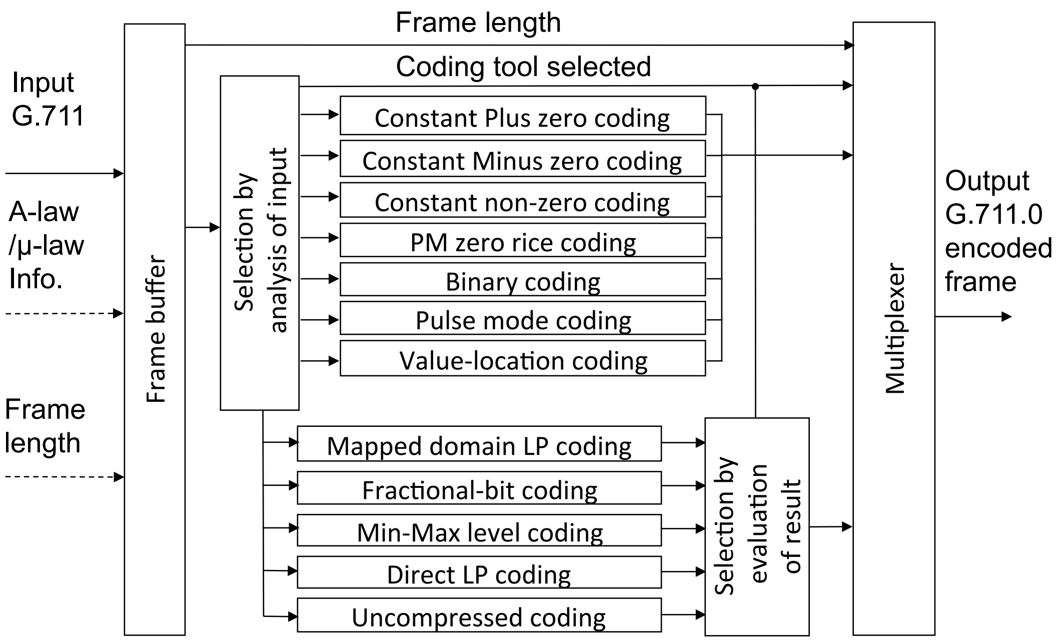


Figure 2.4: High-level block diagram of the G.711.0 encoder.

Figure 2.4 shows the high-level block diagram of the G.711.0 encoder. The encoder selects among one of the coding tools shown in Figure 2.4 to create the G.711.0 encoded output frame [64, 65, 66, 67, 68, 69, 70, 71]. Brief description of tools are as follows:

- Uncompressed coding tool

  If all tools fail to compress an input frame, the encoder produces a one-byte prefix for an uncompressed frame and simply reproduces the original G.711 bitstream (i.e., the input frame). In this case, the encoded bitstream size is equal to the input data size plus one byte.

- Constant value coding tool

  When all sample values in an input frame are the same, one of the constant value coding tools is applied. Constant plus zero values 0^+, such as 0xd5 for A-law and 0xff for μ-law, and constant minus zero values 0^-, such as 0x55 for A-law and 0x7f for μ-law, are signaled with a one-byte prefix and there are no trailing bytes in these cases. Those minimum magnitude values (0^+ and 0^-) are treated specially because they are often observed when the input signal is silence. Constant values other than those above are signaled by a one-byte prefix code followed by the actual constant value in one byte (two bytes total).

- Plus-Minus zero Rice coding tool

  When all sample values of an input frame are either plus zero 0^+ or minus zero 0^-, the encoder tries the Plus-Minus (PM) zero Rice coding tool. The case occurs when the input signal is silence but both the plus and minus minimum magnitude values (0^+ and 0^-) are observed in the frame. First of all, this coding tool counts the number of existing 0^+ and 0^- samples and detects which value has more occurrences. The value is called more zero 0_m. If numbers of occurrences of 0^+ and 0^- are the same, 0_m is set to the value 0^+. Then, the tool converts the input samples into sequential values of number of running more zero 0_m values followed by a less zero value 0_l. Finally, the tool encodes the sequence of the numbers using Rice coding with the best Rice parameter value for the frame. The Rice parameter is Huffman encoded.

- Binary coding tool

  When all sample values in an input frame are either 0^+ or 0^-, and if the PM zero Rice coding tool does not reduce the encoded data size, the Binary coding tool is applied. This tool generates a one-byte prefix code followed by the input sample values which are converted into one bit per sample. In the generated code, 0 indicates 0^+ and 1 indicates 0^-.

- Pulse mode coding tool

  When all sample values in a given input frame are either 0^+ or 0^- except one sample (the input frame is almost silence), the encoder applies the Pulse mode coding tool. After the position and the value of the non zero sample (called pulse) are stored, the sample values are encoded applying the same scheme as Plus-Minus zero Rice coding with considering the pulse sample as more zero 0_m. Huffman coding is applied to the Rice parameter. The pulse position index is binary encoded and the pulse value is differentially coded by Rice coding with Rice parameter 0, based on smaller difference from 0_m or 0_l value which is signaled by 1 bit.

- Value-Location coding tool

  When the input frame is a low-level signal (the sample value that occurs most in an input frame is zero and the frame size and range of the remaining values satisfy a specified criteria [24]), the encoder uses the value-location coding tool. The tool sequentially encodes positions of all values within an input frame that differ from the reference 0 value.

The method effectively decomposes an input frame s as:

$$s = \sum_{k=1}^{L-1} v_k c_k \tag{2.10}$$

where v_k represents the number of non-zero values within s, L is the number of such values, and vectors c_k encode the locations at which a particular v_k occurs (they contain ones at locations where v_k occurs and zeros elsewhere). To enhance coding efficiency, the vectors c_k are encoded sequentially. First, the value locations that have already been encoded in all previous code vectors c_k, $i < k$, are removed from the current c_k vector. The reduced-dimensionality vector c'_k is then coded using Rice coding, binary encoding, or explicit location encoding. The information about the c_k encoding sequence, the encoding method for each c_k, and the corresponding values v_k are transmitted in the bit-stream.

- Mapped Domain Linear Predictive coding tool

  The Mapped domain LP coding tool takes a sequence of N G.711 A-law or μ-law symbols. First, these N G.711 symbols are converted into uniform (linear) PCM domain and a short-term prediction is carried out using LP analysis with progressive linear prediction for the first few samples in the frame. The prediction residual signal lies in the range of $[-255, 255]$ since the predicted value is subtracted from the target value in the 8-bit logarithmic domain (not in the uniform PCM domain). The LPC parameters are quantized as PARCOR coefficients. Additional coding tools such as Bandwidth extension, Long term prediction and Plus-Minus zero mapping tools are further applied depending on the input samples and the frame lengths. The amplitude of the residual signal is calculated and coded in sub-frames using either Rice coding or Escaped-Huffman coding with Adaptive recursive Rice coding.

- Fractional-bit coding tool

  The fractional-bit coding tool identifies the total number of signal levels that exist within an input frame and then combines several samples for joint encoding. Five samples are used at a time to calculate the polynomial:

$$V = l_1 + l_2 L + l_3 L^2 + l_4 L^3 + l_5 L^4 \tag{2.11}$$

  where l_1 represents the value of sample i, and L represents the number of levels within an input frame. Bit-rate saving is achieved by binary encoding the polynomial V (resulting in fractional-bit per sample) instead of encoding each sample individually. Several level-distribution cases that benefit most from this coding approach are identified. In the bit-stream, one of 30 states of the one-byte header prefix code indicates that the fractional-bit coding is applied and specifies the encoded input frame characteristics (frame length and levels present).

- Min-Max level coding tool

  The Min-Max level coding tool is used only for 40-sample frames. This tool calculates the minimum number of bits needed to encode the binary span of the G.711 amplitude levels represented in the input frame and then encodes each G.711 sample with precisely that number of bits per sample. Pre-appended to this encoded sample data is one, or on rare occasion two, additional bytes of overhead information specific to this tool. Lastly, pre-appended to this information is the one byte prefix code described earlier.

- Direct Linear Predictive coding tool

  For 40-sample frames, the Direct Linear Predictive coding tool is used when all the above coding tools fail to compress the input frame. It performs a 4th order LP coding directly in the 8-bit logarithmic domain on absolute values of the input samples. All prediction coefficients are fixed at 0.25. The prediction residual is encoded by Rice coding with the Rice parameter of 5 along with the sign bit of the input sample.

A “prefix code” and the encoded information are sent as a part of the G.711.0 bitstream output. The prefix code defines the frame length and contains information related to the selected encoding tool. Table 2.3 shows the list of supported frame length and corresponding prefix codes. Table 2.4 shows the list of encoding tools and corresponding tool prefix codes.

The G.711.0 decoder reads the prefix code and then presents the remainder of the encoded data (plus any side data contained in the prefix code) to the appropriate decoding tool. The decoder generate 0 samples when the prefix code is 0x00 which indicates "0 sample frame". This code can be used as padding octets.

Table 2.3: List of supported frame lengths and corresponding tool prefix codes.

Frame length (N samples)	Prefix code (in binary representation)
0	0000 0000
40	01-- ----
80	10-- ----
160	11-- ----
240	0010 ----
320	0011 ----

Table 2.4: List of encoding tools and corresponding tool prefix codes.

Tool type		Prefix code for ITU-T G.711.0 encoded frame (initial bits, in binary representation)	
		$N = 40, 80, 160$	$N = 240, 320$
Uncompressed coding		--00 0000	---- 0000
Constant coding tools	Constant plus zero coding	--00 0001	---- 0001
	Constant minus zero coding	--00 0010	---- 0010
	Constant non-zero coding	--00 0011	---- 0011
Plus Minus (PM) zero only	Binary coding	--00 0100	---- 0100
	PM zero Rice coding (Minux $\leq$ Plus)	--01 0ss	---- 100s s
	PM zero Rice coding (Minus $>$ Plus)	--01 1ss	---- 101s s
Plus Minus (PM) zero only except one sample	Pulse mode coding (Minus $<$ Plus)	--01 000	---- 1000 0
	Pulse mode coding (Minus $>$ Plus)	--01 100	---- 1010 0
Value-location coding		--00 0110	---- 0101
Mapped domain LP coding	LTP: enabled (Only for $N \geq 160$)	1100 1	---- 011
	LTP: disabled	--1	---- 11
Fractional bit coding		0000 0010 to 0001 1111	
Min-Max level coding (Only for $N = 40$)		0100 0101	N/A
Direct LP coding (Only for $N = 40$)		0100 1	N/A

Note 1: The first bits "--" (for $N = 40, 80, 160$) or "----" (for $N = 240, 320$) are the prefix codes representing the frame length, given in Table 2.3.

Note 2: For PM zero Rice coding, following two bits after the prefix code $ss! = 00$.

2.6 Open archival information system (OAIS) reference model

2.6.1 Background

Many organizations in various domains are planning or actually managing long-term preservation of information. For example, museums, libraries, TV and radio broadcasters, movie and music industries, governments, companies, individual people are preserving information in different levels. Especially in recent years, preserving digital-born information is getting more important.

There are several difficulties with long-term preservation of digital information. Rapid improvement of information technology causes rapid changes of environments required for accessing the preserved information. For example, digital information needs to be kept fully accessible. 5.25-inch floppy disks were widely used several decades ago but in these days it is very difficult to find any 5.25-inch floppy disk drive that can be connected to the latest PCs. Without having a compatible playback device, the information stored in the floppy disk is hardly accessible. Even though the floppy disk can be accessed, data format may be obsoleted and the information may not be retrieved without any compatible rendering software if the data format is specific to the proprietary software.

Long-term preservation of digital information is tightly related to and highly depending on its access environments (e.g., computer and operating systems, storage devices, data structure and formats, rendering software, etc.). If any piece of these components is missing, proper access of the information fails. In other words, preserving digital information for long-term means keeping the information accessible for long enough to be concerned about changing technologies. Therefore, not only the digital information itself but also playback devices, computer, operating system, and software should also be preserved. Alternatively, data format of the information should be migrated into another accessible format.

The trial for long-term preservation requires careful preservation planning and could be quite expensive. Some times, important lessons can be learned from the best practices. Sharing the knowledge, preservation system design, actual system and storage is the important key for reducing the operating cost of preservation. However, the term "archive" has different meaning to the different community because of the historical or cultural differences among those preservation entities.

The Consultative Committee for Space Data Systems (CCSDS) has established standardization of Open Archival Information System (OAIS) reference model. The reference model was approved as an ISO standard, ISO

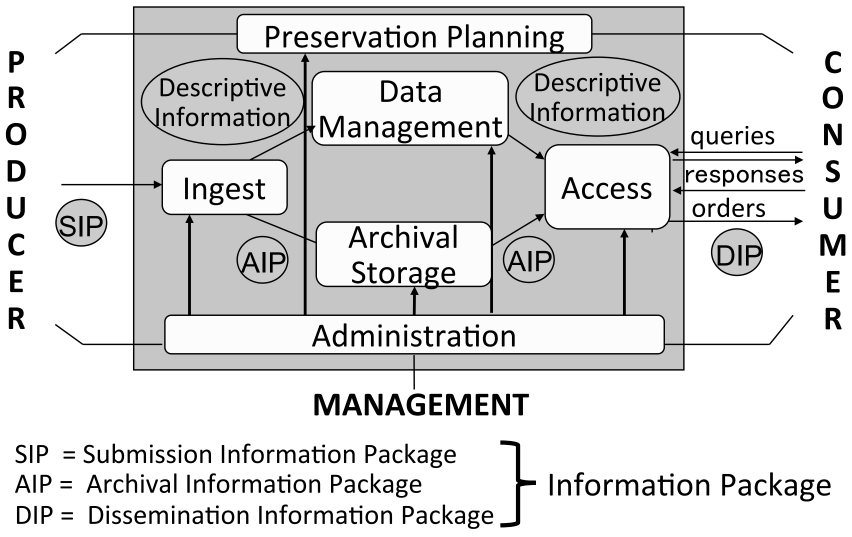


Figure 2.5: Function entities and related interfaces defined in the OAIS reference model.

14721:2003 [23], Space data and information transfer systems – Open archival information system (OAIS) – Reference model. The latest edition (2nd edition) has been published in 2012. This OAIS reference model provides an abstract concept of Archiving system and defines common terms, long-term information preservation and access functions, and roles required in the archive. So that different communities can discuss issues using defined common terms and concepts.

2.6.2 Open archival information system (OAIS) reference model

The Open Archival Information System (OAIS) reference model [23] is a framework for understanding and applying concepts necessary for the long-term preservation of digital information (where long term is long enough to be concerned about changing technologies). The reference model addresses a full range of archival information preservation functions including ingest, archival storage, data management, access, and dissemination. It defines a minimal set of responsibilities that must be met for an archive to be called an OAIS, and it also defines a maximal archive to provide a broad set of useful terms and concepts.

Figure 2.5 shows the function entities and related interfaces defined by the OAIS reference model.

There are three external environmental roles and the OAIS reference model

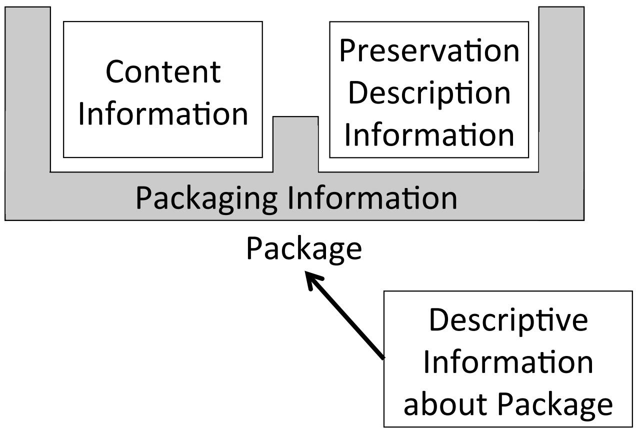

Content Information
Preservation Description Information
Packaging Information
Package
Descriptive Information about Package

Figure 2.6: Information Package Concepts.

characterizes the interface between these entities and the OAIS archive. Management is the role which defines the high level policies that govern the OAIS archive. Producer is the role that generates content information to be preserved in an OAIS archive and ingests the content information to the OAIS archive. Producer could be the individual person, organization, or information generation system that provides content information to the OAIS archive. Consumer is the role of person or system that acquires the preserved information of interest. Some archives are designed for the Designated Community. The Designated Community is defined as the group of specific consumers who can understand the preserved information.

The standard defines six functional entities: Ingest, Archival Storage, Data Management, Administration, Preservation Planning, and Access.

The standard also defines three information packages: Submission Information Package (SIP), Archival Information Package (AIP), and Dissemination Information Package (DIP). Producer uses SIP to ingest content information to the OAIS archive, and the OAIS stores content information in the form of AIP. The Access function of the OAIS archive disseminates the content information to Consumer using DIP. Any transmission of content information between Producer, Consumer and the OAIS archive, Information Package is used. To interconnect OAIS reference model compliant archiving systems, the interface point must be aligned at a certain level. The interface point might be defined as Information Packages.

Figure 2.6 shows the concept of Information Package. An Information Package is defined as a conceptual container of Content Information and Preservation Description Information (PDI). The Content Information is the original target of preservation which consists of Content Data Object (physical or digital object) and associated Representation Information. Representation Information is the information which makes Content Data Object understandable to the Designated Community. Preservation Description Information is decomposed into five types of information: Provenance, Context Reference, Fixity, and Access Rights.

2.7 Summary

This Chapter introduces several fundamental technologies that are tightly related to this study. Some of them are the fundamental basis of the study and some others are the carrying vehicles of devised technologies which are newly proposed in this dissertation.

Chapter 3

Lossless Compression Scheme for Audio Signals in Floating-point Representation

3.1 Introduction

Lossless compression for the IEEE 754 floating-point [16] audio data has several applications but little work has been done since most of lossless audio-coding algorithms are designed for PCM input sound formats.

One conventional scheme makes use of float-to-integer mapping [20, 21, 22]. The scheme is assuming that the signal is originally fed from an Analog to Digital (A/D) converter, therefore it can be expected that the dynamic range of the signal is not much wider than the original integer signal. This means the signal was originally represented as an integer sequence. During mixing or editing process, it was converted into floating-point file format. This conventional scheme can compress floating-point data efficiently, but there is some room for improvement of the compression performance.

In this chapter, a new lossless compression scheme is proposed to improve compression performance for the IEEE 754 floating-point audio data. The following ideas are introduced on top of the basic architecture of decomposition of floating point data into integer and remaining difference mantissas.

- Approximate Common Factor (ApxCF) Coding for the pre-processing of the floating-point data compression.
- Masked Lempel-Ziv (Masked-LZ) compression with masked bit comparison for the compression of remaining difference mantissas.

The proposed scheme has been accepted as a part of an ISO/IEC standard, MPEG-4 Audio Lossless Coding (ALS).

Section 3.2 overviews the IEEE 754 floating-point representation and its arithmetic. Section 3.3 provides an overview of encoding. Section 3.4 shows how to estimate ApxCF value. Detailed description of Masked-LZ compression is shown in Section 3.5. An over view of decoding is described in Section 3.6.

Experimental results are shown in Section 3.7. Finally, Section 3.8 summarizes this chapter.

3.2 IEEE 754 floating-point format and arithmetic

In this section, an overview of IEEE floating-point format defined in IEEE 754 [16] is given. The IEEE 754 standard defines four floating-point formats in two groups, basic and extended, each having two widths, single and double. In this session, only basic formats will be considered.

Figure 3.1 and Table 3.1 show the 32 bit single floating-point representation format defined in IEEE 754 [16]. Where, s denotes sign bit, e is exponent which is normalized with E_{bias}, f represents mantissa part, f is represented in normalized 23 bits, $b_1, b_2, \cdots, b_{23}$. The least significant bit b_{23} is called "unit in the last place" (ulp).

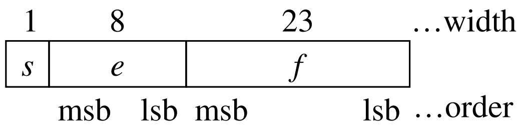


Figure 3.1: IEEE 754 32-bit single floating point format.

Table 3.1: Summary of format parameters in IEEE 754 32-bit single floating-point format.

Parameter	Width in bits	Possible Values
Sign s	1	0 or 1
Biassed Exponent e	8	0 to 255
Mantissa f	23	$.b_1 b_2 \cdots b_{23}$
Total width in bits	32	

In the 32 bit floating-point representation, number of bits assigned to the mantissa is 23 bits and the actual precision is 24 bits when taking a hidden bit into account. Note that mantissa bits are normalized to have the MSB as 1 so that the MSB can be omitted from the format representation. The MSB of mantissa is so called a hidden bit. The E_{bias} value is +127, the range of

Table 3.2: Possible values of 32-bit single floating-point format.

Condition	Value
If $e = 255$ and $f \neq 0$	$x = NaN$
If $e = 255$ and $f = 0$	$x = (-1)^s \cdot \infty$
If $0 < e < 255$	$x = (-1)^s \cdot (1.f) \times 2^{e-127}$
If $e = 0$ and $f \neq 0$	$x = (-1)^s \cdot (0.f) \times 2^{-126}$ (de-normalized numbers)
If $e = 0$ and $f = 0$	$x = (-1)^s \cdot 0$ (zero)

value of the exponent is $-127 \leq e - E_{bias} \leq 128$. The range of possible x values is shown in Table 3.2. Where, x denotes a floating-point represented sample value. The value $x = 0$ can be represented by $e = 0$ and $f = 0$ which is Not a Number (NaN).

Especially when $0 < e < 255$, in other words when $-126 \leq e - E_{bias} \leq 127$, a floating point value of x can be formulated as follow:

$$x = (-1)^{s_x} \cdot (1.f_x) \times 2^{e_x - E_{bias}}. \tag{3.1}$$

In the IEEE 754 floating point arithmetic, multiplication is implemented as an accumulation of shifted values of mantissa $1.f_x$ in multiplicand x according to the multiplier y [72, 73, 74].

Number of bit shift is relative to the mantissa bits $1.f_y$ $(= 1.b_{y1}, b_{y2}, \cdots, b_{y23})$ of a multiplier y.

$$x \times y = (-1)^{(s_x+s_y)} \times \left\{ (1.f_x) + \sum_{i=1}^{23} b_{yi} \cdot (1/2)^i \times (1.f_x) \right\} \times 2^{(e_x+e_y-E_{bias})} \tag{3.2}$$

$$x \times y = (-1)^{(s_x+s_y)} \times m \times 2^{(e_x+e_y-E_{bias})} \tag{3.3}$$

$$m = \left\{ (1.f_x) + \sum_{i=1}^{23} b_{yi} \cdot (1/2)^i \times (1.f_x) \right\} \tag{3.4}$$

Since $1.f_x$ and $1.f_y$ are normalized as $[1.0, 2.0)$, the resulting value range of the mantissa m should be $[1.0, 4.0)$. When the resulting mantissa is $m \geq 2.0$, m will be normalized to be $1.f \times 2$. Therefore, the exponent value shall be increased by 1.

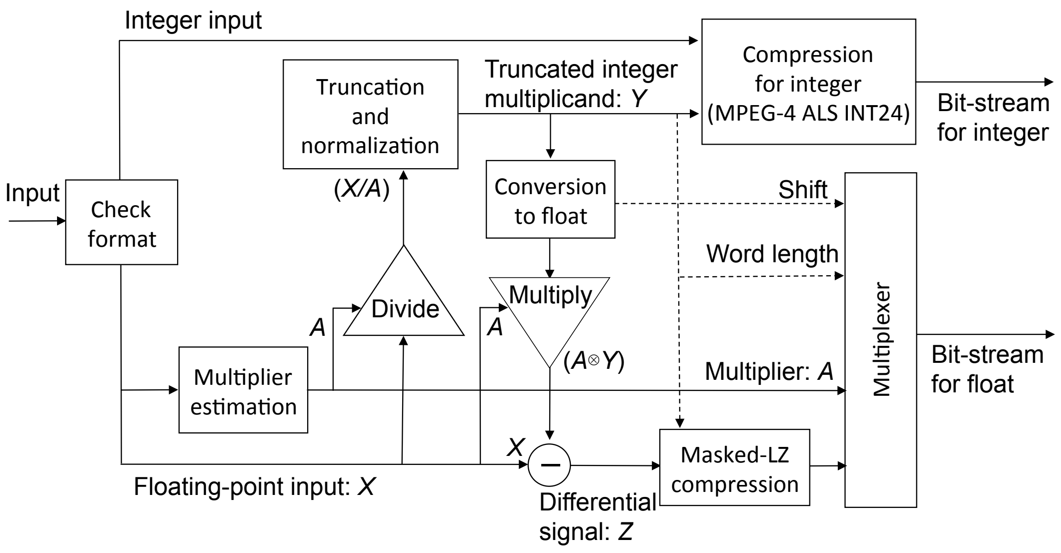


Figure 3.2: Block diagram for the encoder.

The precision of the intermediate multiplication result in the mantissa m will be 47 or 48 bits including a hidden bit. One of following rounding mode will be applied at the *ulp* in order to make the result fits into the limited precision: Round to nearest when tie to even, Round toward $+\infty$, Round toward $-\infty$, and Round toward 0.

3.3 Overview of encoding

Figure 3.2 shows the integrated lossless encoder for integer and floating-point data. In the proposed encoding scheme for 32-bit IEEE 754 floating-point data, an input sequence X is decomposed into three parts: a common multiplier A, a multiplicand sequence Y, and a difference sequence Z. X and Z are vectors containing floating-point values. Y is a sequence of truncated integers and A is a scalar floating-point number.

With the definition above, X is represented as follow:

$$X = A \otimes Y + Z \tag{3.5}$$

where $\otimes$ denotes multiplication with rounding. Any of rounding mode can be chosen if the same rounding mode is applied at the encoder and decoder. In this study, "Round to nearest when tie to even" is used.

The common multiplier A can be represented as

$$A = 1.f_A \times 2^{(shift)} \tag{3.6}$$

where the range of the mantissa part $1.f_A$ is $1.0 \leq 1.f_A < 2.0$. The exponent part $2^{(shift)}$ affects only exponent of multiplicand. When mantissa part $1.f_A = 1.0$, multiplying A to a multiplicand y can be regarded as a decimal alignment by applying a bit-wise shift for separating a floating point represented value x into an integer value y, and a fractional remainder z as follows:

$$y = \left\lfloor \frac{x}{2^{(shift)}} \right\rfloor, \tag{3.7}$$

$$z = \frac{x}{2^{(shift)}} - \left\lfloor \frac{x}{2^{(shift)}} \right\rfloor. \tag{3.8}$$

First, the common multiplier A is estimated by analyzing the input signal vector X in a current frame using rational approximation of ApxCF. The common multiplier A is normalized to $1.0 \leq A < 2.0$. If an appropriate value of ApxCF can not be found in the multiplier estimation module, the common multiplier A is set to 1.0. Then, the magnitude of the floating-point number $(x \oslash A)$ is truncated to the largest integer number y where $\oslash$ denotes division with rounding. Here, y is carefully chosen to minimize $z = x-(y\otimes A)$, and the magnitude of $(y\otimes A)$ must be equal to or less than x where x, y, and z are used as one sample from sequence vector X, Y, and Z. The truncated multiplicand y is set to 0 when x is the NaN (special number) or x is the denormalized number. The value of y is also set to 0 (0 is set to y) if more than 23 bits are needed for representing the differential of mantissa z $(= x - (y \otimes A))$.

All 32 bits of x are copied to z if the value of y equals 0. Otherwise, the residual z $(= x - (y \otimes A))$ is computed. If exponents of x and $y \otimes A$ are different, the mantissa bits of x are left-shifted to align exponents, and the mantissa of $y \otimes A$ is subtracted from shifted mantissa of x. Since y is chosen very carefully, the mantissa bits of z do not exceed 23 bits. When more than 23 bits are needed in order to represent the difference signal, the truncated multiplicand y is set to 0, and all 32 bits of data x are coded separately.

Note if the common multiplier A equals 1.0, the floating-point number $(x\oslash A)$ equals x. In that case, y is the largest integer number whose magnitude is less than that of x. Which means the input signal X is directly truncated into integer Y and the necessary bit of the difference mantissa is uniquely determined by corresponding truncated integer y.

Then, the 23 or fewer bits of the difference-mantissa signal Z are encoded using the Masked-LZ compression module except when y equals 0. If the integer y equals 0, all 32 bits of data x are coded with the module separately. The word length can be uniquely determined by the corresponding truncated integer.

In all cases, the MPEG-4 ALS compression module for the integer input is utilized for compressing the truncated integer sequence Y. The Masked-

LZ module takes the word length into account in compressing the difference mantissa signal.

The encoding scheme contains the following seven steps.

Step 1: Estimate the common multiplier A, which is normalized to $1.0 \leq 1.f_A < 2.0$, by analyzing the input signal X in the multiplier estimation module in every frame using a rational approximation of ApxCF (Approximate Common Factor). When an appropriate ApxCF can not be found, $1.f_A$ is set to 1.0.

Step 2: Compute the truncated integer multiplicand sequence Y from X and the mantissa part $1.f_A$ of the estimated common multiplier A. Detect y_i and an exponent value $shift$ of A with a restriction toward that the sign of z_i is to be the same as that of x_i and y_i, and the largest absolute value of y_i does not exceed the maximum input value allowed by the integer encoding tool. Where x_i, y_i and z_i are the ith sample value of X, Y and Z respectively.

Step 3: Compute the difference signal sequence Z $(= X - (A \otimes Y))$.

Step 4: Reorder the difference signal sequence Z for the Mask-LZ compression. Samples needed to code all 32 bits of the mantissa are moved to the top ($PartA$), and other samples are on the latter part ($PartB$).

Step 5: Align the necessary bits of mantissa Z and separate into characters with word size of 8 bits.

Step 6: Encode characters from differential mantissa Z in $PartA$ and $PartB$ using the Masked-LZ compression module.

Step 7: Encode Y, and pack it together with the mantissa bits of A and the encoded bit stream of Z.

3.4 Estimating approximate common factor

This section explains the fundamentals of proposed Approximate Common Factor (ApxCF) Coding. The assumption is that a gain adjustment factor A was multiplied to the original integer sequence $\hat{Y}$ on the course of some editing process therefore $\hat{Y}$ was transformed into an observed floating point sequence X. This observed sequence X should have a common divisor A. Detecting the common multiplier, or common divisor, A, is not straight forward. So far, a reasonable estimation procedure of the ApxCF using a rational approximation [75] has been devised. The input signal X is analyzed every frame. The estimation is like a finding the Greatest Common Devisor (GCD) with the condition that the input signal X may have an error at the "unit in the last place" (ulp) due to rounding-off or chopping for truncation. The range of the error is between $[-1/2 \times ulp, +1/2 \times ulp]$. The relationship among estimated value of A, y_i and x_i is expressed as

$$x_i - \frac{1}{2} \cdot ulp_{x_i} \leq A \cdot y_i \leq x_i + \frac{1}{2} \cdot ulp_{x_i} \tag{3.9}$$

where x_i is the mantissa part of the ith floating-point sample in X, y_i is corresponding mantissa part of ith floating-point sample in Y, and A is a normalized common multiplier. Rational approximation using the continued fraction is applied to estimate the ApxCF. The interval function for the rational approximation is

$$\frac{x_i - \frac{1}{2} \cdot ulp_{x_i}}{\hat{x} + \frac{1}{2} \cdot ulp_{\hat{x}}} \leq \frac{y_i}{\hat{y}_i} \leq \frac{x_i + \frac{1}{2} \cdot ulp_{x_i}}{\hat{x} - \frac{1}{2} \cdot ulp_{\hat{x}}} \tag{3.10}$$

where $\hat{x}$ is a selected sample of x_i in the frame, and $\hat{y}_i$ is the estimated corresponding value of $\hat{x}$ calculated by $\hat{x}$ and ith sample x_i in X.

The mantissa part of the common multiplier A (ApxCF) is normalized to $1.0 \leq 1.f_A < 2.0$ since the mantissa bits of the floating-point data are also normalized to $1.0 \leq 1.f < 2.0$. Note that the mantissa bits of $y \times 1.5$ and that of $y \times 3$ are the same if the original mantissa bits of y are the same.

Once reliable $1.f_A$ is found, computing Y and Z from X and A will become straightforward. A $shift$ value that comply with following formula will be calculated:

$$X = \{1.f_A \times 2^{(shift)}\} \otimes Y + Z. \tag{3.11}$$

This is just a fundamental explanation. In real cases of coding, not all of x_i have be able to be divided by A. Information in x_i should be separated into y_i and z_i while keeping those values comply with $x_i = A \otimes y_i + z_i$ in order to minimize the resulting encoded number of bits. y_i can be modified for such kind of active search because y_i is directly sent to the decoder. For example, some LSBs of y_i can be set to 0 in order to move those bits to z_i. By setting y_i to 0, all 32 bits of x_i can be moved to z_i.

3.5 Masked-LZ compression

Masked-Lempel-Ziv (Masked-LZ) coding is proposed for encoding the differential sequence Z. Masked-LZ compression is one kind of dictionary-based compression scheme. It is very similar to other Lempel-Ziv compression variants, such as the LZW compression scheme, in that there is a dictionary of previously encountered strings [29, 41, 40, 42]. The longest match string of input characters is searched using the string stored in the dictionary. The main difference is the way in which the input characters are compared with the characters of the string stored in the dictionary. Masked-LZ uses the mask

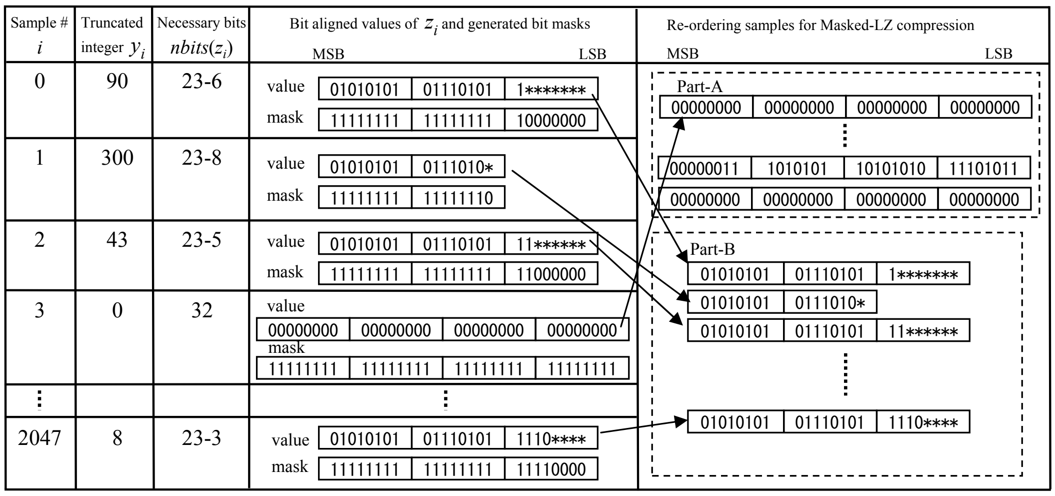

Sample # i
Truncated integer y_i
Necessary bits $nbits(z_i)$
Bit aligned values of z_i and generated bit masks
MSB
LSB
Re-ordering samples for Masked-LZ compression
0
90
23-6
value 01010101 01110101 1*******
mask 11111111 11111111 10000000
1
300
23-8
value 01010101 0111010*
mask 11111111 11111110
2
43
23-5
value 01010101 01110101 11******
mask 11111111 11111111 11000000
3
0
32
value 00000000 00000000 00000000 00000000
mask 11111111 11111111 11111111 11111111
2047
8
23-3
value 01010101 01110101 1110****
mask 11111111 11111111 11110000
Part-A
00000000 00000000 00000000 00000000
00000011 1010101 10101010 11101011
00000000 00000000 00000000 00000000
Part-B
01010101 01110101 1*******
01010101 0111010*
01010101 01110101 11******
01010101 01110101 1110****

Figure 3.3: An example of bit alignment and mask bit generation for Masked-LZ.

to compare them. The mask contains information about the bits of concern and not of concern and uses it to compare two characters. There are following three possible cases in which floating-point error will remain.

The differential sample value z_i in Z varies depending on the integer y_i and mantissa $1.f_a$ of the common multiplier A. For example, when $1.f_a = 1.0$, the number of bits to be encoded for z_i can be obtained from the magnitude of integer value y_i. This is similar to the conventional schemes [20, 21].

Case 1: All 32 bits in a floating-point sample x are packed without being compressed when the corresponding truncated integer value of a sample equals 0. This may happen when input sample x is 0 or when $x \oslash A$ is under-flowed by the truncation to get integer y because the exponent of the x is too small, where A is an estimated common multiplier. This case may also occur when x is a de-normalized number, an infinity, or Not a Number (NaN).

Case 2: Only the necessary bits for representing the difference of the mantissa are extracted and packed for the difference sequence when the ApxCF equals 1.0, which means no appropriate common multiplier was found. The necessary bits are uniquely determined by the value of the corresponding integer. They are varied from 0 to 23 bits.

Case 3: When an appropriate common multiplier A is found, 23 bits of the residual $x - (A \otimes y)$ remain and the residual may be a very small value or may be 0. However, the necessary bits can not be determined except when the residuals in the frame are all zeros. In cases other than that, the residual in the frame are all zeros, all 23 bits of the residual have to be compressed by Masked-LZ.

It follows from this that the characteristics of the remaining data are highly dependent on the input signal. For compressing such data, Masked-LZ, LZ compression with masked bit-comparison, is suitable since the necessary word length is known. Figure 3.5 shows an example of bit alignment and mask bit generation for the difference signal of the mantissa Z. The left-most column shows the sample number in a frame. Necessary bits for representing the difference of the mantissa z are uniquely determined from the corresponding truncated integer y, the common multiplier A, and the highest-bit information (see Table 3.3). First, samples of the difference mantissa are re-ordered into $PartA$ and $PartB$.

PartA: $PartA$ contains the samples needed to code all 32 bits. These are the samples for which the corresponding truncated integer y equals 0. Since the necessary bits of the sample to be encoded are 32, bits of a sample are separated into four characters from Most Significant Bit (MSB) to Least Significant Bit (LSB) order in every sample. The word size of each character is 8 bits. When all characters in $PartA$ are 0s, they are compressed with the 1 bit of all_zero_flag. When all_zero_flag equals 1, it means all bits in Part-A are 0s. Otherwise, all_zero_flag is set to 0 and masks containing all 1s, "11111111", are generated for each character. Then characters and masks from all samples in $PartA$ are ordered from the younger number of the sample to the older and in the MSB to LSB manner and fed into the Masked-LZ module for compression.

PartB: $PartB$ contains samples for which truncated integer y is not 0. Necessary bits of these samples are varied from 0 to 23 bits depending on common multiplier A and corresponding truncated integer y. When necessary bits are not the multiple of the word size, dummy bits are added at the right-most (LSB) part in order to round off any remainder bits to the word size of the character. For example, at the sample number 0, 7 bits of dummy bits are added to be a multiple of the word size since the necessary bits of sample number 0 are 17 $(= 23 - 6)$. Then, the bits of sample number 0 are separated to be the word-sized bits (8 bits) and converted into three characters. At the same time, mask bits for each character are generated by the corresponding dummy bits and they hold the dummy-bit information. Mask bits for three characters on sample number 0 are "11111111", "11111111" and "100000000". Then characters and masks from all samples in $PartB$ are ordered from the younger number of the sample to the older and in the MSB to LSB manner and fed into the Masked-LZ module for compression.

In both *PartA* and *PartB*, all bits of the part are left uncompressed and packed when the compressed size of the part is not smaller than the total size of the original necessary bits. The no-compression flag of each part is set to 1 when they are packed without compression. In the Masked-LZ compression scheme, the module searches the strings in the dictionary to find the longest matching string for input characters. At the search, Masked-LZ uses the corresponding mask containing the information about the bits of the character of concern and not of concern to compare two characters. The index of the dictionary is coded as 9 to 15 bits, depending on the number of the entries stored in the dictionary. BUMP_CODE and a FLUSH_CODE are used for the synchronization of the dictionary in the encoder and decoder. Table 3.4 shows the special index codes of the Masked- LZ.

Table 3.3: Necessary number of bits for z_i.

Condition	Range of absolute integer y_i	$nbits(z_i)$
$acf\_mantissa[c] = 0$	$\|y_i\| = 0$	32
	$2^k \leq \|y_i\| < 2^{(k+1)}$ $(0 \leq k < 23)$	$23-k$
$acf\_mantissa[c] \neq 0$	$\|y_i\| = 0$	32
	$\|y_i\| \neq 0$	23

Table 3.4: Special index codes of Masked-LZ.

Range of code	Special index code	Value
$9 \leq code\_bits \leq 15$	FLUSH_CODE	256
	FREEZE_CODE	257
	FIRST_CODE	258
	BUMP_CODE	$(2^{code\_bits}) - 1$
	MAX_CODE	$2^{15} - 1$

3.6 Overview of decoding

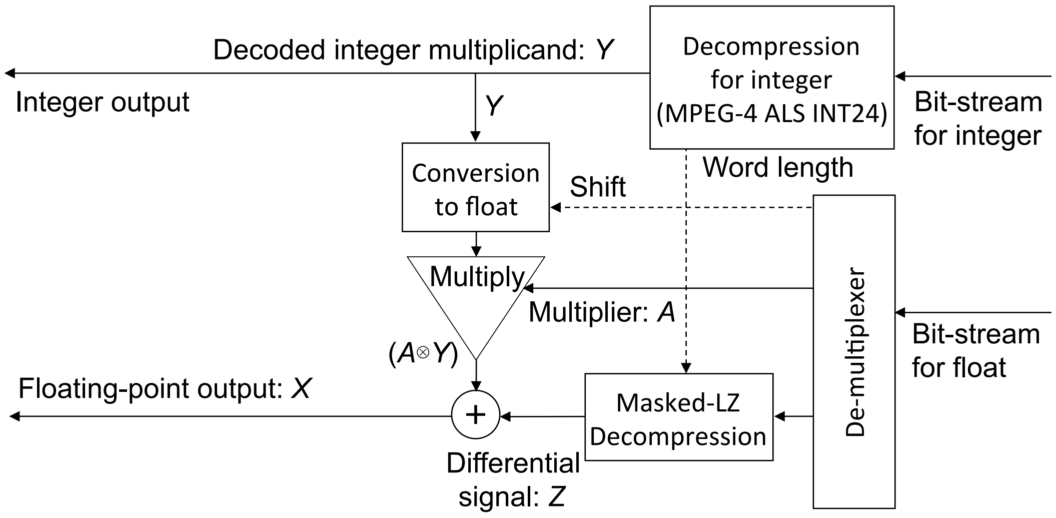


Figure 3.4: Block diagram for the decoder.

The integrated decoder is shown in Figure 3.4. For floating-point represented data, the integer multiplicand sequence Y is reconstructed and the multiplier A is multiplied to it to get the floating-point sequence $(A \otimes Y)$. The rounding mode "round to nearest, to even when tie" is used to round off the operation after the multiplication. And the difference sequence is decoded by the Masked-LZ decompression module and converted to a floating-point format sequence Z. $PartA$ and $PartB$ are decoded separately and aligned for reconstruction. If multiplier A equals 1.0, the difference sequence is decoded using the word-length information, which is defined from the value of the corresponding integer value. Additional bits longer than the necessary bit length are cut off (thrown away) since they are dummy bits added by the encoder. Both sequences, $(A \otimes Y)$ and Z, are summed to generate the output floating-point sequence. The operation of the floating-point multiplication, truncation, and summation are emulated by integer multiplication and summation in the decoding process.

$$x_i = A \otimes y_i + z_i \tag{3.12}$$

Table 3.5: Bit stream for the differential signal.

Field	#Bits	Description / Values
Frame header		
$num\_bytes\_diff\_float$	32	Encoded size of differential code
use_acf	1	1: $acf\_flag[c]$ exists 0: $acf\_flag[c]$ does not exist
Channel info. (repeated for number of channels)		
$acf\_flag[c]$	1	1: $acf\_mantissa[c]$ exists 0: $acf\_mantissa[c]$ does not exist
$acf\_mantissa[c]$	23	Mantissa part $(1.f)$ of the common multiplier A $(1.f_A = 1.0$ when $0)$
$highest\_byte[c]$	2	1-3: Max. num. of bytes per sample in $PartB$ 0: $PartB$ does not exist
$PartA\_flag[c]$	1	1: $PartA$ exists 0: $PartA$ does not exist
$shift\_amp[c]$	1	1: $shift\_value[c]$ exists 0: $shift\_value[c]$ does not exist
$shift\_value[c]$	8	Exponent part of the common multiplier A
$PartA$		
$compressed\_flagA[c]$	1	1: Compressed with the Masked-LZ coding tool 0: No compression is applied
$string\_code$	Variable	Encoded bit-stream of $PartA$
$PartB$		
$compressed\_flagB[c]$	1	1: Compressed with the Masked-LZ coding tool 0: No compression is applied
$string\_code$	Variable	Encoded bit-stream of $PartB$
Byte align		
align bits	1..7	byte align (0)

Table 3.6: Initial values for decoding parameters.

Variable	Initial value	Description
$last\_acf\_mantissa[c]$	0	initialize the A of previous frame with 1.0
$last\_shift\_value[c]$	0	
$freeze\_flag$	0	0: enable updating the directionally (set to 1 when received $FREEZE\_CODE$)
$code\_bits$	9	Number of bits for the dictionary index
$bump\_code$	511	
$next\_code$	258	Initialized to $FIRST\_CODE$

3.7 Performance evaluation

The proposed compression scheme was implemented and tested using several floating-point music sequences. Floating-point sequences for the test were generated from 16-bit and 24-bit integer sequences shown in Table 3.7. All four conditions in Table 3.7 contain the same items listed in Table 3.8 in different format (the 192 kHz, 24-bit set only contains a subset). The music source is originally played by New York Symphonic Ensemble and recorded by Matsushita Electric Industrial Co., Ltd. (“MEI”). Integer sequences of lower sampling rate (96 kHz, 48 kHz) and lower wordlength (16 bit) were made available by downsampling and truncating (with proper dither) sequences of higher sampling rates and higher wordlength. Duration of those sequences are 30 seconds each.

In order to convert those integer sequences into floating-point representation, 32-bit floating-point multiplication factors of 1.0, 1.5 and 2.99 are applied. In addition, fade-in/-out data sets are generated by applying triangle windows to the integer sequences.

For each input condition, compression performance of ZIP, the proposed scheme with and without ApxCF coding and Masked-LZ are compared. The 24-bit integer compression module of MPEG-4 ALS is used. For the conditions without ApxCF coding or Masked-LZ, the multiplier A was fixed to 1.0 for ApxCF coding, and the compressed flag for the Masked-LZ compression was fixed to 1 (uncompressed) in all frames.

The compressed ratio R is calculated as

$$R : Compressedratio = \frac{Compressed\ size}{Original\ size} \times 100. \tag{3.13}$$

Floating-point sequences for the test are generated from 16-bit and 24-bit integer sequences shown in Table 3.7. Input integer sequences were converted into 32-bit floating-point by applying floating-point multiplications factors of 1.0, 1.5 and 2.99 as shown in Table 3.9. Figures 3.5 shows the averaged compressed ratio for all generated input data (15 sequences from 16-bit, 36

Table 3.7: Integer sequences.

48 kHz sampling rate, 16-bit resolution, stereo	15 items
48 kHz sampling rate, 24-bit resolution, stereo	15 items
96 kHz sampling rate, 24-bit resolution, stereo	15 items
192 kHz sampling rate, 24-bit resolution, stereo	6 items

Table 3.8: Music source.

Conditions	Files (30 sec each)	
48 kHz, 16 bit, stereo 48 kHz, 24 bit, stereo 96 kHz, 24 bit, stereo 192 kHz,24 bit, stereo	6 files originally recorded in 192 kHz, 24 bit, stereo	
	avemaria.wav	(Avemaria / C. Gounod)
	broadway.wav	
	cymbal.wav	(MEI original recording)
	dcymbals.wav	
	etude.wav	(Etude / F. Chopin)
	mfv.wav	
48 kHz, 16 bit, stereo 48 kHz, 24 bit, stereo 96 kHz, 24 bit, stereo	9 files originally recorded in 96 kHz, 24 bit, stereo	
	blackandtan.wav	
	cherokee.wav	
	clarinet.wav	(Concerto for Clarinet and Orchestra in A major K.622 / Mozart)
	etude.wav	(Etude / F. Chopin)
	flute.wav	(Concerto for Two Flutes and Orchestra RV.533 Op.42 No.2 in C major / Vivaldi)
	fouronsix.wav	
	haffner.wav	(Symphony No.35 in D major "Haffner", K.385 / Mozart)
	violin.wav	(Concerto for Violin and String Orchestra No.1, BWV1041 / Bach)
	waltz.wav	

Table 3.9: Generated floating-point sequences.

32-bit float conditions	Generated from
16-bit integer ×1.0	48 kHz, 16 bit, stereo, 15 items
16-bit integer ×1.5	48 kHz, 16 bit, stereo, 15 items
16-bit integer ×2.99	48 kHz, 16 bit, stereo 15 items
24-bit integer ×1.0	48-, 96-, 192 kHz, 24 bit, stereo, 36 items
24-bit integer ×1.5	48-, 96-, 192 kHz, 24 bit, stereo, 36 items
24-bit integer ×2.99	48-, 96-, 192 kHz, 24 bit, stereo, 36 items
16-bit fade	48 kHz, 16 bit, stereo, 15 items
24-bit fade	48-, 96-, 192 kHz, 24 bit, stereo, 36 items

sequences from 24-bit, 18 minutes total duration). The frame size was 2048 samples.

Figure 3.6 shows the averaged compression performance for the signals modified with triangle windows. Input floating-point sequences were generated from 16-bit and 24-bit integers with triangle windows applied for fade-in and fade-out. The results of proposed scheme with ApxCF coding/Masked-LZ performed are much (13.65%) better than that of without ApxCF coding/Masked-LZ in 16-bit and slightly (0.27%) better than without ApxCF coding/Masked-LZ in 24-bit.

Figures 3.7 shows the averaged compressed ratio for all generated input data (15 sequences from 16-bit, 36 sequences from 24-bit, 18 minutes total duration) in tandem truncation conditions. In the process, two tandem truncations were applied as described below: Floating-point sequences for the test were generated from 16-bit and 24-bit integers. First, input integer sequences were converted into 32-bit floating-point and the maximum magnitude of samples were normalized to 1.0. Then, 32-bit floating-point multiplications factors of 1.0, 1.5 and 2.99 were applied. Therefore, truncations have been applied twice. The encoding frame size was 2048 samples.

When input floating-point sequences were generated from 16-bit and 24-bit integers with multiplication factors of 1.5 and 2.99, the compressed size of the data was comparable to that with multiplication factor 1.0. It is obvious that the proposed scheme with ApxCF coding/Masked-LZ significantly outperformed that without ApxCF coding/Masked-LZ, under that condition. For example, the compressed ratio of the proposed coder at 16-bit integer with gain factor 2.99 was 50% better than that of ZIP and that of the proposed scheme without ApxCF coding/Masked-LZ. This means that appropriate ApxCFs were found even after two tandem truncation.

Figure 3.8 shows the results for a real music recording signal recorded and edited by a professional mixing engineer (96 kHz sampling, 32-bit float, 6 tracks, 20 to 158 sec each). With the proposed scheme (ApxCF+MLZ), improvements were observed on three tracks out of six. Though significant improvements were observed only on two tracks, the total compressed size was reduced by 4.86% compared to that of the conventional scheme (None). This means that signals in these two tracks contain a kind of redundancy that ApxCF coding and/or Masked-LZ can reduce. In other words, a common factor has been found in several frames in the tracks.

In the results, the proposed scheme with ApxCF/Masked-LZ significantly outperforms any other conditions (Without ApxCF/Masked-LZ and ZIP).

The accuracy of multiplier-estimation was checked using artificially generated sound files of 32-bit floating point format. A 24-bit integer sequence with 192-kHz sampling frequency was multiplied by 2058 types of gain factors

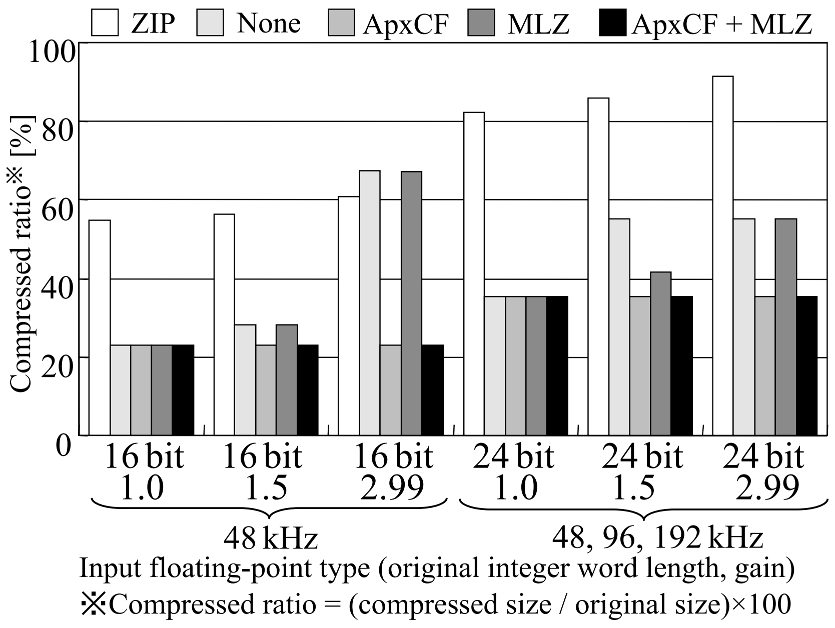


Figure 3.5: Compressed ratio for floating-point signals generated from integer signal multiplied by gain factors.

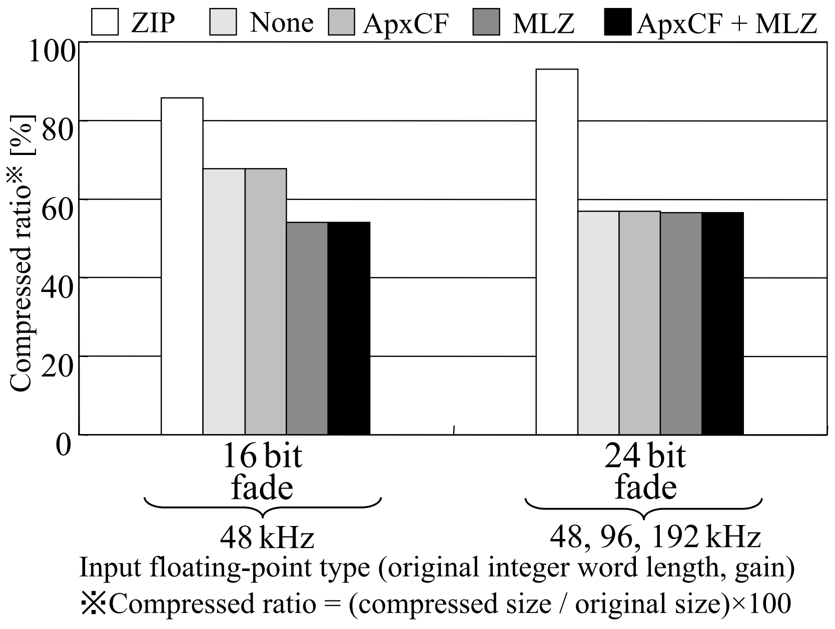


Figure 3.6: Compressed ratio for floating-point signals generated from integer signal multiplied by triangle windows.

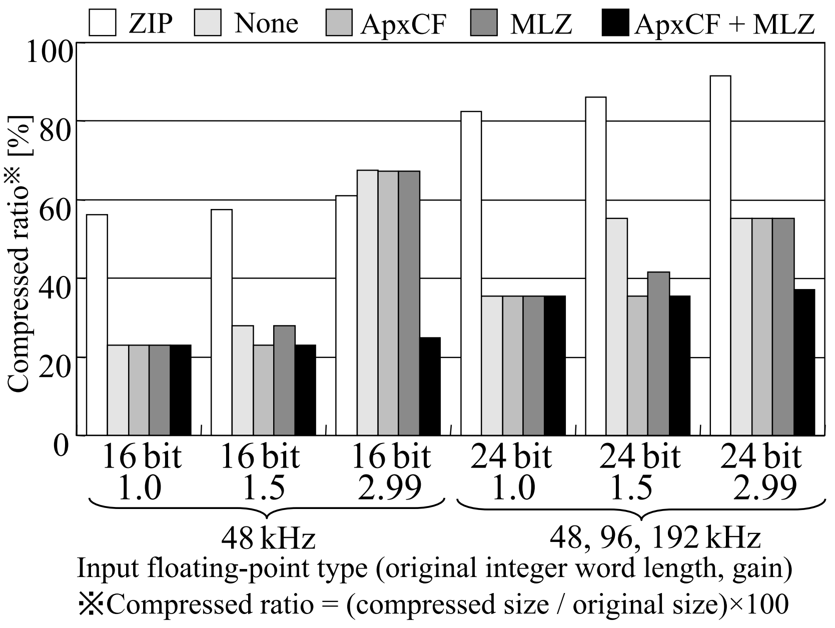


Figure 3.7: Compressed ratio for floating-point signals generated from integer signal normalized and multiplied by gain factors with two tandem truncations.

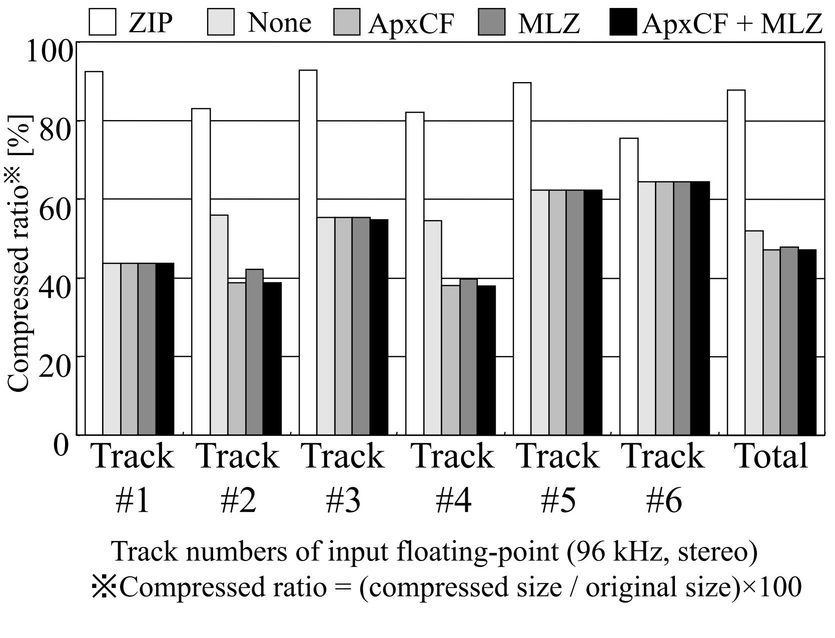


Figure 3.8: Compressed ratio for a set of multi-track floating-point music data made using a professional signal editing tool.

obtained from $1.0 \leq gain < 2.0$. Here, the gain value was set as the 23-bit mantissa part of the gain parameter. The gain can vary from 1.0 to less than 2.0, in minimum resolution of *ulp* (equivalent to the least significant bit of the mantissa). In this evaluation, mantissa bits varied from 0 to 1051638 in steps of 511. This means the gain was varied from "100000000000000000000001" to "100000000000101111110110".

The averaged compressed ratio of the proposed scheme for all input data was 27.123%. The maximum ratio and the minimum ratio are 27.209% and 27.104% respectively.

The range of the compressed ratio between maximum and minimum values of the proposed encoder is very small because compressed sizes of input data are almost the same as that when the gain factor is 1.0, though the gain factor was varied. In all sound files, appropriate common multipliers were estimated except in the first few frames of some sound files. Note that in cases when the estimated common multiplier is not good enough, the common multiplier is set to 1.0.

3.8 Summary

In this chapter, a new coding scheme, comprising Approximate Common Factor (ApxCF) coding and the Masked Lempel-Ziv (Masked-LZ) compression for the lossless coding of IEEE 754 floating-point data is introduced. In the proposed scheme, an input sequence X is decomposed into three parts: a common multiplier A, a multiplicand sequence Y, and a difference sequence Z. Instead of re-inventing a brand new coding tool, proposed scheme makes use of existing efficient encoding tool for integer input sequences.

Experimental test results show that the ApxCF coding combined with the Masked-LZ coding can reduce the bit rates considerably, especially when the input values in a frame are constructed by multiplication of the sequence of integer values and a floating-point constant. A set of real music recording signal recorded and edited by a professional mixing engineer (96 kHz sampling, 32-bit float, 6 tracks, 20 to 158 sec each) was also tested. The obtained maximum data size reduction was more than 17% for the best case file.

The scheme has been accepted as a part of an ISO/IEC standard, MPEG-4 Audio Lossless Coding (ALS).

Chapter 4

Lossless Compression Scheme for Log-companded Speech and Audio Signals

4.1 Introduction

The ITU Recommendation G.711 [1] is widely used for narrowband telephony applications, including Public Switched Telephone Network (PSTN) and General Switched Telephone Network (GSTN) and packet-based network applications such as Voice Over Internet Protocol (VoIP), and has been used for many decades because of its proven voice quality, ubiquity, and utility. ITU has established a lossless coding technology for G.711 encoded payloads.

The standard is ITU-T Recommendation G.711.0 [24]. G.711.0 is a lossless compression algorithm that operates on 40, 80, 160, 240, and 320 samples per 8-kHz sampled G.711 input frame. The bit rate is variable and the size of the (compressed) output frame depends on the input signal characteristics. The minimum size of an encoded frame is one byte. The maximum size of an encoded frame is the input frame size plus one byte. Following coding tools are included in G.711.0 [64, 65, 66, 67, 68, 69, 70, 71]: An uncompressed coding tool, constant coding tools (Constant Plus zero coding, Constant Minus zero coding, Constant non-zero coding), a Plus-Minus (PM) zero Rice coding tool, a binary coding tool, a pulse mode coding tool, a value-location coding tool, a fractional-bit coding tool, a min-max level coding tool, a direct linear predictive coding tool, and a mapped domain linear predictive (MDLP) coding tool. The MDLP coding is a kind of LP coding but especially designed for G.711 A-law and μ-law input (A similar scheme had been also proposed by F. Ghido, et. al. [76, 77]).

This chapter introduces some new coding schemes proposed and applied to the G.711.0 codec, especially related to the MDLP coding tool. PM zero mapping and Escaped-Huffman (E-Huffman) combined with adaptive recursive Rice coding are newly proposed. PM zero mapping is used to calculate the prediction residual and E-Huffman coding combined with adaptive recursive Rice coding is used as an entropy coding scheme for the prediction residual. Test results are examined in terms of the compression performance/computational complexity trade-off based on the Figure of Merit (FoM).

This author designed a structure of complete lossless codec and those proposed coding schemes are integrated into it. The resulting specification has been approved as an ITU-T standard G.711.0.

Section 4.3 overviews the mapped domain linear prediction. Sections 4.4 and 4.5 introduce PM zero mapping and E-Huffman coding combined with adaptive recursive Rice coding, respectively. Section 4.6 shows the performance evaluation test results of the proposed schemes and ITU-T standard G.711.0. Finally, Section 4.7 summarizes this chapter.

4.2 G.711 pulse code modulation

The ITU Rec. G.711 [1] coding is a form of a non-linear quantization whereby individual uniform PCM samples of 13 or 14 bit precision are compressed to 8 bits using one of two logarithmic conversion laws (A-law and μ-law). For the details, see Section 2.2.

4.3 Mapped domain linear prediction

Figure 4.1 shows a block diagram of the MDLP encoding tool used in the G.711.0 encoder. The MDLP coding tool takes a sequence of N G.711 A-law: $I_A(n)$, or G.711 μ-law: $I_\mu(n)$ symbols. First, these N G.711 symbols are converted into $x_{PCM}(n) : 0 \leq n < N$, in the uniform (linear) PCM domain and a short-term prediction is carried out in that domain using LP analysis. The prediction residual signal, however, is obtained in the range of $[-255, 255]$ since the predicted value is subtracted from the target value $x_{int8}(n)$ in the 8-bit logarithmic domain (denoted as the $int8$ domain in this section). PARCOR coefficients are used to represent and signal the LPC parameter. Linear prediction is applied as follows:

$$\hat{x}_{int8}(n) = f_{PCM \to int8}\left(-\sum_{i=1}^{P} a_i \cdot f_{int8 \to PCM}\left(x_{int8}(n-i)\right)\right) \tag{4.1}$$

where a_i is the i-th LPC coefficient of P-th order prediction, x_{int8} and $\hat{x}_{int8}(n)$ are the previous sample value and the predicted sample value in the $int8$ domain, and $f_{PCM \to int8}$ and $f_{int8 \to PCM}$ are the mapping function from uniform PCM to $int8$ and the inverse mapping function. Prediction residual is calculated in the $int8$ domain as

$$r(n) = x_{int8}(n) - \hat{x}_{int8}(n), \quad 0 \leq n < N. \tag{4.2}$$

4.4 PM zero mapping and residual calculation

In order to improve the coding efficiency of MPDL coding, a new coding scheme called PM zero mapping is newly introduced.

Because of the definition of the μ-law [1], there are two zeros (plus zero 0^+ and minus zero 0^-) in the minimum quantization interval of the μ-law signal. In the MDLP coding, the value 0^- can not be predicted because the value can't be represented in the uniform (linear) PCM domain. However, in order not to lose the value 0^-, the value 0^- has to be kept as the value -1 in the $int8$ domain (See the mapped values for $I_\mu(n)$=0x7F in Table 4.1).

Here, the PM zero mapping for the residual calculation in μ-law case has been proposed as

$$r(n) = f_{int8 \rightarrow int8'}\left(x_{int8}(n)\right) - f_{int8 \rightarrow int8'}\left(\hat{x}_{int8}(n)\right) \tag{4.3}$$

where $f_{int8 \rightarrow int8'}$ is one of the mapping functions shown in Table 4.1, which maps the values for the $int8$ domain depending on the observed existence of 0^+ and 0^- samples. First, the numbers of samples of which the value is 0^+ and the value is 0^- are counted in the input frame. Then, depending on the observation, one of the non-linear mappings shown in Table 4.1 is applied to the target value $x_{int8}(n)$ and to the predicted value $\hat{x}_{int8}(n)$ before the residual calculation. For instance, if neither 0^+ nor 0^- is observed in the frame, all input sample values $x_{int8}(n)$ and predicted sample values $\hat{x}_{int8}(n)$ of $+1$, 0 and -1 are mapped to 0, and other negative values are increased by 1 before the residual calculation of Equation (4.3). The selected mapping (existence of 0^+ and 0^-) for the frame is signaled by the corresponding mapping index codes shown in Table 4.1. With this mapping, the magnitude of the prediction residual can be reduced by 1 (or 2) when one of (or both of) the zeros is (are) not found in the input frame and the signs of the target value and the predicted value are different. The code is sent only for μ-law input.

4.5 Prediction residual coding

In the G.711.0, E-Huffman coding combined with adaptive recursive Rice coding is newly proposed as an entropy coding scheme for the prediction residual signal $r(n) : 0 \leq n < N$, and applied to frames larger than 40 samples. For 40-sample frames, Rice coding is used. First, the residual $r(n)$ is decomposed into a quotient $k(n)$ and a remainder $j(n)$ based on a separation parameter S, as described later in Sections 4.5.1 and 4.5.2. The separation parameter S (Rice parameter for Rice coding) is calculated as

$$S = \left\lfloor log_2\left(2ln(2) \cdot \bar{r}\right) + \lambda \right\rfloor \tag{4.4}$$

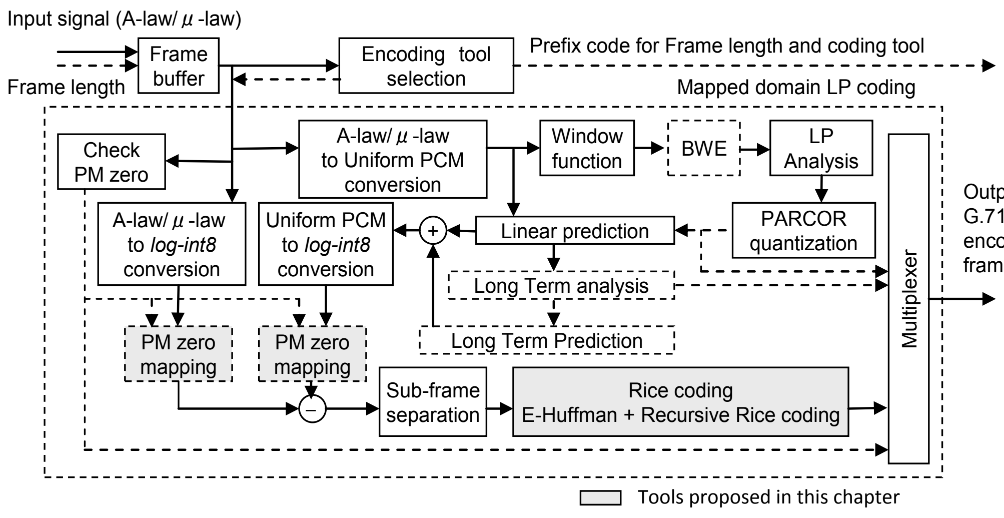

Input signal (A-law/ μ -law)
Frame length
Frame buffer
Encoding tool selection
Prefix code for Frame length and coding tool
Mapped domain LP coding
Check PM zero
A-law/ μ -law to Uniform PCM conversion
Window function
BWE
LP Analysis
A-law/ μ -law to log-int8 conversion
Uniform PCM to log-int8 conversion
Linear prediction
PARCOR quantization
Long Term analysis
PM zero mapping
PM zero mapping
Long Term Prediction
Sub-frame separation
Rice coding E-Huffman + Recursive Rice coding
Multiplexer
Output G.711.0 encoded frame
Tools proposed in this chapter

Figure 4.1: Block diagram of the mapped domain linear prediction tool in the G.711.0 encoder.

Table 4.1: PM zero mapping functions for μ-law values.

$I_\mu(n)$	$x_{PCM}(n)$	$x_{int8}(n)$ $\hat{x}_{int8}(n)$	0^+ and 0^- exist	No 0^+ nor 0^- exist	No 0^+ exist	No 0^- exist
0x80	+8031	+127	+127	+126	+127	
0x81	+8015	+126	+126	+125	+126	
⋮	⋮	⋮	⋮	⋮	⋮	
0xFE	+2	+1	+1	+0	+1	
0xFF	+0	0	0	0	0	
0x7F	-0	-1	-1	0	0	
0x7E	-2	-2	-2	-1	-1	
⋮	⋮	⋮	⋮	⋮	⋮	
0x01	-8015	-127	-127	-126	-126	
0x00	-8031	-128	-128	-127	-127	
Mapping index code			0	100	101	110

where $\bar{r}$ is averaged absolute amplitude of $r(n)$ calculated in the range $min(2, P) \leq n < N$, $\lambda = 0.5$ for Rice coding (for 40-samples frame) and $\lambda = 0.3$ for E-Huffman coding. The value of S is Huffman encoded in 1- to 6-bits and transmitted to the decoder. When sub-frame separation is applied, a difference of S from the previous sub-frame is encoded for the second and latter sub-frames.

4.5.1 Golomb-Rice coding (conventional scheme)

Rice code (also known as Golomb-Rice code) [25, 26] of the residual value $r(n)$ is calculated as follows when a Rice parameter S is given:

If $S = 0$, after $k(n)$ 0s, one 1 is presented.

$$k(n) = \begin{cases} 2r(n) & if \quad r(n) \geq 0 \\ -2r(n) - 1 & if \quad r(n) < 0 \end{cases} \tag{4.5}$$

For cases $S > 0$, after $k(n)$ 0s, one 1 appears. Then remainder $j(n)$ follows in S-bit representation.

$$k(n) = \begin{cases} \lfloor 2^{-(S-1)} r(n) \rfloor & if \quad r(n) \geq 0 \\ \lfloor 2^{-(S-1)} (-r(n) - 1) \rfloor & if \quad r(n) < 0 \end{cases} \tag{4.6}$$

$$j(n) = \begin{cases} r(n) \circledast \left(2^{(S-1)} - 1\right) + 2^{(S-1)} & if \quad r(n) \geq 0 \\ (-r(n) - 1) \circledast \left(2^{(S-1)} - 1\right) & if \quad r(n) < 0 \end{cases} \tag{4.7}$$

where $\circledast$ denotes an AND bit-operator.

4.5.2 E-Huffman coding with adaptive recursive Rice coding

The distribution of the residual signal $r(n)$, however, sometimes does not follow the expected model of Rice coding. Some of the quotient values $k(n)$ are relatively larger than expected.

In order to improve the coding performance on such an input, a new coding scheme, E-Huffman coding combined with adaptive recursive Rice coding has been devised. The coding scheme is used for encoding the quotient $k(n)$ calculated by Equations (4.5) and (4.6) with the separation parameter S. Table 4.2 shows the E-Huffman code table entries. The best one of the four E-Huffman tables is selected for every frame on the basis of the estimated encoded bitstream size of the frame. The selected E-Huffman table for the frame is signaled by an index code. Note that the number of entries is limited to $maxCode$, which is either 6 or 7. Quotient values $k(n) < maxCode$ can be encoded by one of the Huffman codes listed in E-Huffman tables I to IV. For

Table 4.2: Escaped Huffman code tables.

$k(n)$	E-Huffman table			
	I	II	III	IV
0	01	1	1	1
1	10	01	01	01
2	11	0001	001	001
3	001	0010	00001	0001
4	0001	00001	00010	00001
5	00001	000000	000000	000000
6	000000	000001	000001	000001+E
7	000001+E	0011+E	00011+E	-
$maxCode$	7	7	7	6
Index Code	01	000	001	1

Note: E denotes an extra code for $k(n) - maxCode$.

the quotient values $k(n) \geq maxCode$, an extra code $E(k(n) - maxCode)$ is produced after the escape code, e.g., '00001'. The coding scheme used for the extra code E is adaptively switched depending on the separation parameter S (Therefore, the coding scheme is switched depending on averaged amplitude and individual values of the residual signal). Unary code is used for the extra coding if $S = 0$. Rice coding with a Rice parameter of 1 is used for the extra coding if $S > 0$. Note that E-Huffman table IV is equivalent to Rice code when $S = 0$ or $k(n) < maxCode$.

4.6 Evaluation of the proposed schemes

The proposed coding algorithms and conventional algorithms were implemented in ANSI-C using software tool library STL2005 v2.2 [36]. The coding performance of the codec in various conditions were evaluated in terms of the computational complexity/compression performance trade-off based on the Figure of Merit (FoM).

4.6.1 Figure of Merit (FoM)

The performance of the codec should be measured in terms of the computational complexity/compression performance trade off.

The trade-off is assessed by using the following FoM:

$$FoM = 100 - R - w \times max(C, C_{obj}) \tag{4.8}$$

where C is averaged computational complexity for encoder/decoder pair measured using basic operators defined in ITU-T Recommendation averaged over all input frame sizes and conditions in weighted million operations (WMOPS), and R is compressed ratio defined as

$$R : Compressed\ ratio\ [\%] = \frac{Compressed\ size}{Original\ size} \times 100. \tag{4.9}$$

The weighting factor for the complexity penalty was set to $w = 2.0$ based on the discussion for ToR [63] among the experts in ITU-T. The minimum penalty factor was clipped on the basis of the objective complexity value $C_{obj} = 1.0$.

4.6.2 Test corpora

The test corpora used for the evaluation test are shown in Tables 4.3 and 4.4. The Corpus I was generated from the P.501 speech corpus [78]. Corpus II was chosen from the "Restricted Languages Multilingual Speech Database 2002" [79] and the "Ambient Noise Database CD-ROM" [80] provided by NTT Advanced Technology Corporation. The durations of input speech signals for Corpora I and II are 523 seconds and 751 seconds, respectively, which result in over 425 hours of processed data for all categories of test conditions listed in Table 4.3. A μ-law corpus recorded from an in-service network operated in Japan (48 hours/1.4 GB) was provided by Nippon Telegraph and Telephone Corporation [81] and was also used as the corpus III shown in Table 4.4.

4.6.3 Performance of the PM zero mapping

Tables 4.5 and 4.6 shows the compression performance of the conventional MDLP coding scheme (similar to [76, 77]) and the MDLP with the PM zero mapping scheme newly introduced. Test corpora described in Section 4.6.2 were used for the test. It is shown that the proposed PM zero mapping (PMZ) performs better than the conventional scheme (No PMZ) for all frame lengths except 40. Better results are under lined. Based on the results in Table 4.5, PM zero mapping was applied only for μ-law signals in 80-sample frames and larger in the actual G.711.0 specification. Table 4.7 shows the complexity of encoder and decoder and the FoM for the G.711.0 coder with and without PM zero mapping for μ-law input signals. The FoM score is increased by 0.19 and the compression ratio is improved by 0.2%, while averaged complexity is increased by 0.06 WMOPS.

Table 4.3: Information of Corpora I and II.

	Corpus I	Corpus II
Speech duration	523 seconds	751 seconds
Languages	American English, British English, Mandarin Chinese, Finnish, French, German, Italian, Japanese, Polish, and American Spanish	American English, British English, Cantonese Chinese, Mandarin Chinese, French, German, Japanese, and Spanish
Speakers	4 sentence pairs spoken by 4 different speakers (2 male and 2 female)	100 Cantonese sentences, 180 Mandarin sentences, 300 American English sentences, 150 Spanish sentences, and 200 sentences of other languages spoken by 2 different speakers (1 male and 1 female)
Test categories	(a1): Clean speech case of input levels -16, -26 and -16 dBov; voice activity factor (VAF) of 45 % ±1 % both A-law and μ-law	
	(a2): Noisy speech case of input level -26 dBov; VAF of 45 % ±1 %; both A-law and μ-law; SNRs of 15, 20, and 25 dB; noise conditions: cafeteria, street, office noise, interfering talker, background music.	
	(a3): Tandem cases with G.711.1 R1, EFR+DTX, G.729, and G.726 for clean speech/noisy speech conditions same as above (a1) and (a2) (for EFR, also car noise added).	
Total duration of all test signals	137 hours	194 hours

Table 4.4: Information of Corpus III.

	Corpus III
Speech duration	48 hours
Languages	Japanese
Speakers	4 different speakers (2 male and 2 female) DTMF and FAX signals are also included.
Test categories	(b): Recorded μ-law corpus that includes conditions of DTMF, FAX, Office noise, Babble noise, Street noise, Train station, Hotel lobby, Interference talkers and back ground music; with conditions of IP phones, ISDN, DSTN, Mobile phones (tandem with G.726 and AMR).
Total duration of all test signals	48 hours

Table 4.5: Compression performance of the conventional and with PM zero mapping (PMZ) for μ-law signal [%].

	Frame length	No PMZ	PMZ
μ-law	40	<u>53.90%</u>	54.03%
	80	50.41%	<u>50.25%</u>
	160	48.04%	<u>47.76%</u>
	240	47.44%	<u>47.16%</u>
	320	47.16%	<u>46.87%</u>

Table 4.6: Test results with/without PM zero mapping (PMZ) for μ-law input signal.

	No PMZ (ref.)	PMZ ($N > 40$)
Input data size [Mbytes]	3.354	3.354
Encoded size [Mbytes]	1.657	1.650
Compressed ratio [%]	49.39%	<u>49.19%</u>
Complexity [WMOPS, weighted millions of operation]		
Average complexity (total enc.+dec.)	1.125	1.131
Worst-case complexity (enc.+dec.)	1.655	1.667
FoM score	48.3584	<u>48.5493</u>

4.6.4 Performance of the adaptive recursive Rice coding and E-Huffman coding

Table 4.5 shows the compression performance, complexity, and FoM for conventional Rice coding (Rice), adaptive recursive Rice coding (RR), E-Huffman without adaptive recursive Rice coding (EH), and E-Huffman combined with adaptive recursive Rice coding (EH+RR). The (EH+RR) is the scheme applied to the G.711.0. Note that all additional tools are applied only for the 80-sample frames and larger because the additional index code required for the E-Huffman coding had made the FoM score worse for 40-sample frames. All results are averaged over all input categories and all frame lengths for both A-law and μ-law with the test corpora described in Section 4.6.2. It is shown that the compression performance of the RR is 0.03% better than that of Rice but that the FoM score became worse because the computational complexity is increased by 0.015 WMOPS. Both the compression performance and FoM score are improved by EH. When EH is combined with adaptive RR, com-

Table 4.7: Performance results for the adaptive recursive Rice coding (RR) and E-Huffman (EH) coding.

	Rice	RR ($N > 40$)	EH ($N > 40$)	EH+RR ($N > 40$)
Compressed ratio [%]	46.87% (reference)	46.84% (-0.03)	46.75% (-0.12)	<u>46.71%</u> (-0.16)
Complexity [WMOPS, weighted millions of operation]				
Average complexity (total enc.+dec.)	1.025	1.040	1.070	1.071
Worst-case complexity (enc.+dec.)	1.588	1.616	1.663	1.667
FoM score	51.080	51.079	51.106	<u>51.152</u>

pression performance is further improved because EH reduces the code size for quotient values $k(n) < maxCode$ and RR reduces the code size for larger quotient values, $k(n) \geq maxCode$. The combined scheme improves both the compression performance and FoM score. Note that the average complexity of EH+RR is increased only 0.001 compare to that of EH because RR can be implemented on top of EH at almost no cost.

4.6.5 Performance of ITU-T G.711.0

This author designed a structure of complete lossless codec. The proposed coding schemes, PM zero mapping, E-Huffman combined with adaptive recursive Rice coding are integrated along with the other coding tools described in Section 2.5.2. The resulting specification has been approved as an ITU-T Recommendation G.711.0.

The lossless codec is implemented in ANSI-C using the ITU-T Software Tool Library STL2005 v2.2 [36]. Complexity and compression performance per each set of conditions specified in the G.711.0 ToR [63] and processing plan [81] are examined.

Test corpora I, II, and III shown in Tables 4.3 and 4.4 in Section 4.6.2 are used for the performance evaluation. The durations of input speech signals for Corpora I and II are 523 seconds and 751 seconds, respectively, which resulted in over 425 hours of processed data for all categories of test conditions listed in Table 4.3. Table 4.8 provides results for Corpora I, II. Table 4.9 shows the results for each test category in Corpora I, II, and III.

The compressed ratio is calculated as:

Table 4.8: Results for Corpora I, II and III.

	Corpus I	Corpus II	Corpus III
Input data size [Mbytes]	5.327	6.969	1317.822
Encoded size [Mbytes]	2.489	3.005	647.563
Compressed ratio [%]	46.73%	43.13%	49.17%

$$R : Compressed\ ratio\ [\%] = \frac{Compressed\ size}{Original\ size} \times 100 \qquad (4.10)$$

Table 4.8 shows the compression performance of the integrated codec for Corpora I, II and III. Table 4.9 shows the result of each test category for Corpora I, II and III. Compressed size and complexity results are averaged over all input frame lengths under the condition. The computational complexity is reported in Weighted Millions of Operations Per Second (WMOPS).

Tables 4.10 and 4.11 show the contribution of each tool for μ-law and A-law conditions in each frame length and overall contribution. Table 4.12 shows the complexity of the codec for each frame length in μ-law and A-law conditions.

It is shown that the Mapped Domain Linear Prediction (MDLP) coding tool is used around 83% to 89% of the input frames for μ-law, and 70% to 73% for A-law. According to the complexity, the 160-sample frame under μ-law condition is the worst case in both for encoder and decoder.

Table 4.13 shows the comparison results of MDLP coding tool with and without proposed schemes, PM zero mapping, E-Huffman and adaptive recursive Rice coding. Conventional Rice coding is used for the case when those proposed schemes are disabled. Note that PM zero mapping is used only for μ-law and all proposed schemes are used only in frame length $N > 40$. The test results show that MDLP with proposed schemes is used 0.11% more for μ-law and 0.01% more for A-law conditions compared to simple Rice coding. The proposed schemes improve the coding performance relatively 0.51% for μ-law and 0.17% for A-law within the MELP coding tool. Overall compression gain introduced by the proposed schemes are 0.4% for μ-law and 0.13% for A-law.

The G.711.0 read only memory (ROM) size and random access memory (RAM) data requirements and program size (in number of basic operators) are shown in Table 4.14. Note that μ-law compression is less than A-law (A-law encodes low amplitude signals more coarsely) and greater than 50% compression is achieved for the recorded (service provider) corpus and all but the two high noise μ-law test conditions.

Table 4.9: Results for Each Test Category.

Test category		Compressed ratio [%]	
		A-law	μ-law
(a1): Clean Speech	-16 dBoV	40.44%	40.33%
	-26 dBoV	30.61%	39.38%
	-36 dBoV	22.99%	27.45%
(a2): Noisy speech	SNR 15 dB	49.10%	55.48%
	SNR 20 dB	43.57%	52.85%
	SNR 25 dB	39.36%	47.57%
(a1) and (a2) conditions in total		42.45%	49.76%
(a3): Tandem conditions in Total		39.92%	45.48%
(b): Recorded μ-law (Corpus III)		-	49.17%

Table 4.10: Contribution of coding tools (μ-law).

Coding tools	Frame size	40	80	160	240	320	Total
Uncompressed coding tool	Used in	0.13%	0.06%	0.00%	0.00%	0.00%	0.04%
	Comp. ratio	102.50%	101.25%	100.63%	100.42%	100.31%	101.02%
Constant coding tools	Used in	4.16%	3.85%	3.51%	3.32%	3.10%	3.59%
	Comp. ratio	2.50%	1.25%	0.63%	0.42%	0.31%	1.10%
PM zero Rice coding tool	Used in	0.46%	0.21%	0.21%	0.20%	0.21%	0.26%
	Comp. ratio	11.48%	6.24%	3.04%	2.75%	2.22%	6.38%
Binary coding tool	Used in	0.19%	0.10%	0.03%	0.03%	0.03%	0.08%
	Comp. ratio	15.00%	13.75%	13.13%	12.92%	12.81%	14.20%
Pulse mode coding tool	Used in	0.04%	0.01%	0.00%	0.00%	0.00%	0.01%
	Comp. ratio	9.69%	6.94%	1.97%	0.15%	0.40%	5.00%
Value location coding tool	Used in	0.00%	5.80%	4.87%	10.05%	9.48%	6.04%
	Comp. ratio	0.00%	13.99%	11.46%	15.45%	14.80%	14.32%
Mapped Domain LP coding tool	Used in	83.43%	87.03%	89.54%	85.95%	86.83%	86.56%
	Comp. ratio	59.55%	55.82%	52.20%	52.90%	52.25%	54.49%
Fractional bit coding tool	Used in	9.68%	2.92%	1.85%	0.45%	0.35%	3.05%
	Comp. ratio	22.51%	24.46%	23.34%	24.53%	24.68%	23.09%
Min-Max level coding tool	Used in	0.67%	0.00%	0.00%	0.00%	0.00%	0.13%
	Comp. ratio	73.71%	0.00%	0.00%	0.00%	0.00%	73.71%
Direct LP coding tool	Used in	1.24%	0.00%	0.00%	0.00%	0.00%	0.25%
	Comp. ratio	98.18%	0.00%	0.00%	0.00%	0.00%	98.18%
Total	Used in	100%	100%	100%	100%	100%	100%
	Comp. ratio	53.90%	50.25%	47.76%	47.16%	46.87%	49.19%

Table 4.11: Contribution of coding tools (A-law).

Coding tools	Frame size	40	80	160	240	320	Total
Uncompressed coding tool	Used in	0.41%	0.08%	0.00%	0.00%	0.00%	0.10%
	Comp. ratio	102.50%	101.25%	100.63%	100.42%	100.31%	102.29%
Constant coding tools	Used in	13.53%	11.88%	10.21%	9.33%	8.72%	10.73%
	Comp. ratio	2.50%	1.25%	0.63%	0.42%	0.31%	1.05%
PM zero Rice coding tool	Used in	9.28%	10.40%	11.41%	14.48%	14.50%	12.01%
	Comp. ratio	10.42%	7.55%	5.75%	5.75%	5.30%	6.67%
Binary coding tool	Used in	0.19%	0.05%	0.02%	0.01%	0.01%	0.05%
	Comp. ratio	15.00%	13.75%	13.13%	12.92%	12.81%	14.48%
Pulse mode coding tool	Used in	3.80%	3.42%	2.96%	0.00%	0.00%	2.03%
	Comp. ratio	12.22%	10.02%	8.83%	13.75%	13.44%	10.50%
Value location coding tool	Used in	0.00%	1.72%	1.95%	3.62%	3.77%	2.21%
	Comp. ratio	0.00%	16.81%	13.23%	13.85%	12.79%	13.84%
Mapped Domain LP coding tool	Used in	70.63%	72.45%	73.44%	72.55%	73.01%	72.42%
	Comp. ratio	60.81%	57.72%	54.28%	54.06%	53.50%	56.04%
Fractional bit coding tool	Used in	0.50%	0.01%	0.00%	0.00%	0.00%	0.10%
	Comp. ratio	23.41%	23.75%	24.99%	0.00%	0.00%	23.42%
Min-Max level coding tool	Used in	0.23%	0.00%	0.00%	0.00%	0.00%	0.05%
	Comp. ratio	86.14%	0.00%	0.00%	0.00%	0.00%	86.14%
Direct LP coding tool	Used in	1.44%	0.00%	0.00%	0.00%	0.00%	0.29%
	Comp. ratio	98.87%	0.00%	0.00%	0.00%	0.00%	98.87%
Total	Used in	100%	100%	100%	100%	100%	100%
	Comp. ratio	46.90%	43.47%	41.11%	40.60%	40.34%	42.48%

Table 4.12: Computational complexity [WMOPS].

μ-law	Frame size	40	80	160	240	320	Total
	Average (total enc.+dec.)	1.02	1.10	1.22	1.15	1.17	1.13
	Worst-case (enc.+dec.)	1.50	1.39	1.67	1.56	1.57	1.67
	Worst-case (enc.)	1.02	0.87	1.08	1.00	1.01	1.08
	Worst-case (dec.)	0.48	0.52	0.59	0.56	0.56	0.59
A-law	Frame size	40	80	160	240	320	Total
	Average (total enc.+dec.)	0.86	0.93	1.05	0.99	1.01	0.97
	Worst-case (enc.+dec.)	1.50	1.36	1.64	1.52	1.55	1.64
	Worst-case (enc.)	1.02	0.85	1.05	0.96	0.99	1.05
	Worst-case (dec.)	0.48	0.52	0.59	0.55	0.56	0.59

Table 4.13: Contribution of MDLP tool with/without proposed schemes.

μ-law	Frame size	40	80	160	240	320	Total
MDLP (Conventional Rice coding)	Used in	83.43%	86.71%	89.31%	85.96%	86.83%	86.45%
	Comp. ratio	59.55%	56.34%	52.88%	53.54%	52.90%	55.00%
	Total comp.	53.90%	50.59%	48.29%	47.71%	47.44%	49.59%
MDLP (PMZ + E-Huffman + Recursive Rice)	Used in	83.43%	87.03%	89.54%	85.95%	86.83%	86.56%
	Comp. ratio	59.55%	55.82%	52.20%	52.90%	52.25%	54.49%
	Total comp.	53.90%	50.25%	47.76%	47.16%	46.87%	49.19%
Improvements	Used in	0.00%	0.32%	0.23%	-0.01%	0.00%	0.11%
	Comp. ratio	0.00%	-0.52%	-0.68%	-0.64%	-0.65%	-0.51%
	Total comp.	0.00%	-0.34%	-0.54%	-0.55%	-0.57%	-0.40%
A-law	Frame size	40	80	160	240	320	Total
MDLP (Conventional Rice coding)	Used in	70.63%	72.42%	73.44%	72.55%	73.01%	72.41%
	Comp. ratio	60.81%	57.86%	54.50%	54.30%	53.75%	56.21%
	Total comp.	46.90%	43.58%	41.28%	40.77%	40.52%	42.61%
MDLP (E-Huffman + Recursive Rice)	Used in	70.63%	72.45%	73.44%	72.55%	73.01%	72.42%
	Comp. ratio	60.81%	57.72%	54.28%	54.06%	53.50%	56.04%
	Total comp.	46.90%	43.47%	41.11%	40.60%	40.34%	42.48%
Improvements	Used in	0.00%	0.03%	0.00%	0.00%	0.00%	0.01%
	Comp. ratio	0.00%	-0.14%	-0.22%	-0.24%	-0.25%	-0.17%
	Total comp.	0.00%	-0.11%	-0.17%	-0.17%	-0.18%	-0.13%

Note: Proposed tools are enabled in frames $N > 40$. PMZ is used only in μ-law.

Table 4.14: Required ROM and RAM sizes and number of Basic Operators for the G.711.0 C code.

ROM size [bytes]	Word16 and Word8 tables (including 2-byte pointers)	5,481 (5,721)
RAM size [bytes]	Encoder	3,586
	Decoder	1,372
	Total	4,958
Program size [number of basic operations]		3,554

4.7 Summary

In this chapter, coding schemes newly proposed and applied to the G.711.0 were described. PM zero mapping is proposed for the prediction residual calculation and E-Huffman coding combined with adaptive recursive Rice coding is proposed for the prediction residual compression. It is shown that the PM zero mapping improves the compression performance by 0.2% and improves the FoM score by 0.19 for μ-law input. The E-Huffman coding combined with adaptive recursive Rice coding improves the compression by 0.16% and the FoM score by 0.072 averaged for all test conditions, compare to the conventional Rice coding scheme. Average computational complexity is 1.071 WMOPS for the encoder/decoder pair and the worst-case complexity is 1.667 WMOPS in total.

This chapter also presented compression and complexity results of the G.711.0 standard. G.711.0 provides more than 50% average compression in service provider environments while keeping low computational complexity for the encoder/decoder pair (1.0 WMOPS average, $<$1.7 WMOPS worst case) and low memory footprint (about 5k octets RAM, 5.7k octets ROM, and 3.6k basic operators).

CHAPTER 5

Designing an Archival Information Package Format for Long-term Preservation

5.1 Introduction

There is a great need to archive or preserve digital content while making it fully accessible in its original form. Many organizations, such as libraries, movie studios, and record companies, have started implementing archiving systems, but cost-efficiency remains an issue. Sharing resources and enabling interoperability among archiving systems is one key to reducing costs and increasing efficiency. Standards for archival tools or systems can help in sharing tools and in minimizing the cost of maintaining them. In addition, defining a standard archival format for content is of critical importance. Archiving policies and the required sets of metadata and file types to be archived are difficult to generalize because individual archives have different historical and cultural backgrounds, which means the requirements for the archiving system are quite different. There might be local file formats for specific application domains still being used (for example, an old local file format for a word processor or a special file format for professional application software). In addition, different character sets used in different languages on different operating systems present challenges for standardizing archival tools.

Table 5.1 shows some package format candidates for long-term preservation. Traditionally, ZIP for Microsoft Windows, Tape ARchive (TAR) [82] for UNIX/Linux and Apple Disc iMaGe (DMG) [83] for Mac OS are used as software delivery packages for binary files on each operating system. Metadata Encoding and Transmission Standard (METS) format [84, 85], MPEG21 file format (File format with metadata), and Media eXchange File format (MXF) [86] are used in some specific application domains but those file format do not always provide the best functionalities for serving as an Information Package format.

MPEG has called for requirements for information packages to be applied to preservation and archiving contents.

Table 5.1: Package format candidates for long-term preservation.

	Packaging function	Preserve hierarchical folder structure	Selective compression schemes	Metadata handling	OS/file system interoperability	
					multi-byte char-code for filename	file attributes
ISO 9660	☆☆☆	☆☆☆	×	×	×	×
ZIP	☆☆☆	☆☆☆	×	×	×	×
WinZip +WavPack	☆☆☆	☆☆☆	☆	×	×	×
Mac DMG	☆☆☆	☆☆☆	×	×	×	×
TAR	☆☆☆	☆☆☆	×	×	×	×
METS	×	×	×	☆☆☆	×	×
MPEG-21	☆☆☆	×	☆☆	☆☆☆	×	×
MXF	☆☆☆	×	☆☆	☆☆☆	×	×

On the course of this study, an archival information package format has been designed to meet such challenges. The proposed package format offers sustainability, and playability of digital content for maximum interoperability. It can be regarded as an implementation of the information package specified by the Open Archival Information System (OAIS) reference model [23]. To support various types of input files and to maximize interoperability for handling those files, the package format design is based on packaging files with a hierarchical folder structure.

The type of content information and metadata that should or should not be stored in an archive is up to an archive's own policy or agreements. To give users the freedom to define their own set of metadata, the proposed information package format allows users to include any kind of metadata in a package. It also provides a mechanism for linking metadata to a certain object file so that applications that adopt the proposed information package format can handle any application-specific metadata via a standardized interface.

5.2 Open archival information system (OAIS) reference model

The OAIS reference model [23] is a framework for understanding and applying concepts necessary for the long-term preservation of digital information (where long term is long enough to be concerned about changing technologies). The reference model addresses a full range of archival information preservation functions including ingest, archival storage, data management, access, and dissemination. It defines a minimal set of responsibilities that must be met for an archive to be called an OAIS, and it also defines a maximal archive to provide a broad set of useful terms and concepts. The standard defines three information packages: Submission Information Package (SIP), Archival Information Package (AIP), and Dissemination Information Package (DIP). To interconnect archiving systems, the interface point must be aligned at a certain level. The interface point might be defined as information packages. For the details, see Section 5.2.

5.3 Scope of archival information package format and requirements

MPEG-A Application Format is a series of standards that make use of MPEG standards and other standards to provide a structured file format and metadata description for designated application. PA-AF is one of the specifications in this suite of standards. The work on PA-AF was done in coordination with the following organizations: JPEG, Society of Motion Picture and Television Engineers, and ISO TC 20/SC 13 with the Data Archive Ingest Working Group of the Consultative Committee for Space Data Systems.

Tables 5.2, 5.3 and 5.4 show the PA-AF requirements at the call for proposal. All requirements had been satisfied by PA-AF combined with some external pre-processing tools and application-specific context information. Figure 5.1 shows the scope of the PA-AF specification. The purpose of ISO/IEC 23000-6 PA-AF is to provide a standardized packaging format for digital files. The packaging format can serve as an implementation of the information package specified by the OAIS reference model. PA-AF specifies a metadata description to describe the original structure and attributes of digital files archived in a PA-AF file; a metadata description to describe necessary information to reverse the preprocessing applied to digital files prior to archiving them in a PA-AF file; a metadata description to describe context information related to a PA-AF file and digital files archived in it; and a file format for

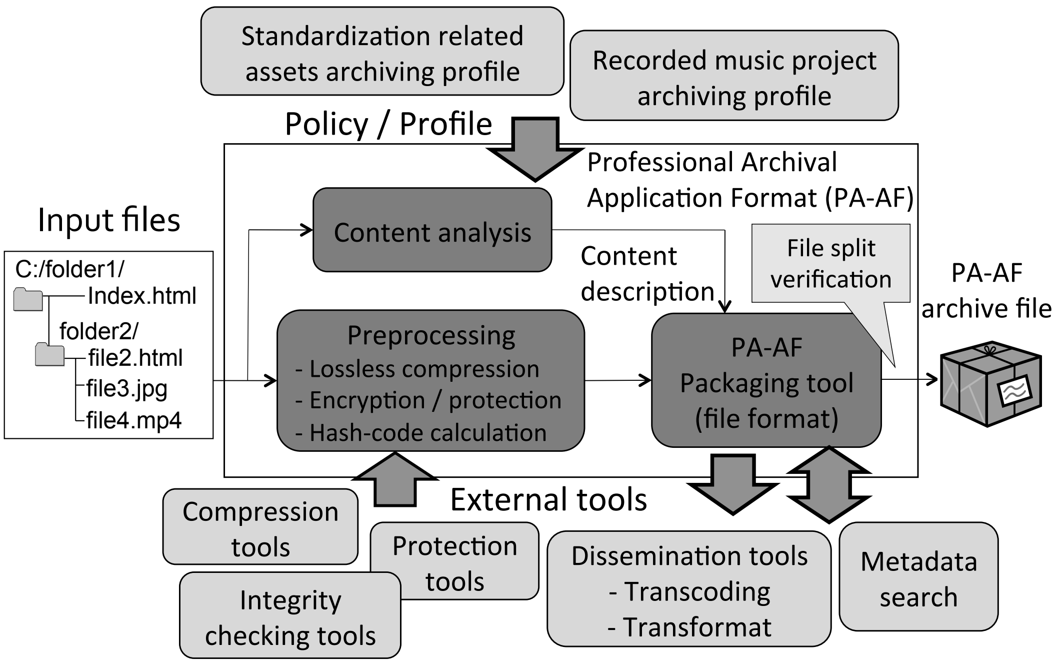


Figure 5.1: Scope of professional archival application format specification.

carrying the metadata formats and digital files.

While a general archival process may include processes ranging from creation, delivery to the archival system (ingestion), to dissemination to consumers, PA-AF is limited in scope. PA-AF specifies neither how to create input content nor any agreement on how the content should be handled, ingested into the archive, or disseminated to consumers. The archiving policy and agreements are not included in the scope of PA-AF. PA-AF is independent of any kind of compression scheme or any kind of metadata format. PA-AF provides a mechanism for identifying the preprocessing tools applied to the archived content files. Any kind of compression tool or encryption tool can be specified as an external preprocessing tool. In addition, although PA-AF optionally provides a predefined minimum set of descriptive metadata for its archived content, any kind of application-specific metadata can be stored in the PA-AF package as a content file or files if the archive's policy or agreements require it. For this purpose, PA-AF provides a mechanism for linking that metadata to the archived content file.

Table 5.2: Requirements of PA-AF (1/3)

Requirement	
A: Packaging	
A01	Packaging format of PA-AF should be able to serve as an implementation of the information package defined in the OAIS reference model.
A02	PA-AF should pack files and folder structures into single archive.
A03	PA-AF should support large files exceeding 4GB.
A04	PA-AF should support a mechanism to allow the splitting of large archive files into several smaller archive files without loss of information.
A05	PA-AF should be able to pack any kind of files including Audio, Video, Images, Metadata files and any other files.
A06	PA-AF should preserve the original file names, attributes, and the folder structure.
A07	PA-AF shall support perfect extraction of archive into its original form.
A08	Files output from extraction of an archive shall have the same directory structure as that of input files.
A09	It shall be possible to un-pack an archived file losslessly, which means complete reconstruction of original files, including the filenames, folder structures, and attributes of those files or folders.
A10	Files outputted from extraction of an archive shall be the bit-for-bit same as the original input files.
A11	PA-AF shall support extraction of all files in the archive.
A12	PA-AF shall support extraction of single files out of a collection of files in archive.
A13	PA-AF shall support browsing of archived files without having to extract the files from the archive.

Table 5.3: Requirements of PA-AF (2/3)

Requirement	
B: Cross-platform operation	
B01	PA-AF should support cross-platform operation, such as Windows, Mac, and Linux platforms.
B02	The cross-platform support should include interoperability among different file systems (file attributes) and character sets (including multiple-byte character sets).
C: Compression	
C01	PA-AF itself shall be compression-scheme independent.
C02	PA-AF should compress an archive's input files.
C03	If any compression scheme is used, PA-AF shall use a lossless compression algorithm.
C04	PA-AF shall allow different compression algorithms for different data types (e.g., MPEG-4 ALS for audio data type, JPEG2000LS for image data type, ZIP scheme for text data type).
C05	PA-AF shall allow the use of one or more compression algorithms for files with composite data type.
D: Meta-information	
D01	PA-AF should support association of richer information about files in the archive.
D02	PA-AF should provide context information about the content in the archive and the archive file itself.
D03	PA-AF should provide creation context information.
D04	PA-AF should provide content profile information.
D05	PA-AF should provide access and/or modification history of the archive.
D06	PA-AF shall provide a mechanism to accommodate application specific context information.
D07	PA-AF should provide Modality information about the content (text, images, audio, video, graphics, etc.).
D08	PA-AF should provide File format type information of the content (MP3 audio, AAC audio, MP4 video, JPEG images, GIF images, etc.).
D09	PA-AF should provide Resolution information of the content (bit rates, spatial resolution, temporal resolution, etc.).

Table 5.4: Requirements of PA-AF (3/3)

Requirement	
E: Identification	
E01	PA-AF should have a mechanism for detecting a type of the file stored in the Professional Archival AF file.
E02	PA-AF should have a mechanism for detecting a type of the file stored in the Professional Archival AF file.
F: DRM	
F01	PA-AF should support inclusion of Digital Rights Management (DRM) information for trusted exchange of an archive among rights holders, intermediaries, and users.
F02	PA-AF should support governance of archive usage and distribution.
F03	There should be a mechanism to store licensing and intellectual property rights information for each item.
F04	A mechanism should be available to allow fine-grained access control to all data items (essence, metadata, administrative data) in the archive system.
F05	PA-AF should protect individual files and file structures at various levels of granularity including the entire archive.
F06	PA-AF should support a simple passwords-lock-encryption mechanism for the Professional Archival AF file.
F07	PA-AF should support detection of post-creation content tampering.
F08	PA-AF should support validation of the Professional Archival Application Format file.
F09	PA-AF should support a mechanism for identifying the encapsulated DRM system.

5.4 Overview of proposed archival information package

On the course of this study, this author has designed an archival information package format; the resulting specification was approved as an ISO/IEC standard, MPEG-A Professional Archival Application Format (PA-AF).

PA-AF is a general packaging format that can serve the various information packages defined in the OAIS reference model. PA-AF archives digital files in a PA-AF file. In addition to containing digital files being archived, a PA-AF file also contains information for the preservation of the archived digital files. It preserves file attributes along with hierarchical folder structure information. Any kind of metadata can be stored in a PA-AF file as a content information file. PA-AF is independent from archiving policies and encoding schemes applied to content information. It maximizes interoperability on several OSs, file systems, and solves issues on OS-dependent character sets.

Advantages offered by PA-AF compared to other conventional data archival software, such as tar and zip, are many. PA-AF provides comprehensive metadata to model context information of files archived in a PA-AF file. Context information plays an important role in understanding what data is being archived in a PA-AF file. Without rich context information attached, archived data might have less value or in extreme cases become useless. Examples of context information provided by PA-AF includes: creation information (what the content is, who created it, where and when it was created, and how it was created); classification information that describes content category, e.g., Multipurpose Internet Mail Extensions (MIME) Type; and media profile information that describes the media format such as audio sampling rate, bit depth, bit rate, file format, file size, and required bandwidth.

PA-AF provides a flexible mechanism to accommodate context information specific to an application domain. Context information other than one defined by the PA-AF specification can be included in a PA-AF file. An example of such application-specific context information is Metadata Encoding and Transmission Standard metadata. PA-AF also provides a flexible mechanism to process input files prior to archiving them in a PA-AF file. It doesn't specify any mandatory preprocessing tools or modules prior to archiving content information, but does specify a mechanism to describe the use of such preprocessing tools or modules. In this way, a PA-AF file creator can choose specific preprocessing tools. PA-AF supports cross-platform file extraction. Because it preserves the structure and value of original file attributes in a platform-independent way, files archived in a PA-AF file can be extracted and put into another target platform. For example, files archived under the Microsoft

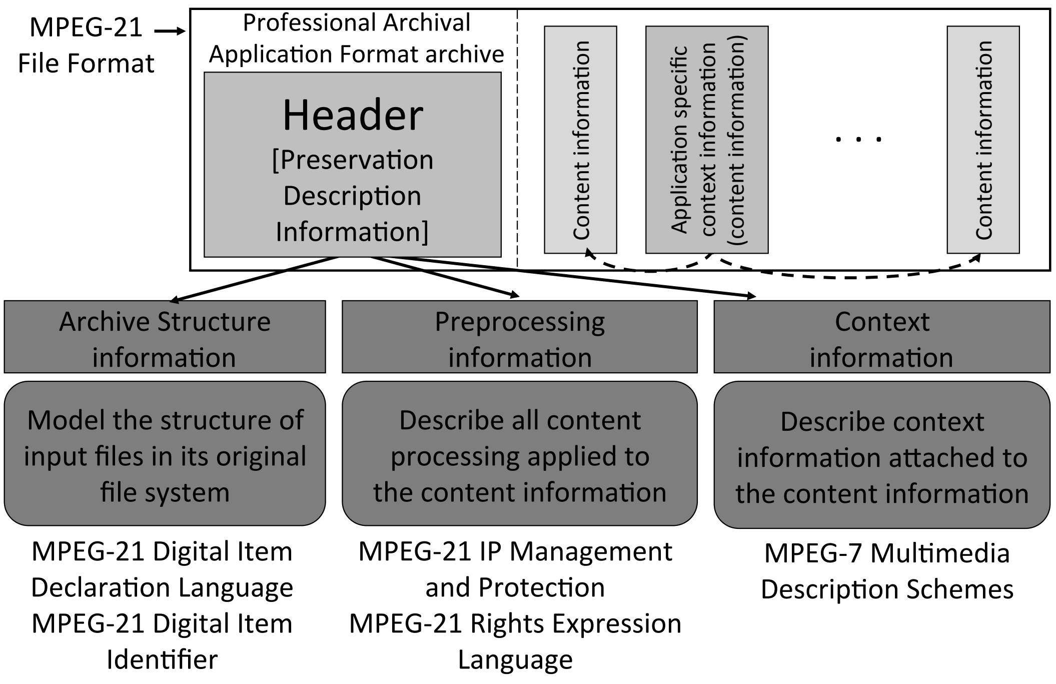

MPEG-21 File Format
Professional Archival Application Format archive
Header [Preservation Description Information]
Content information
Application specific context information (content information)
. . .
Content information
Archive Structure information
Preprocessing information
Context information
Model the structure of input files in its original file system
Describe all content processing applied to the content information
Describe context information attached to the content information
MPEG-21 Digital Item Declaration Language MPEG-21 Digital Item Identifier
MPEG-21 IP Management and Protection MPEG-21 Rights Expression Language
MPEG-7 Multimedia Description Schemes

Figure 5.2: Structure of a professional archival application format file.

FAT32 file system can be extracted to a Linux or Mac OS file system seamlessly. File names coded with multibyte character sets will be converted to the compatible encoding used on the target platform. For example, Japanese file names coded with Shift-JIS on Windows will be converted into UTF-8 for Linux or UTF-8-MAC for Mac OS X.

The PA-AF file design supports separation of metadata and files being archived. A PA-AF file can contain only metadata, while files being archived can be stored in one or more archived files. This feature is applicable for a large archival system where there are many large files to be archived. The advantage of this design is that it provides a feature to browse a set of archived collections by accessing only one file (the file that has the metadata) and provides a link to the desired archive file.

Figure 5.2 illustrates the structure of a PA-AF file. A PA-AF file consists of header content. The header part contains what is called Preservation Description Information, which is stored in the XML metadata format and includes archive Structure Information, Preprocessing Information, and Context Information. The content part contains one or more archived files that are called Content Information. Content Information includes digital data in its original format as input into the PA-AF file or in the format after it's preprocessed (transformatted) with preprocessing tools. Some example categories of preprocessing tools are lossless data compression, reversible data protection (such as encryption), and removable data attached to the content for usage governance and data integrity, such as checksums and digital licenses.

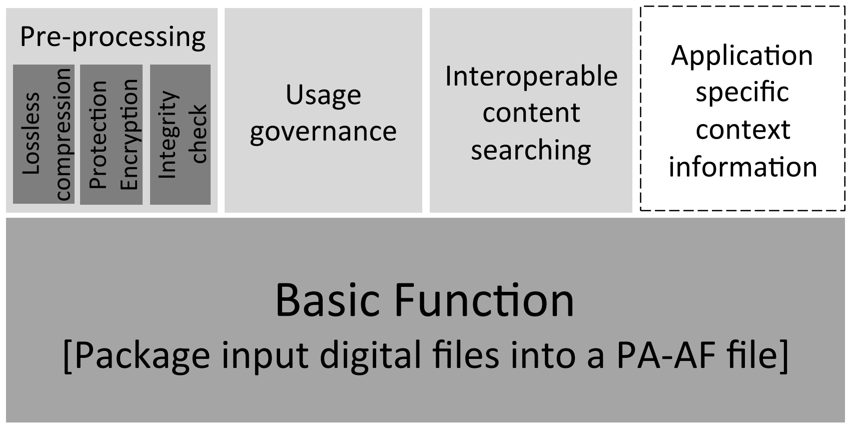

Pre-processing
Lossless compression
Protection Encryption
Integrity check
Usage governance
Interoperable content searching
Application specific context information
Basic Function
[Package input digital files into a PA-AF file]

Figure 5.3: Basic and additional functionalities of professional archival application format

Figure 5.3 shows the basic and additional functionalities of professional archival application format. Several existing MPEG-7/21 components are used to realize the PA-AF format including MPEG-21 File Format [87], MPEG-21 Digital Item Declaration Language (DIDL) [88], Digital Item Identifier (DII) [89], and MPEG-7 Multimedia Description Schemes (MDS) [90]. The basic functionality of packaging input files into the PA-AF format can be satisfied with these tools, while more advanced functionality such as usage governance would additionally require the use of MPEG-21 Intellectual Property Management and Protection (IPMP) Components [91] as well as MPEG-21 Rights Expression Language (REL) [92]. More advanced search capabilities are also enabled through additional MPEG-7 description schemes. In addition, pre-processing tools and other additional metadata dedicated to specific applications can be optionally used.

Context information which is not defined in the PA-AF specification can be included in a PA-AF file as Appliation-specific context information. Figure 5.3 illustrates an example of the uses of Application-specific context information in archiving internet page files. This example shows the benefit of using relative Uniform Resource Identifier (URI) for identifying the relation among Application-specific context information and the referenced content files.

In a PA-AF file, Application-specific context information can be stored as a file or files. The PA-AF regards the virtual location of the Application-specific context information as the (virtual) root URI. Suppose that the complete internet page files are index.html (bootstrap file/entry page), file2.html, file3.jpg, and file4.mp4. In this example, index.html is the Applicaiton-specific context information therefore the URI paths for the remaining files other than index.html are described as relative to the URI path of Index.html. Table 5.5 lists an example of the html contents of the index.html. The use of relative URI in archived internet page files allows a PA-AF compliant parser to detect the relation among the Application-specific context information file (index.html) and other files because PA-AF preserves the hierarchical folder structure of the archived files.

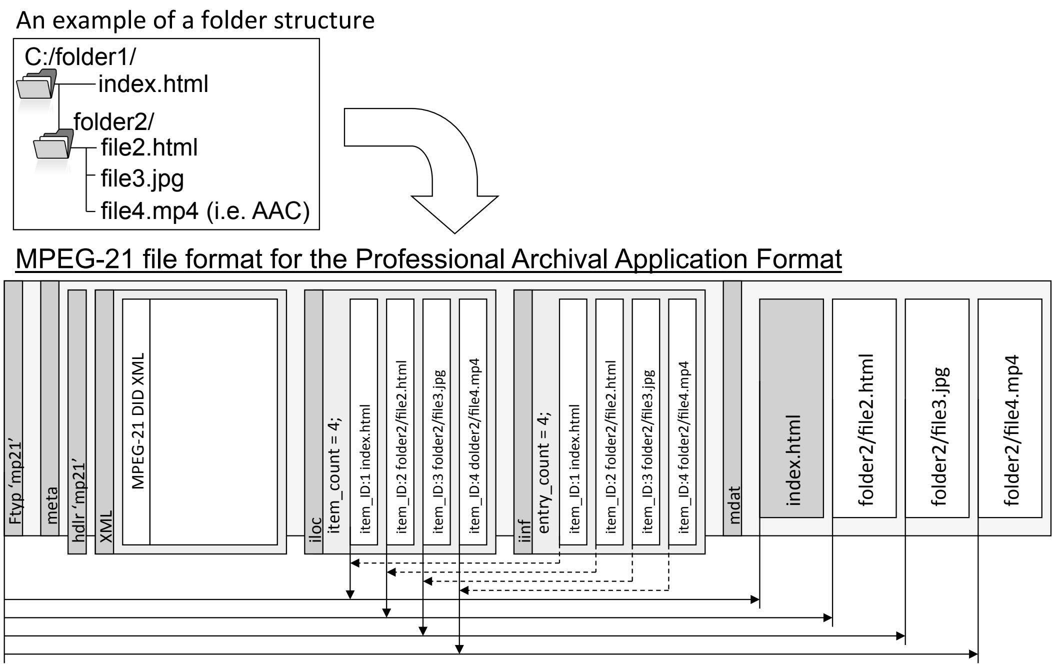

An example of a folder structure
C:/folder1/
index.html
folder2/
file2.html
file3.jpg
file4.mp4 (i.e. AAC)
MPEG-21 file format for the Professional Archival Application Format
Ftyp 'mp21'
meta
hdlr 'mp21'
XML
MPEG-21 DID XML
iloc
item_count = 4;
item_ID:1 index.html
item_ID:2 folder2/file2.html
item_ID:3 folder2/file3.jpg
item_ID:4 dolder2/file4.mp4
iinf
entry_count = 4;
item_ID:1 index.html
item_ID:2 folder2/file2.html
item_ID:3 folder2/file3.jpg
item_ID:4 folder2/file4.mp4
mdat
index.html
folder2/file2.html
folder2/file3.jpg
folder2/file4.mp4

Figure 5.4: An example of archived html files.

5.5 Implementation of PA-AF packaging/un-packaging tool

An example PA-AF API is designed and implemented as PA-AF packaging and un-packaging tools. Figure 5.5 shows a proposed library structure and application interface (API) design of the PA-AF lib/dll.

The PA-AF API library and a sample Character User Interface (CUI) application tool were implemented and tested on the following software platforms: Microsoft Visual C++ 2003, 2005 on Windows XP operating system,

Table 5.5: An example of URI links to archived content information.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.1 Transitional//E">
<HTML><HEAD><TITLE>My Index Page</TITLE>
<BODY bgcolor="#000000" text="#ccffcc" link="#cc6666" vlink="#999999" alink="#cccc99">
<a href="folder2/file2.html">reference to file2.html</a><br>
<hr>
<table>
<tr>
<td><IMG src="folder2/file3.jpg"></td>
<td>
<h1>Description text</h1>
<a href="folder2/file4.mp4">reference to file4.mp4</a><br>
</td>
</tr>
</table>
</body>
</html>
```

gcc ver 4.1.2 on Linux (Cent OS 5.2) system, gcc ver 4.0.1 on Mac OS X 10.5.7 system.

The Graphical User Interface (GUI) application tool was implemented and tested on the following software platform: Microsoft Visual C++ 2003 on Windows XP operating system.

Implementations of PA-AF API library and sample CUI and GUI application softwares have the following features:

PA-AF API library:

1. Packages archive into PA-AF file format
2. Extracts PA-AF file archive
3. Enables any pre-processing tools and post-processing tools
4. Enables access to any archived file and meta-information

CUI and GUI application:

1. Packages archive into PA-AF file format
2. Extracts PA-AF file archive and displays corresponding metadata of the contents

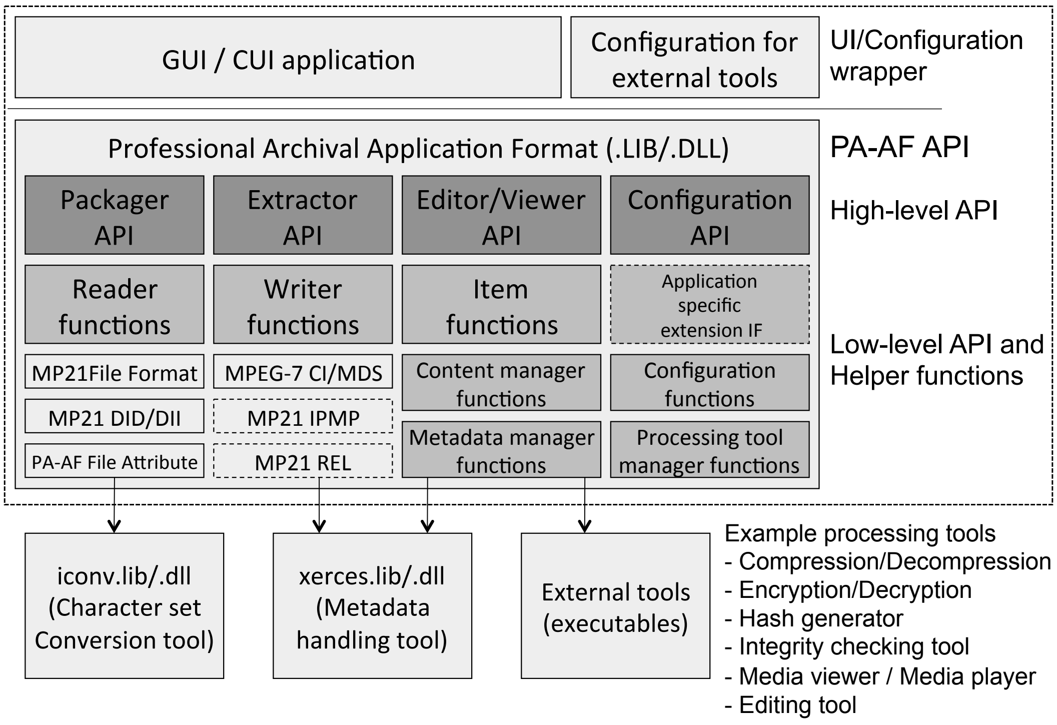
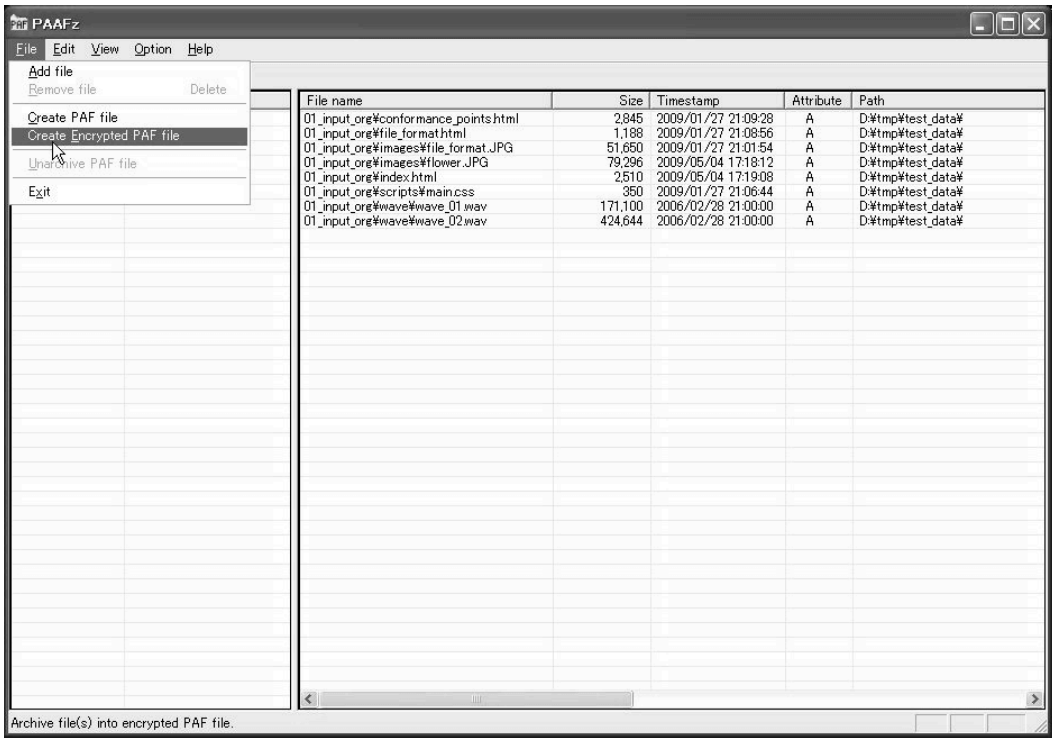

GUI / CUI application
Configuration for external tools
UI/Configuration wrapper
Professional Archival Application Format (.LIB/.DLL)
PA-AF API
Packager API
Extractor API
Editor/Viewer API
Configuration API
High-level API
Reader functions
Writer functions
Item functions
Application specific extension IF
Low-level API and Helper functions
MP21File Format
MPEG-7 CI/MDS
Content manager functions
Configuration functions
MP21 DID/DII
MP21 IPMP
Metadata manager functions
Processing tool manager functions
PA-AF File Attribute
MP21 REL
iconv.lib/.dll (Character set Conversion tool)
xerces.lib/.dll (Metadata handling tool)
External tools (executables)
Example processing tools
- Compression/Decompression
- Encryption/Decryption
- Hash generator
- Integrity checking tool
- Media viewer / Media player
- Editing tool

Figure 5.5: Proposed API design of PA-AF LIB/DLL and an open-source implementation.

3. Enables application of any pre-processing tool and post-processing tool
4. Enables application of encryption and decryption tools with IPMP descriptors
5. Enables application of integrity checking tools with IPMP descriptors
6. Annotates sets of MPEG-7 metadata descriptions
7. Enables inclusion of MPEG-21 REL metadata in the .paf package
8. Configures the tool settings using a configuration file

Sample CUI and GUI applications were implemented using the PA-AF API library. Any kind of command line tools specified in the configuration file can be used as pre-processing and post-processing tools. The CUI tool takes some command line options and generates a PA-AF package file and extracts it back to its original files. For the GUI application, users can simply drag the desired files or directories into the file list box; in addition to dragging, users can also add file to be archived using the “File -> Add file” menu. Figure 5.6 shows the screen image of the GUI application.



File name	Size	Timestamp	Attribute	Path
01_input_org¥conformance_points.html	2,845	2009/01/27 21:09:28	A	D:¥tmp¥test_data¥
01_input_org¥file_format.html	1,188	2009/01/27 21:08:56	A	D:¥tmp¥test_data¥
01_input_org¥images¥file_format.JPG	51,650	2009/01/27 21:01:54	A	D:¥tmp¥test_data¥
01_input_org¥images¥flower.JPG	79,296	2009/05/04 17:18:12	A	D:¥tmp¥test_data¥
01_input_org¥index.html	2,510	2009/05/04 17:19:08	A	D:¥tmp¥test_data¥
01_input_org¥scripts¥main.css	350	2009/01/27 21:06:44	A	D:¥tmp¥test_data¥
01_input_org¥wave¥wave_01.wav	171,100	2006/02/28 21:00:00	A	D:¥tmp¥test_data¥
01_input_org¥wave¥wave_02.wav	424,644	2006/02/28 21:00:00	A	D:¥tmp¥test_data¥

Archive file(s) into encrypted PAF file.

Figure 5.6: The PA-AF GUI application.

5.6 Summary

MPEG-A PA-AF is a standardized packaging format for archiving digital files with maximized interoperability on several operating systems, file systems, and multibyte character sets. PA-AF can serve as information packages defined in the OAIS reference model, such as SIP, AIP, and DIP. The Information Package is one of the most important interfaces for maximizing interoperability among several archiving systems. By sharing resources among archiving organizations, it is expected that maintenance costs for archiving can be reduced.

Chapter 6

Applications of Proposed Schemes

6.1 MPEG-A PA-AF and MPEG-4 ALS applied to archiving of recorded audio projects

6.1.1 Introduction

There are strong needs to preserve contents and information in various domain therefore long-term preservation or archiving systems are considered to be implemented [93]. In music industry, a guideline has been set for preserving digital master recordings because of the rapid grows of digital-born contents made by the Digital Audio Workstation (DAW) or audio editing software, such as ProTools [94, 95, 96]. A method of data-quality assurance for writable DVD disks (hereinafter disks) specified for long term data storage, and a data migration method has been standardized in IEC TC100 [97].

In recent years, preserving digital master recording data is getting more important because of the increased number of digital-born contents. The final master mix is the most important content according to the usage guidelines mentioned above. All the content files including inter-mediate files or another version of takes shall be preserved for the future use. In addition, album arts and metadata for recording settings and other additional information are preserved together. In different part of the pre-production and post-production processes, several types of servers in different operating systems (OS) environment are used. For example, DAW software is often installed on Mac PC but data storage system is on Windows server.

DAW software is frequently updated depending on the technology improvement. Interoperability among different version of DAW softwares is limited because of the technology update. An ideal solution for longterm preservation should provide unified handling of data files regardless of file types or the versions of DAW software. There is no suitable single package format that enables seamless data exchange and migration. In the conventional system, Apple Mac Disc iMaGe (DMG) [83] is popular on Mac OS, Tape ARchive (TAR) [82] is used in UNIX/Linux systems, ZIP is used on Windows platform. Only ASCII character-code set is allowed for file names because no interoperability can be expected among different operating systems and file systems.

Metadata handling in the package format is also an important aspect for preservation. Metadata Encoding and Transmission Standard (METS) [84, 85] has been actively developed and used in Libraries but METS uses the base64 coding for packaging binary data. Therefore, other external package format is needed to pack content files into one archive file. As a result, some important functionalities, such as interoperability among OSs and file systems, are not fully provided. There are some other limitations on handling media files when Audio, Video, Picture image files are stored together in an archive package.

The Moving Picture Experts Group (MPEG) has foreseen a need for an interoperable multimedia content archival format for the preservation of multimedia contents and has been working on producing an MPEG-A standard [98, 99] to address this need. As described in Capter 5, an archival information package format has been designed and proposed; the resulting specification was approved as an ISO/IEC standard for the interoperable multimedia content archive format called MPEG-A Professional Archival Application Format (PA-AF) [100, 101]. PA-AF is designed to fulfil general requirements for the archiving shown in Section 5.3. It aims at a standardized solution that offers sustainability, accessibility, and playability of digital contents for maximum interoperability. PA-AF can be regarded as one implementation of the information packages specified by the Open Archival Information System (OAIS) reference model [23].

In order to support various types of input files, including local file types as well as well-known file types, and to maximize interoperability for handling those files, the PA-AF design is based on packaging files with a hierarchical folder structure. In contrast, what kind of content information and metadata should or should not be stored in an archive is up to an archive's own policy or agreements. To give users the freedom to define their own set of metadata, PA-AF allows users to include any kind of metadata in a PA-AF package as a file or files. It also provides a mechanism for linking metadata to a certain object file so that the PA-AF application can handle any application-specific metadata via a standardized interface.

Archiving recorded audio contents is one application domain of the PA-AF. There are many analogue audio recordings in analogue disks and tapes that need to be preserved because there will be no playback devices available in the near future. In addition, it has become popular to store professional-level recording projects as a set of files that contains not only audio tracks but also metadata and other non-audio files, such as, plug-in binaries, notes, and cover-art images.

The first part of this chapter proposes open-source and optimized implementations of PA-AF archiving tools for audio archiving applications. Though the authors have proposed a package format for audio archiving that makes use of MPEG-4 Audio Lossless Coding (ALS) [102, 103, 104], these are the first PA-AF standard-compliant implementations for audio archiving systems [105]. The open-source version of the implementation (Proposed implementation 1) is approved as the PA-AF reference software (RM003). The optimized version of the implementation (Proposed implementation 2) is based on the Proposed implementation 1 but makes use of an optimized MPEG-4 ALS codec library for audio data compression. Other libraries, such as ZLIB and OpenSSL lib are also adopted. The PA-AF specification is extended to support platform specific attributes of Mac OS X while keeping interoperability among different OSs.

Section 6.1.2 overviews the OAIS reference model and Section 6.1.3 shows an overview of the PA-AF standard specification, especially essential for preserving recorded audio projects. Section 6.1.4 introduces the optinized MPEG-4 ALS encoder and decoder. Section 6.1.5 provides a detailed description of open-source and optimized implementations of the PA-AF packaging and un-packaging tool designed for audio archiving. Enhanced metadata schemes for improved interoperability are also described. In section 6.1.6, performance evaluation test results for the proposed implementations for actual audio data, such as ProTools HD projects, are compared to those for Tar-gz, MacDMG and WinZip. Section 6.3 summarizes the first part of this chapter.

6.1.2 OAIS reference model

Open Archival Information System (OAIS) reference model [23] is a framework for understanding and applying concepts necessary for the long-term preservation of digital information (where long term is long enough to be concerned about changing technologies). The reference model addresses a full range of archival information preservation functions including ingest, archival storage, data management, access, and dissemination. It defines a minimal set of responsibilities that must be met for an archive to be called an OAIS, and it also defines a maximal archive to provide a broad set of useful terms and concepts. In the standard, information packages – Submission Information Package (SIP), Archival Information Package (AIP), and Dissemination Information Package (DIP) – are defined. In order to interconnect archiving systems, the interface point should be aligned at a certain level. The interface point might be defined as any of the Information Packages. For the details, see Section 5.2.

6.1.3 Overview of MPEG-A PA-AF

PA-AF is a general packaging format that can serve as an AIP, DIP, and SIP defined in the OAIS reference model. It archives digital files in a PA-AF file. In addition to containing digital files being archived, a PA-AF file also contains information for the preservation of the archived digital files. It preserves file attributes along with hierarchical folder structure information. Any kind of metadata can be stored in a PA-AF file as a content information file. PA-AF is independent from archiving policies and encoding schemes applied to content information. It maximizes interoperability on several OSs, file systems, and multiple-byte character sets.

Advantages offered by PA-AF compared to other conventional data archival software, such as Tar and Zip, are as follows.

- PA-AF provides comprehensive metadata to model context information of files archived in a PA-AF file. Context information plays an important role in understanding what data is being archived in a PA-AF file. Without good context information attached, archived data may have less value or in extreme cases become useless.

- PA-AF provides a flexible mechanism to accommodate context information specific to an application domain. Context information other than one that defined by the PA-AF specification can be included in a PA-AF file.

- PA-AF provides a flexible mechanism to process input files prior to archiving them in a PA-AF file. It does not specify any mandatory pre-processing tools or modules [data compression, data protection, data integrity checking (authentication of originality), and data governance validation checking] prior to archiving Content Information but does specify a mechanism to describe the use of such pre-processing tools or modules. In this way, a PA-AF file creator can choose specific pre-processing tools depending on its application.

- PA-AF supports cross-platform file extraction. Because it preserves the structure and value of original file attributes in a platform-independent way, files archived in a PA-AF file can be extracted and put into any other target platform besides its original one. For example, files archived on Mac OS X can be extracted to Windows seamlessly.

- The PA-AF file design supports separation of metadata and files being archived. A PA-AF file can contain only metadata, while files being archived can be stored in one or more archived files. This feature is applicable for a very large archival system where there are many large files to be archived. The advantage of this design is that it provides a feature for browsing a set of archived collections by accessing only one file (the file that has the metadata) and provides a link to the desired archive file.

A PA-AF file consists of a header and content part. The header part contains what is called Preservation Description Information. The Preservation Description Information is in the XML metadata format and includes Archive Structure Information, Pre-processing Information, and Context Information. The content part contains one or more archived files which are called Content Information. Content Information includes digital data in its original format as input into the PA-AF file or in the format after pre-processing (trans-formatting) with pre-processing tools. Some example categories of pre-processing tool are lossless data compression, reversible data protection, such as encryptions, and removable data attached to the content for usage governance and data integrity, such as checksums, and digital licenses.

In order to maximize the inter-operabilities, the proposed package format models the hierarchical folder structure information in contents descriptor using the MPEG-21 Digital Item Declaration (DID) XML [88].

6.1.4 Opmitization of MPEG-4 ALS

MPEG-4 ALS is an extension of the MPEG-4 audio coding family [14, 19, 33, 34, 58, 59, 106]. The ALS core codec is based on forward-adaptive linear prediction, which offers remarkable compression together with low complexity. Additional features include long-term prediction, multi-channel coding, and compression of floating-point audio material [18, 20, 107, 108, 109]. MPEG-4 ALS also offers much flexibility in terms of the compression-complexity trade-off, ranging from very low-complexity implementations to maximum compression modes, and thus adaptability to different requirements. For the details, see Section 2.4.

In this chapter, an open-source implementation of MPEG-4 ALS, Reference Model 23 (RM23) [110], is used in the Proposed implementation 1 (Open-source). In addition to the open-source version, another version with an optimized implementation, called the Optimized-ALS codec library is also provided. Several optimization algorithms proposed for the MPEG-4 ALS encoder and decoder [111, 112, 113, 114] have been applied to the library. The Optimized-ALS codec library is used in the Proposed implementation 2 (Optimized).

Performance evaluation is conducted with using audio files in the MPEG testing sequences shown in Table 3.7 of Section 3.7. (1) 9 audio files sampled in 192 kHz, 24 bit, 2ch, (2) 15 audio files sampled in 96 kHz, 24 bit, 2ch, (3) 15 audio files sampled in 48 kHz, 24 bit, 2 ch, and (4) 15 audio files sampled in 48 kHz, 16 bit, 2ch. Duration of each file is 30 seconds. The total number of files are 51. Total file size is 749.8 MB.

Table 6.1 shows the performance comparison results of following four lossless coding schemes: (a) MPEG-4 ALS reference software: Reference Model 23 (RM23), (b) Optimized-ALS implementation, (c) WavePack Win32 version 4.70.0 [115], and (d) FLAC version 1.3.1 [43]. Encoding and decoding parameters are set to the default of the each codec.

Experiments are conducted on Windows 7 Professional Service Pack 1, 32 bit, Intel (R) Core (TM) i7-3667U CPU (2.00 GHz, 1.00 GB RAM). "timeit.exe" command is used to obtain the processing time. Compressed ratio is calculated by following equation:

$$Compressed\ ratio\ [\%] = \frac{Compressed\ \ size}{Original\ \ size} \times 100. \tag{6.1}$$

Optimization of the MPEG-4 ALS decoder enables a real-time playback even with an energy-efficient light-weight CPU. This means that more efficient decoder, the battery life of a portable music player got longer. Another example is sorter waiting time for a snap-shot back-up of the music project files being edited. It is important to find the best trade-off between compression performance and processing time.

In those experiments, difference of the compression performance among examined codecs are less than 5 %. Recent research activities on this area are tend to improve the compression-complexity tradeoffs or the compression performance for multi-channel signals. In some cases, lossless compression schemes are applied to non-audio time sequences [116, 117, 118].

Table 6.1: Compression performance and processing time of lossless coding schemes.

	Compressed ratio [%]	Time [sec]	
		Enc.	Dec.
(*a*) MPEG-4 ALS RM23 (Open-source)	45.85 %	21.471	18.003
(*b*) Optimized-ALS	45.85 %	8.262	11.322
(*c*) WavPack Win32 version 4.70.0	53.55 %	14.841	14.433
(*d*) FLAC version 1.3.1	52.98 %	11.526	7.446

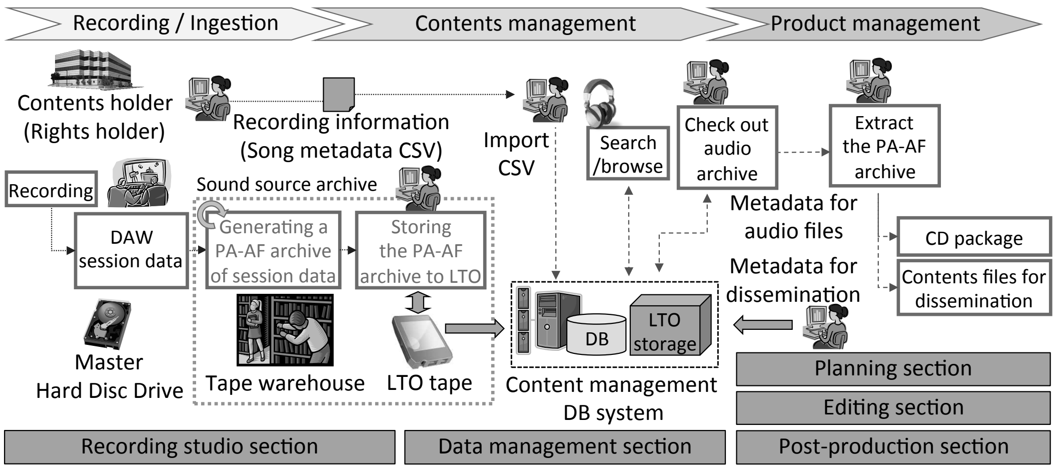


Figure 6.1: Overview of the archiving system and workflow.

6.1.5 Audio archiving format based on MPEG-A PA-AF and MPEG-4 ALS

Figure 6.1 shows the workflow of music production and an overview of a recorded master project archiving system that make use of the proposed information package format.

Sound source are recorded at the recording studio using DAW sound editing software and stored into a master hard disk drive. Recorded session data files are put in a hierarchical folder structure that complies with the folder structure guideline of the recorded audio project. The top folder name is set to the name of the artist. All files are packet into a proposed information package. The generated information package is stored into Linear Tape-Open (LTO) and registered into the content management database. Then, album-art images, artist photo, lyrics, album-metadata for registering to Japanese Society for Rights of Authors, Composers and Publishers (JASRAC), are provided by the post-production section. Once content is registered to the system, dissemination information package for contents download, metadata for search are managed in unified manner. In the proposed information package, MPEG-4 Audio Lossless Coding (ALS) is used as a lossless compression scheme for the audio data files.

Figure 6.2 shows an overview of PA-AF packager/un-packager structure.

Two implementations of the PA-AF packager and un-packager Proposed implementation 1 (Open-source) and Proposed implementation 2 (Optimized) were tested. The implementation 1 (Open-source) is described in Section 5.5 and has been approved as a PA-AF reference software. The source code can be obtained from the PA-AF development website [119]. The proposed imple-

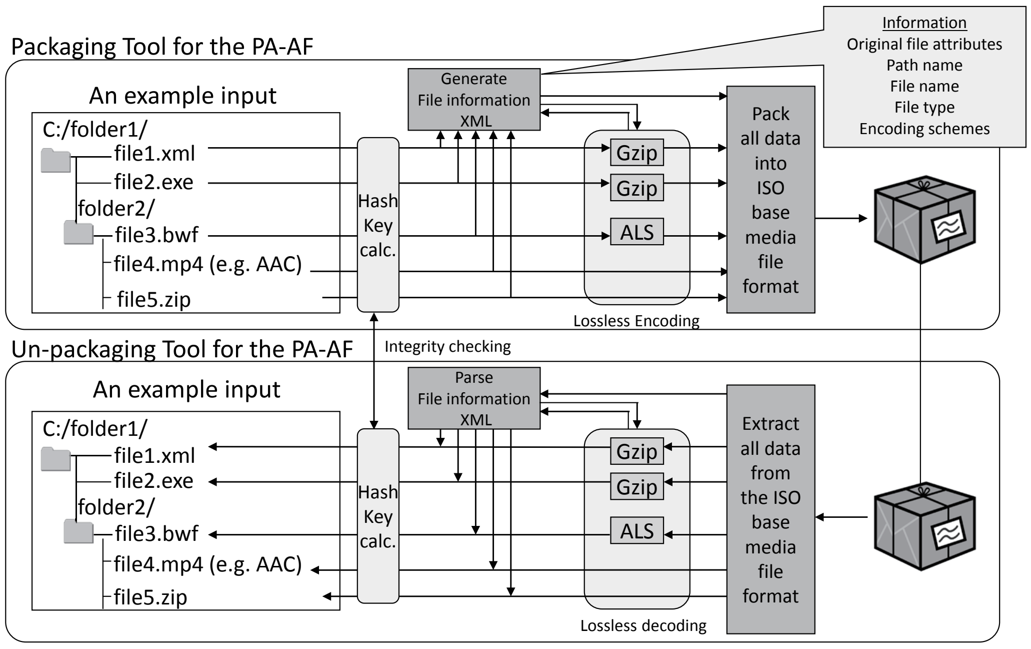


Figure 6.2: Overview of PA-AF packager/un-packager structure.

mentation 2 (Optimized) was developed based on the implementation 1. In this implementation 2, the Optimized-ALS library described in Section 6.1.4 is adopted for audio input files, Zlib is used for the GZIP compression [48, 50, 51] of other input files, and OpenSSL lib [120] for MD5 [121] or SHA-1 [122] hash key generation and Advanced Encryption Standard (AES)128 encryption. By applying those built-in libraries for pre-processing tools, several file copies caused by temporary file creation that had been required in the implementation 1 were reduced.

Tables 6.2 shows examples of MIME types related to file extensions and corresponding compression tools applied for those input files. Table 6.3 shows possible identification values for integrity checking and encryption schemes.

In the proposed implementations, depending on the archiving policy, any compression scheme can be specified by configuration file. In this example configuration, audio files are compressed by using MPEG-4 ALS, other text and binary executable files are compressed by using Gzip. Note that no compression scheme is applied to the files that are already encoded by any lossy or lossless compression scheme. Those files are simply copied into the PA-AF package.

In addition, the implementations were extended to support platform specific attributes of Mac OS X while keeping interoperability among other OSs. The application-specific file attribute in the PA-AF specification was extended by using a newly defined metadata schema. Figure 6.3 shows the extended schema definition in "OSLocalMacAttributes.xsd", which supports extended file attributes on Mac OS X. This new element can be contained in <paaf:UserDefinedAttributes> of the PA-AF file attribute model. With this extension, Mac OS's specific file attributes can be inter-operable among OSs in addition to the existing inter-operability of multiple-byte character set on file names.

The proposed implementation 1 uses Xerces lib [123] for parsing XML structure. Xerces provides fairly good performance and enough flexibility but it turned out that parsing XML using Xerces can be a bottle-neck because a recorded audio project may contain huge number of files. In the proposed implementation 2 (Optimized), the XML parsing algorithm was also improved in order to make the packaging/un-packaging process fast enough for the professional use.

Furthermore, in order to apply MPEG-4 ALS compression to audio input files that don't have appropriate file extension, this optimized PA-AF implementation tries to estimate input file types by applying the MPEG-4 ALS encoder on the first few thousand bytes in the input file.

This optimized implementation has been integrated into a commercial archiving system as a packaging tool.

6.1.6 Performance evaluation

The performance evaluation test was conducted on Windows and Mac OS X. Two ProTools HD projects (Songs Int1 and Int2), and one Nuendo project (Song Float) were used. The projects are professional recordings containing multi-track audio files (48 kHz, 24 bit in Songs Int1 and Int2, and 48 kHz, 32 bit float in Song Float). Some project description files and other files (e.g., video files) are included in the projects along with audio files. Detailed conditions are shown in Table 6.4. Multi-channel sample master data for technology evaluation was provided by courtesy of Memory-Tech Corporation.

Test results are shown in Tables 6.6 and 6.7. Compressed ratio is calculated by Equation 6.1.

On Windows, (a_W) Proposed Implementation 1 (Open-source) [124] with MPEG-4 ALS RM23 [110], was compared to (b_W) Proposed implementation 2 (Optimized), (c_W) WinZip 14.5 [125] with a traditional zip format [.zip], and (d_W) WinZip 14.5 extended with WavPack [115] compression [.zipx]. All tests on Windows were conducted on Microsoft Windows XP Professional x64 Edition Version 2003 Service Pack 2 with an Intel Xeon CPU (5130@2.00 GHz, 2.00 GHz, 3.99 GB RAM). A command "timeit.exe" was used to measure the processing time.

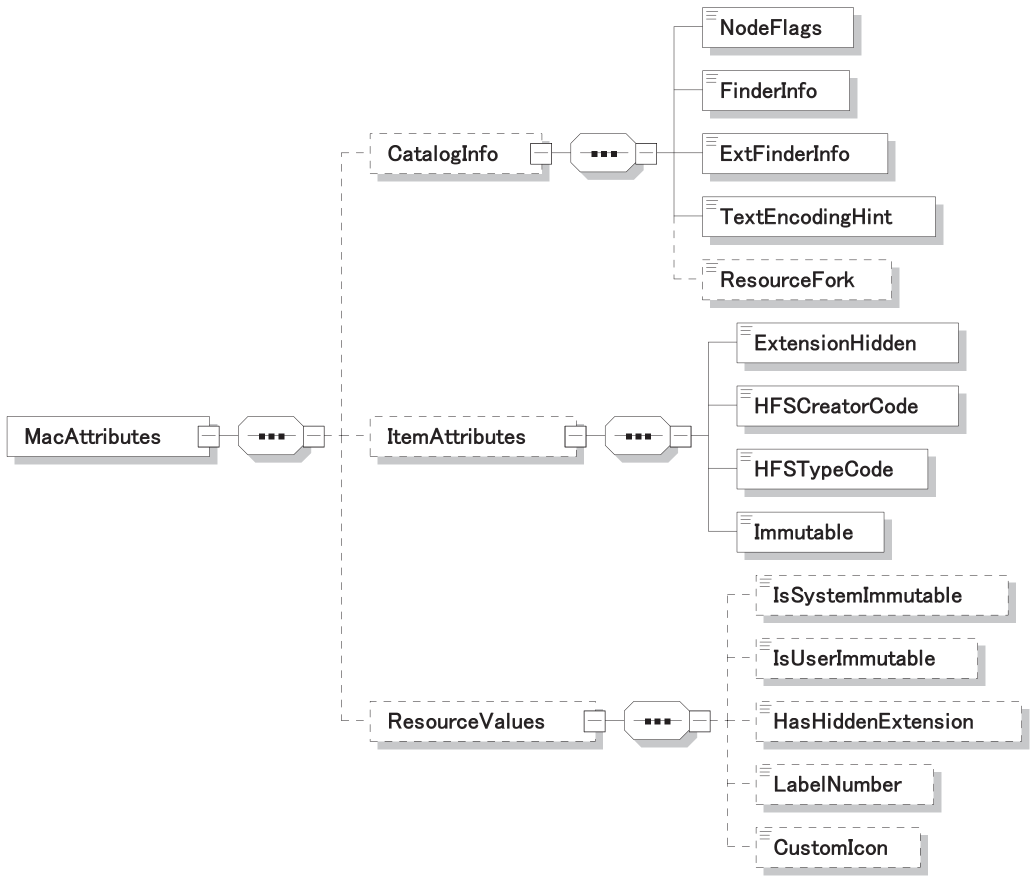


Figure 6.3: Schema definition for extended file attributes on Mac OS X.

On Mac OS X, (a_M) Proposed implementation 1 (Open-source) [124] with MPEG-4 ALS RM23 [110], was compared to (b_M) Proposed implementation 2, (c_M) MacDMG [83], and (d_M) Tar-gz [50, 51, 82]. All tests on the Mac were conducted on Mac OS X Server version 10.4.11 with PowerPC G5 (3.1) (CPU 2.3 GHz (two CPUs), 8 GB RAM). A command "/usr/bin/time -p" was used to measure the processing time.

For the conditions (a_W), (b_W), (a_M) and (b_M), SHA-1 hash-key generation was used for integrity checking. Encoding options for MPEG-4 ALS encoder were set via a PA-AF configuration file. Tenth order linear prediction with Rice coding of prediction residual signal, a Multi-channel coding scheme for stereo, and long-term prediction were used. Default settings were used for other tools. Table 6.5 shows the commands and options used to pack and unpack files.

Table 6.2: Examples of MIME Types related to file extensions and corresponding coding schemes applied to those file types.

File extension names	Corresponding file formats	MIME Types/ MIME SubType	Compression scheme
.bwf, .bwf64	BWF file	audio/x-bwf	x-mp4als
.wav, .wave	WAVE file	audio/x-wav	
.wav64, .w64	Wave64 file	audio/x-wave64	
.aif, .aifa, .aiff	AIFF file	audio/x-aiff	
.mp4, .m4a	MPEG-4 file (AAC, ALS, etc.)	audio/mp4	Identity (No compression)
.als, .mp4	MPEG-4 ALS encoded audio	audio/x-mp4als	
.zip	zip compressed	application/zip	
.gz, .gzip	gzip compressed	application/gzip	
.txt	Plain text	text/plain	x-gzip
.exe	Binary data	application/binary	
Others	Unknown type	application /octet-stream	

Test results show that the processing speed of the Proposed implementation 2 (Optimized) is faster than any of the other tools with the option setting.

The compression performances of the Proposed implementations 1 and 2 are much better than that of WinZip [.zip], Tar-gz [.tgz], and MacDMG [.dmg]. When consider the test result shown in Section 6.1.4, we could expect that the compression performance improvement of the Proposed implementation 2 over WinZip [.zipx] which contains WavPack could be around 5 % but unfortunately, the actual improvement was approximately 1 %. On the other hand, processing speed has been improved by 40 to 60 % for encoding and 68 to 96 % for decoding.

While keeping almost the same compression performance, the encoding time and decoding time of the Proposed implementation 2 became 13 to 41 % and 27 to 46 % of those of Proposed implementation 1 respectively.

Table 6.3: Possible identification values of the integrity checking and encryption schemes.

Identification	Schemes
http://www.w3.org/2000/09/xmldsig#sha1	SHA-1 hash
http://www.w3.org/2001/04/xmldsig-more#md5	MD5 hash
http://www.w3.org/2001/04/xmlenc#tripledes-cbc	Triple DES
http://www.w3.org/2001/04/xmlenc#aes128-cbc	AES128
http://www.w3.org/2001/04/xmlenc#aes192-cbc	AES192
http://www.w3.org/2001/04/xmlenc#aes256-cbc	AES256

The improvement of the Proposed implementation 2 over the Proposed implementation 1 is mainly due to:

1) The optimized MPEG-4 ALS encoder/decoder.

2) Reduced temporary file creation for pre-processing such as SHA-1 hash-key generation, MPEG-4 ALS encoding/decoding, and Gzip encoding/decoding.

3) Improved XML parsing.

Note that the performances of the Proposed implementations 1 and 2 on Windows and Mac OS X were slightly different because the performance of the generated code depends on the the compiler, CPU types, and HDD speed of the target platform.

According to the results of Song Int2 and Song Float shown in Table 6.6, (b_W) Proposed implementation 2 took more time compared to (c_W) WinZip. This is due to:

1) (c_W) WinZip can provide almost no compression therefore the decoding process of WinZip requires only direct copy of binary data in most cases. In contrast, MPEG-4 ALS needs actual decoding process. This may consume some more time.

2) (b_W) Proposed implementation 2 needs to parse XML structure for retrieving the hierarchical folder structure. This parsing is taking time compared to WinZip which has rather primitive and simple data structure for preserving the hierarchical folder structure. Especially when the

Table 6.4: Input files used for the performance avaluation.

	Project file type	Qriginal size	Total number of files (Number of audio files)
Song Int1	ProTools HD	17.06 GB	97 (58)
Song Int2	ProTools HD	70.26 GB	1097 (858)
Song Float	Nuendo 3.0	2.44 GB	1052 (517)

number of files in an archive package exceeds 1000, the parsing overhead becomes significant. This should be improved in the future study.

For the input signals used in the evaluation, the proposed scheme can compress the input data almost the half of its original size. This is not only saving the storage space but also waiting time for transmission (e.g., cloud storage or for relatively slow transmission such as LTO) can be reduced by 50 %.

For example, let's say Song Int2 files are transported from a recording studio to a cloud storage server via a commercial optical fiver network. An average thorough put of an ordinal optical fiver connection of100 Mbit/s should be around 10 MByte/s. It takes around 2 hours when all the 1097 files/70.26 GB in total are sent uncompressed. Total duration of transmission can be less than an hour when Proposed implementation 2 (Optimised) is applied. Here, the encoding process and transmission can be done sequentially with small segments of data frames. Those data segments can be decoded frame by frame as well.

In recent years, the total size of a recorded audio project is getting larger and larger because of the high resolution recoding. Even if the improvement of the network thoroughput and performance of processor continues, still we can expect a benefit from the compression. Sending the contents files in a PA-AF package is much safer choice than sending 1097 files independently thorough via internet connection. Furthermore, error correction information can be added.

Figures 6.4 and 6.5 show that Japanese filenames of Song Int1 packed on Mac OS X were correctly extracted on Windows when PA-AF was used but that they the file names could not be read on Windows when Zip or Tar-gz was used as the packaging format. This is because PA-AF can convert the original multiple-byte character sets to be compliant with that of the target platform. In this example, Japanese characters in UTF-8-Mac on Mac OS X were converted to in Shift JIS on Windows.

Table 6.5: Tools and option settings for the performance evaluation.

Windows XP	
(a_W) Proposed implementation 1 (Open-source)	
Pack	paaftestRM003.exe -i -g -a outfile.paf infolder
Unpack	paaftestRM003.exe -x infile.paf -o outfolder
(b_W) Proposed implementation 2 (Optimized)	
Pack	paaftest.exe -i -g -a outfile.paf infolder
Unpack	paaftest.exe -x infile.paf -o outfolder
(c_W) WinZip [.zip]	
Pack	winzip32.exe -min -a -r outfile.zip infolder
Unpack	winzip32.exe -min -e outfile.zip outfolder
(d_W) WinZip+WavPack [.zipx]	
Pack	winzip32.exe -min -a -r outfile.zipx infolder
Unpack	winzip32.exe -min -e outfile.zipx outfolder
Mac OS X	
(a_M) Proposed implementation 1 (Open-source)	
Pack	paaftestRM003 -i -g -a outfile.paf infolder
Unpack	paaftestRM003 -x infile.paf -o outfolder
(b_M) Proposed implementation 2 (Optimized)	
Pack	paaftest -i -g -a outfile.paf infolder
Unpack	paaftest -x infile.paf -o outfolder
(c_M) MacDMG [.dmg]	
Pack	hdiutil create -ov -srcfolder infolder -format UDZO outfile.dmg

Unpack	hdiutil attatch -mountpoint mountpath infile.dmg;
	cp -R mountpath outfolder;
	hdiutil detatch mountpath

(d_M) Tar-gz [.tgz]	
Pack	tar zcvf outfile.tgz infolder
Unpack	tar zxvf infile.tgz

Table 6.6: Test results on Windows platform.

Song Int1	Comp. size (Comp. ratio)	Time [h:mm:ss]	
		Enc.	Dec.
(a_W) Proposed implementation 1 (Open-source)	8.07 GB (47.31 %)	1:29:48	0:45:23
(b_W) Proposed implementation 2 (Optimized)	8.09 GB (47.41 %)	0:14:48	0:13:57
(c_W) WinZip [.zip]	14.10 GB (82.69 %)	0:35:37	0:22:36
(d_W) WinZip+WavPack [.zipx]	8.13 GB (47.67 %)	0:26:32	0:30:14

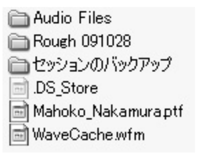
Song Int2	Comp. size (Comp. ratio)	Time [h:mm:ss]	
		Enc.	Dec.
(a_W) Proposed implementation 1 (Open-source)	33.00 GB (46.98 %)	8:26:33	3:32:26
(b_W) Proposed implementation 2 (Optimized)	33.05 GB (47.03 %)	1:03:56	1:17:02
(c_W) WinZip [.zip]	60.92 GB (86.71 %)	2:18:54	0:57:10
(d_W) WinZip+WavPack [.zipx]	33.89 GB (48.23 %)	1:46:45	1:53:37

Song Float	Comp. size (Comp. ratio)	Time [h:mm:ss]	
		Enc.	Dec.
(a_W) Proposed implementation 1 (Open-source)	1.38 GB (56.65 %)	0:25:42	0:08:44
(b_W) Proposed implementation 2 (Optimized)	1.28 GB (52.25 %)	0:04:28	0:03:31
(c_W) WinZip [.zip]	1.67 GB (68.20 %)	0:06:32	0:01:21
(d_W) WinZip+WavPack [.zipx]	1.30 GB (53.50 %)	0:10:08	0:03:40

Table 6.7: Test results on Mac OS X platform.

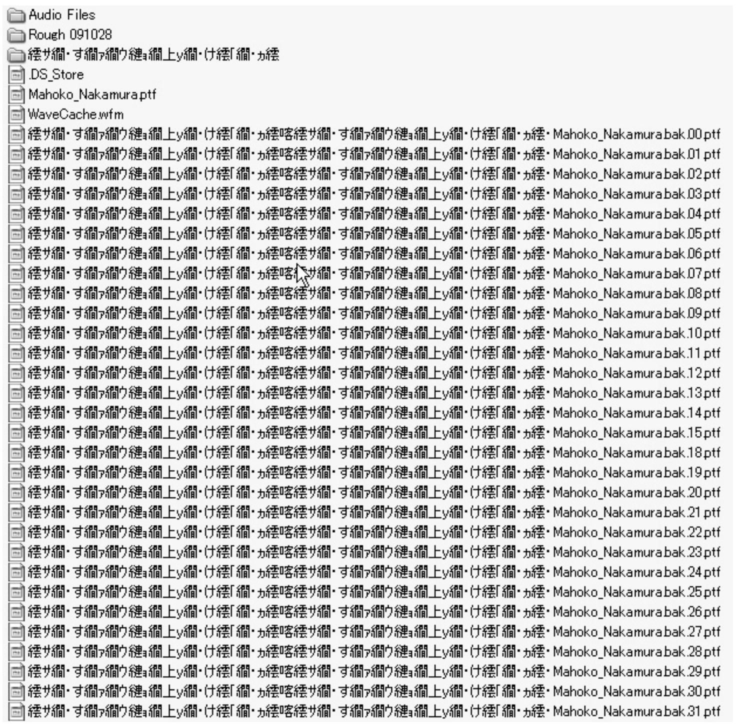
Song Int1	Comp. size (Comp. ratio)	Time [h:mm:ss]	
		Enc.	Dec.
(a_M) Proposed implementation 1 (Open-source)	8.18 GB (47.95 %)	0:39:30	0:36:52
(b_M) Proposed implementation 2 (Optimized)	8.13 GB (47.64 %)	0:15:46	0:14:46
(c_M) MacDMG [.dmg]	14.24 GB (83.48 %)	0:58:05	0:19:33
(d_M) Tar-gz [.tgz]	14.11 GB (82.70 %)	0:39:14	0:16:27

Song Int2	Comp. size (Comp. ratio)	Time [h:mm:ss]	
		Enc.	Dec.
(a_M) Proposed implementation 1 (Open-source)	33.92 GB (48.28 %)	2:38:02	2:39:55
(b_M) Proposed implementation 2 (Optimized)	33.22 GB (47.28 %)	1:13:36	1:07:50
(c_M) MacDMG [.dmg]	61.30 GB (87.25 %)	4:10:02	1:32:34
(d_M) Tar-gz [.tgz]	60.93 GB (86.72 %)	2:37:03	1:12:48



Audio Files
Rough 091028
セッションのバックアップ
.DS_Store
Mahoko_Nakamura.ptf
WaveCache.wfm

Figure 6.4: Mac OS X file names extracted on Windows by PA-AF.



Audio Files
Rough 091028
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧
.DS_Store
Mahoko_Nakamura.ptf
WaveCache.wfm
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.00.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.01.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.02.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.03.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.04.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.05.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.06.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.07.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.08.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.09.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.10.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.11.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.12.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.13.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.14.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.15.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.18.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.19.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.20.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.21.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.22.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.23.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.24.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.25.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.26.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.27.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.28.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.29.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.30.ptf
繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧喀繧ｻ繝・す繝ｧ繝ｳ縺ｮ繝上ャ繝・け繧｢繝・ヵ繧・Mahoko_Nakamura.bak.31.ptf

Figure 6.5: Mac OS X file names extracted on Windows by Zip (Corrupted).

6.2 MPEG-A PA-AF and ITU-T G.711.0 applied to archiving of speech data for telephone customer support system

6.2.1 Introduction

ITU-T Recommendation G.711 is the benchmark standard for narrowband telephony [24]. It has been successful for many decades because of its proven voice quality, ubiquity, and utility. ITU-T Recommendation, G.711.0, has been established for defining a stateless and lossless compression for G.711 packet payloads typically used in VoIP networks. ITU-T Recommendation G.711.0 is also known as ITU-T Recommendation G.711 Annex A, as ITU-T Recommendation G.711 Annex A is effectively a pointer ITU-T Recommendation G.711.0.

Because of the tremendous popularity of the internet, customer service often takes place in the cyber space, such as on web or via e-mail. On the other hand, direct contact to the customers still plays an important roll. Customer support service over telephone-line is one of the most important contact channel for many enterprises. Therefore, there are many telephone contact centre services available all over the world and so that there are many service system designed and used in customer support centers.

Figure 6.6 shows an overview of a customer support service system. Many telephone operators are working in an office (they may be working at home office in some cases) responding calls from customers. During the call, the state-of-the-art supporting system observes the conversation between an operator and a customer, performs speech recognition, and shows expecting list of answer for the ongoing Q&A. In general, a customer who made telephone call may not be fully satisfied with the enterprise's service therefore communicating with the customer is not always easy.

In order to improve service level of the customer-support system, the conversation between the operator and the customer is recorded. Both directions of the conversation (up-link and down-link) are recorded in different audio channels. The speech data is captured in files in G.711 payload format.

This data is analysed and used for:

1) analysing speech conversation to detect the issues, which affect customer satisfaction.

2) improving the support system by providing better speech recognition function.

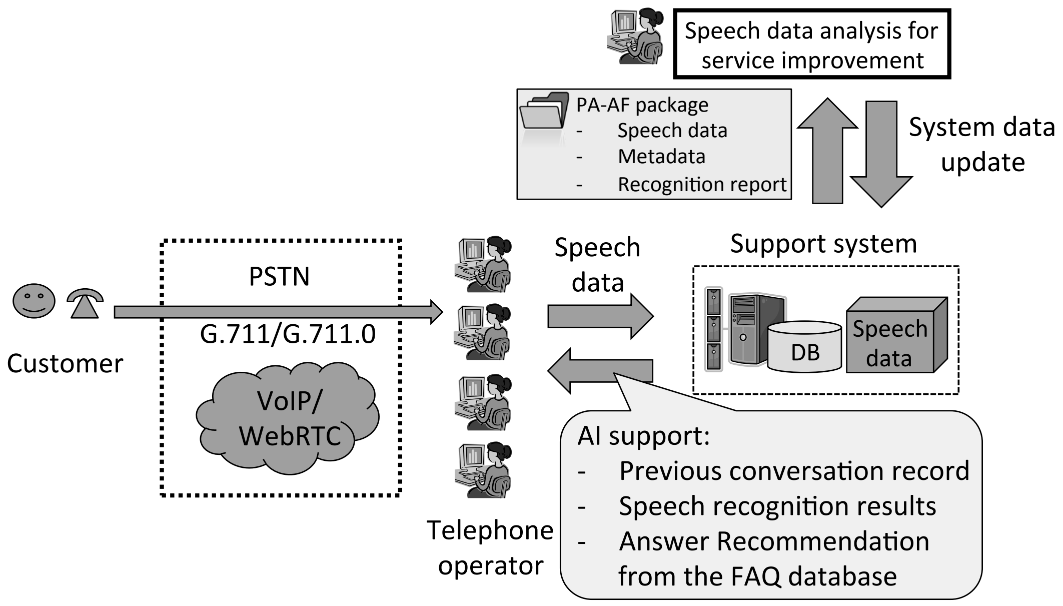


Figure 6.6: Overview of a customer support service system.

On the course of this study, the author has designed an example application for such a telephone customer-support system. In the example system, MPEG-4 PA-AF combined with ITU-T G.711.0 is applied as the information package format such as AIP and DIP. AIP is used for preserving speech data in the storage. DIP is used as the exchange format of speech data among system entities.

Section 6.2.2 shows an RTP payload format for G.711.0 defined in RFC7655 [126]. Proposed decoding algorithm for G.711.0 RTP payload is described in Section 6.2.3. Section 6.2.4 introduces a proposed example profile and the PA-AF package configuration applied for speech data exchange.

6.2.2 RTP payload format for G.711.0

ITU-T Recommendation G.711.0 is used not only for telecommunication such as VoIP or WebRTC but also used for data storage and exchange of speech communication. For this purpose, a G.711.0 storage mode format is defined in addition to the normal RTP payload format in International Engineering Task Force (IETF) Request for Comment (RFC) 7655 [126]. On the course of this study, this author contributed to the RFC7655 RTP payload format for G.711.0 along with some other contributors of ITU-T Recommendation G.711.0.

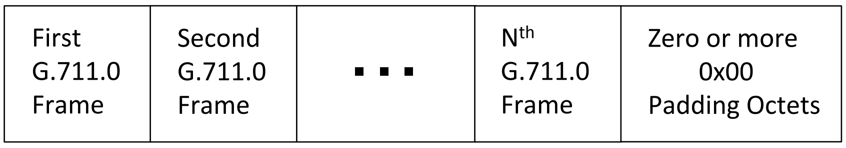


Figure 6.7: One or more G.711.0 frames in RTP payload.

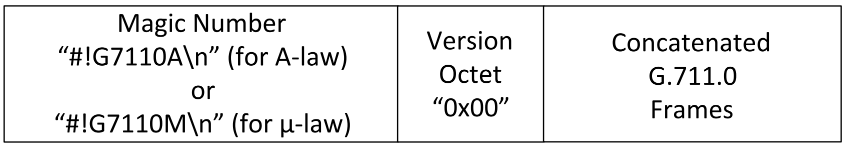

Magic Number “#!G7110A\n” (for A-law) or “#!G7110M\n” (for μ-law)
Version Octet “0x00”
Concatenated G.711.0 Frames

Figure 6.8: G.711.0 storage mode format.

Figure 6.7 shows an overview of the RTP payload format for G.711.0 and Figure 6.8 shows an overview of a storage format for G.711.0.

The storage mode file consists of a magic number and a version octet followed by the individual G.711.0 frames concatenated together. The magic number for G.711.0 A-law corresponds to the ASCII character string "#!G7110A\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x30 0x41 0x0A". Likewise, the magic number for G.711.0 μ-law corresponds to the ASCII character string "#!G7110M\n", i.e., "0x23 0x21 0x47 0x37 0x31 0x31 0x4E 0x4D 0x0A". The version number octet is currently fixed to 0.

6.2.3 G.711.0 RTP payload decoding algorithm

G.711.0 is stateless and has a self-contained property therefore each encoded frame is independent from other frames. In contrast with the traditional fixed bit-rate coders, G.711.0 is the variable length coder therefore the encoded frame size varies in general.

This author proposed a processing algorithm that does not require pre-knowledge of encoded frame size for decoding G.711.0 payload. As described in Section 2.5.2, there is a prefix code indicating a frame size of the frame. This author proposed to include a special prefix code 0x00, which generate 0 G.711 samples as the decoding result. This code can be used as padding octet(s) to be placed in any frame boundaries of the G.711.0 payload.

The frame size of the frame can be extracted by sequentially decoding from the first bit to the last even though several G.711.0 encoded frames are concatenated. Knowing the fact that the minimum code length of a G.711.0 encoded frame is 1 byte and the maximum code length is 321 bytes, the decoding algorithm of the concatenated G.711.0 encoded frames is proposed as follow:

Let N be the number of octets in the RTP payload (i.e., excluding any RTP padding, but including any RTP payload padding), let P equal the number of RTP payload octets processed by the G.711.0 decoding process, let K be the number of G.711 symbols presently in the output buffer, let Q be the number of octets contained in the G.711.0 frame being processed, and let "$! =$" represent not equal to. The keyword "STOP" is used below to indicate the end of the processing of G.711.0 frames in the RTP payload. The algorithm below assumes an output buffer for the decoded G.711 source symbols of length sufficient to accommodate the expected number of G.711 symbols and an input buffer of length 321 octets.

Step 1: Initialization of counters: Initialize P, the number of processed octets counter, to zero. Initialize K, the counter for how many G.711 symbols are in the output buffer, to zero. Initialize N to the number of octets in the RTP payload (including any RTP payload padding). Go to Step 2.

Step 2: Read internal buffer: Read $min\{320+1,(N-P)-1\}$ octets into the internal buffer from the $(P+1)$ octet of the RTP payload. We note at this point, $N-P$ octets have yet to be processed and that $320+1$ octets is the largest possible G.711.0 frame. Also note that in the common case of zero-based array indexing of a $uint8$ array of octets, that this operation will read octets from index P through index $[min\{320+1,(N-P)\}]$ from the RTP payload. Go to Step 3.

Step 3: Analyze the first octet in the internal buffer: If this octet is 0x00 (a padding octet), go to Step 4; otherwise, go to Step 5 (process a G.711.0 frame).

Step 4: Process padding octet (no G.711 symbols generated): Increment the processed packets counter by one (set $P = P + 1$). If the result of this increment results in $P \geq N$, then STOP (as all RTP Payload octets have been processed); otherwise, go to Step 2.

Step 5: Process an individual G.711.0 frame (produce G.711 samples in the output frame): Pass the internal buffer to the G.711.0 decoder. The G.711.0 decoder will read the first octet (called the "prefix code" octet in ITU-T Rec. G.711.0 [24]) to determine the number of source G.711 samples M are contained in this G.711.0 frame. The G.711.0 decoder will produce exactly M G.711 source symbols (M can only have values of 0, 40, 80, 160, 240, or 320). If $K = 0$, these M symbols will be the first in the output buffer and are placed at the beginning of the output buffer. If $K! = 0$, concatenate these M symbols with the prior symbols in the output buffer (there are K prior symbols in the buffer). Set $K = K + M$ (as there are now this many G.711 source symbols in the output buffer). The G.711.0 decoder will have consumed some number of octets, Q, in the internal buffer to produce the M G.711 symbols. Increment the number of payload octets processed counter by this quantity (set $P = P + Q$). If the result of this increment results in $P \geq N$, then STOP (as all RTP Payload octets have been processed); otherwise, go to Step 2.

At this point, the output buffer will contain precisely K G.711 source symbols that should correspond to the ptime signaled if SDP was used and the encoding process was without error. If ptime was signaled via SDP and the number of G.711 symbols in the output buffer is something other than what corresponds to ptime, the packet must be discarded unless other system design knowledge allows for otherwise (e.g., occasional 5 ms clock slips causing one more or one less G.711.0 frame than nominal to be in the payload). Lastly, due to the buffer reads in Step 2 being bounded (to 321 octets or less), N being bounded to the size of the G.711.0 RTP payload, and M being bounded to the number of source G.711 symbols, there is no buffer overrun risk.

It should be noted that the the algorithm above (and the ITU-T G.711.0 reference source code) accommodates padding octets (0x00) placed anywhere between G.711.0 frames in the RTP payload as well as prior to or after any or all G.711.0 frames. The G.711.0 decoder "silently ignores" 0x00 padding octets at the beginning of what it believes to be a frame boundary encoded by G.711.0.

6.2.4 Speech data exchange based on MPEG-A PA-AF and ITU-T G.711.0

An example preservation profile is designed for the telephone customer support system. In order to improve the quality of service, speech recognition system is always updated with the captured speech data. For the update, some speech data which failed the recognition will be analysed as shown in Figure 6.6. Therefore, speech data needs to be sent to the special engineers maintaining the speech recognition engine.

In the example system, all conversation between the customer and the operator is captured as G.711 stream. Note that G.711 μ-law codec is used in the PSTN or VoIP network in Japan. Since telephone conversation is bi-directional, a conversation is decomposed into two streams, one is for up-link and the other is for down-link. A unique identifier (ID) of the conversation session is generated with a random number of N byes represented in Hex.

The ID is used as the main filename string of the captured speech. Actual file name is generated by concatenating the ID plus either of up-link/down-link identifier suffix "_U" or "_D", plus a file extension string representing the coding low, e.g., ".ml8" or ".al8". For example, when $N = 16$, the generated filenames for a call are "0123456789abcdef0123456789abcdef_U.ml8", and "0123456789abcdef0123456789abcdef_D.ul8".

Then, those files are encoded by the G.711.0 encoder and stored in the G.711.0 storage mode format described in 6.2.2. File extension name for the encoded bitstream file is set to ".g7110m", e.g., "0123456789abcdef0123456789abcdef_U.g7110m", and "0123456789abcdef0123456789abcdef_D.g7110m".

The information related to the call is described in a descriptive metadata file, named with the generated random number plus file extension name ".xml", e.g., "0123456789abcdef0123456789abcdef.xml".

Selected files in the Speech data server are packed into a PA-AF package and the package file is used for data exchange. By using MPEG-A PA-AF combined with ITU-T G.711.0, the total file size of speech data is compressed down to 50% of its original size and transmission got much easier than sending separate files independently.

6.3 Summary

In the first half part of this Chapter, two standard-compliant implementations of the PA-AF packaging and un-packaging tool are introduced. Proposed implementation 1 is an open-source version and Proposed implementation 2 is optimized for audio archiving applications. The implementation 2 made use of an optimized MPEG-4 ALS codec library for lossless compression of audio data compression. Other libraries, such as ZLIB and OpenSSL lib are also adopted. Test results show that the proposed archiving tool performs much better than widely used archiving tools such as Tar-gz, MacDMG and WinZip. Compression performance of the proposed PA-AF implementation is equivalent to or much better than other tools while keeping processing speed much faster. The devised PA-AF tool is used in commercial archiving systems in music industry.

In the latter half part of this Chapter, another example application of the proposed archival information package is introduced. In the example, MPEG-A PA-AF combined with ITU-T G.711.0 is applied to archiving of speech data for telephone customer support system. In the customer support system, speech data is efficiently preserved and easily accessed for improving the end-user experience.

In order to apply G.711.0 to the system, a RTP payload format for G.711.0 is designed and proposed as an essential part of the example system. The proposed payload format has been approved as an IETF standard, RFC7655.

Chapter 7
Conclusion

The goal of this study was to provide efficient lossless coding schemes that can be used in real world application. In this dissertation, several new technologies have been proposed in order to support lossless compression of IEEE 754 floating-point represented audio signal and G.711 encoded speech and audio signal. Standardization of the proposed technologies was another important aspect of this study as well as improving the compression performance.

For lossless compression of audio signals represented in IEEE 754 floating-point, a new coding scheme, comprising Approximate Common Factor (ApxCF) coding and the Masked Lempel-Ziv (Masked-LZ) compression, is introduced. In the proposed scheme, an input sequence X is decomposed into three parts: a common multiplier A, a multiplicand sequence Y, and a difference sequence Z. Instead of re-inventing a brand new coding tool, proposed scheme makes use of existing efficient encoding tool for integer input sequences.

The proposed rational approximation scheme provides the good estimation of the ApxCF. It is shown that the ApxCF was found when the common factor exists. In all sound files, appropriate common multipliers were estimated except in the first few frames of some sound files. Thos frames were almost silence. Experimental test results using professional music recording data show that the ApxCF coding can reduce the bit rates considerably, especially when the input values in a frame are constructed by multiplication of the sequence of integer values and a floating-point constant. In addition, the Masked-LZ compression scheme has the potential to reduce bit rates of the difference mantissa. A set of real music recording signal recorded and edited by a professional mixing engineer (96 kHz sampling, 32-bit float, 6 tracks, 20 to 158 sec each) was also tested. The obtained maximum data size reduction was more than 17% for the best case file. The scheme has been accepted as a part of an ISO/IEC standard, MPEG-4 Audio Lossless Coding (ALS).

For lossless compression of log-companded speech signals, the input target is ITU-T G.711 encoded sequence sampled with 8 kHz, 8 bit, 64 Kbit/s. Plus Minus Zero (PMZ) mapping is proposed for the prediction residual calculation in Mapped Domain Linear Prediction (MDLP) and Escaped-Huffman (E-Huffman) coding combined with adaptive recursive Rice coding is proposed for the prediction residual compression. It is shown that the PMZ mapping improves the compression performance by 0.2% for μ-law input. The E-Huffman coding combined with adaptive recursive Rice coding improves the compression by 0.16% averaged for all test conditions, compare to the conventional Rice coding scheme. Average computational complexity is 1.071 WMOPS for the encoder/decoder pair and the worst-case complexity is 1.667 WMOPS in total. These proposed schemes are approved as a part of ITU-T Recommendation G.711.0. The G.711.0 standard provides more than 50% average compression in service provider environments while keeping low computational complexity for the encoder/decoder pair (1.0 WMOPS average, $<$1.7 WMOPS worst case) and low memory footprint (about 5k octets RAM, 5.7k octets ROM, and 3.6k basic operators).

In addition, in order to apply those proposed coding schemes to a long term preservation, an archival information package format was proposed. The proposed archiving format is approved as an ISO/IEC standard: MPEG-A Professional Archival Application Format (PA-AF). PA-AF can serve as information packages defined in the OAIS reference model, such as SIP, AIP, and DIP. The Information Package is one of the most important interfaces for maximizing interoperability among several archiving systems. The type of content information and metadata that should or should not be stored in an archive is up to an archive's own policy or agreements. To give users the freedom to define their own set of metadata, the proposed information package format allows users to include any kind of metadata in a package as application-specific context information. It also provides a mechanism for linking metadata to a certain object file so that applications that adopt the proposed information package format can handle any application-specific metadata via a standardized interface. By sharing resources among archiving organizations, it is expected that maintenance costs for archiving can be reduced.

MPEG-A PA-AF is applied to archiving of recorded audio project. Two standard-compliant implementations of the PA-AF packaging and unpackaging tool are introduced. The implementations made use of MPEG-4 ALS for lossless compression of audio files and Gzip for other input files. Proposed implementation 1 is an open-source version and Proposed implementation 2 is optimized for audio archiving applications in terms of processing speed. An optimized MPEG-4 ALS codec library is applied to the implementation 2. Experimental test results show that the proposed archiving tool with the optimized implementation performs much better than widely used archiving tools such as Tar-gz, MacDMG and WinZip. Compression performance of the proposed PA-AF implementation is equivalent to or much better than other tools while keeping processing speed much faster. The devised PA-AF tool is used in commercial archiving systems in music industry.

Another example application is proposed. MPEG-A PA-AF combined with ITU-T G.711.0 is applied to archiving of speech data for telephone customer support system. In the proposed system, speech data is efficiently preserved and easily accessed for improving end user experience therefore customer satisfaction. In order to apply G.711.0 to the system, a RTP payload format for G.711.0 is designed and proposed as an essential part of the example system. The proposed payload format has been approved as an IETF standard, RFC7655.

By applying the proposed enhancement of lossless coding schemes and the proposed package format, long-term preservation of time domain signals along with related contents and metadata has been made possible.

Appendix A

List of Related Publications and Awards

Journal articles

1. N. Harada, T. Moriya, H. Sekigawa, K. Shirayanagi, and Y. Kamamoto, "Lossless Compression of IEEE754 Floating-point Signal in ISO/IEC MPEG-4 Audio Lossless Coding (ALS)," IEICE Trans. Communications, vol. J89-B, no. 2, pp. 204–213, 2006 (in Japanese).

2. N. Harada, Y. Kamamoto, and T. Moriya, "MPEG-4 Audio Lossless Coding (ALS) Applied to Archiving System of Recorded Audio Project," IPSJ Trans., vol. 57, no. 5, pp. 1355–1364, 2016 (in Japanese).

Magazine

1. N. Harada, Y. Kamamoto, T. Moriya, Hendry, H. Sabirin, and M. Kim, "Archive and Preservation of Media Content Using MPEG-A," IEEE Multimedia Magazine, vol. 17, no. 4, pp. 94–98, 2010.

Conference papers

1. N. Harada, T. Moriya, H. Sekigawa, and K. Shirayanagi, "Lossless Compression of IEEE Floating-point Audio Using Approximate Common Factor Coding and Masked-LZ Compression," in Proc. 118th Audio Engineering Society Convention, no. 6352, Barcelona, Spain, 2005, pp. 1–8.

2. N. Harada, T. Moriya, and Y. Kamamoto, "An Audio Archiving Format Based on the MPEG-4 Audio Lossless Coding (ALS)," in Proc. 121st Audio Engineering Society Convention, no. 6895, San Francisco, CA, USA, 2006, pp. 1–6.

3. N. Harada, Y. Kamamoto, and T. Moriya, "MPEG-A Professional Archival Multimedia Application Format (MAF) Under Development," in Proc. 31st Audio Engineering Society International Conference, no. 20, London, UK, 2007, pp. 1–8.

4. N. Harada, T. Moriya, and Y. Kamamoto, “An Implementation of MPEG-4 ALS Standard Compliant Decoder on ARM Core CPUs,” in Proc. 125th Audio Engineering Society Convention, no. 7625, San Francisco, CA, USA, 2008, pp. 1–6.

5. N. Harada, T. Moriya, and Y. Kamamoto, "MPEG-A Processional Archival Application Format and Its Application for Audio Archiving Combined with MPEG-4 Audio Lossless Coding (ALS)," in Proc. Conference Unlocking Audio 2, London, UK, 2009.

6. N. Harada, Y. Kamamoto, T. Moriya, Y. Hiwasaki, M. A. Ramalho, L. Netsch, J. Stachurski, L. Miao, H. Taddei and F. Qi, "Emerging ITU-T Standard G.711.0 – Lossless Compression of G.711 Pulse Code Modulation," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010), 2010, pp. 4658-4661.

7. N. Harada, Y. Kamamoto and T. Moriya, "Escaped-Huffman and Adaptive Recursive Rice Coding for Lossless Compression of the Mapped Domain Linear Prediction Residual," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010), 2010, pp. 4646-4649.

8. N. Harada, Y. Kamamoto and T. Moriya, "Lossless Compression of Mapped Domain Linear Prediction Residual for ITU-T Recommendation G.711.0," in Proc. Data Compression Conference (DCC2010), 2010, pp. 532–532.

9. N. Harada, Y. Kamamoto, and T. Moriya, "MPEG-A Professional Archival Application Format and Its Application for Audio Archiving," in Proc. 129th Audio Engineering Society Convention, no. 8207, San Francisco, CA, USA, 2010, pp. 1–6.

10. T. Liebchen, T. Moriya, N. Harada, Y. Kamamoto, and Y. A. Reznik, “The MPEG-4 Audio Lossless Coding (ALS) Standard – Technology and Applications,” in Proc. 119th Audio Engineering Society Convention, no. 6589, New York, NY, USA, 2005, pp. 1–14.

11. T. Moriya, N. Harada, and Y. Kamamoto, “An Enhanced Encoder for the MPEG-4 ALS Lossless Coding Standard,” in Proc. 121st Audio Engineering Society Convention, no. 6869, San Francisco, CA, USA, 2006, pp. 1–7.

Reviews

1. N. Harada, T. Moriya, and Y. Kamamoto, “MPEG-4 ALS: Performance, Applications, and Related Standardization Activities,” in NTT Technical Review, vol. 5, no. 12, 2007.

2. T. Moriya, N. Harada, Y. Kamamoto, and H. Sekigawa, "MPEG-4 ALS International Standard for Lossless Audio Coding," in NTT Technical Review, vol. 4, no. 8, 2006, pp. 40–45.

3. Y. Kamamoto, T. Moriya, N. Harada, and C. Kós, "Enhancement of MPEG-4 ALS Lossless Audio Coding," in NTT Technical Review, vol. 5, no. 12, 2007.

Contributed international standards

1. ISO/IEC 14496-3:2005/Amd. 2:2006, Information Technology – Coding of Audio-visual Objects – Part 3: Audio, 3rd Ed. Amendment 2: Audio Lossless Coding (ALS), New Audio Profiles and BSAC Extensions, ISO/IEC Std., 2006.

2. ISO/IEC 14496-3:2009/Amd. 2:2010, Information Technology – Coding of Audio-visual Objects – Part 3: Audio, 4th Ed. Amendment 2: ALS Simple Profile and Transport of SAOC, ISO/IEC Std., 2010.

3. ISO/IEC 14496-5:2001/Amd. 10:2007, Information Technology – Coding of Audio-visual Objects – Part 5: Reference Software, Amendment 10: SSC, DST, ALS and SLS Reference Software, ISO/IEC Std., 2010.

4. ISO/IEC 23000-6:2009, Information Technology – Multimedia Application Format (MPEG-A) – Part 6: Professional Archival Application Format, ISO/IEC Std., Apr, 2009.

5. ISO/IEC 23000-6:2009/Amd. 1:2010, Information Technology – Multimedia Application Format (MPEG-A) – Part 6: Amendment 1: Conformance and Reference Software for Professional Archival Application Format, ISO/IEC Std., 2010.

6. ISO/IEC 23000-6:2012, Information Technology – Multimedia Application Format (MPEG-A) – Part 6: Professional Archival Application Format, 2nd. Ed., ISO/IEC Std., 2012.

7. ISO/IEC 14496-12:2012, Information Technology – Coding of Audio-visual Objects – Part 12: ISO Base Media File Format, 4th Ed., ISO/IEC Std., 2012.

8. ITU-T Rec. G.711.0 – Lossless Compression of G.711 Pulse Code Modulation, ITU-T Std., Oct. 2009.

9. M. Ramalho, P. Jones, N. Harada, M. Perumal, and L. Miao, RFC7655: RTP Payload Format for G.711.0, IETF Std., 2015.

Awards

1. Telecom System Technology Encouraging Award from the Telecommunications Advancement Foundation, Mar. 19, 2007.

2. ISO/IEC Certificate of Appreciation: Project Editor in the development of International Standard "ISO/IEC 14496-5:2001/Amd.10:2007, Information technology – Coding of audio-visual objects – Part 5: Reference software AMENDMENT 10: SSC, DTS, ALS and SLS reference software," from ISO/IEC, Jun. 2008.

3. International Standard Development Award from Information Technology Standard Commission of Japan, Oct. 2008.

4. ISO/IEC Certificate of Appreciation: Project Editor in the development of International Standard "ISO/IEC 14496-3:2005/Amd.8:2008, Information technology – Coding of audio-visual objects – Part 3: Audio, AMENDMENT 8: MP4FF box for original audio file information," from ISO/IEC, Jul. 2009.

5. International Standard Development Award from the Information Technology Standard Commission of Japan, Mar. 2010.

6. International Standard Development Award from the Information Technology Standard Commission of Japan, Jun. 2010.

7. ISO/IEC Certificate of Appreciation: Project Editor in the development of International Standard "ISO/IEC 23000-6:2009, Information technology – Multimedia application format (MPEG-A) – Part 6: Professional archival application format," from ISO/IEC, Jun. 2010.

8. International Standardization Encouragement Award (Director-General, Industrial Science and Technology Policy and Environment Bureau Award) from the Ministry of Economy, Trade and Industry (METI), Japan, Oct. 2011.

9. ISO/IEC Certificate of Appreciation: Project Editor in the development of International Standard "ISO/IEC 23000-6:2009/Amd.1:2010, Information technology – Multimedia application format (MPEG-A) – Part 6: Professional archival application format, AMENDMENT 1: Conformance and reference software for professional archival application format" from ISO/IEC, Nov. 2011.

10. ISO/IEC Certificate of Appreciation: Project Editor in the development of International Standard "ISO/IEC 23000-6:2009/Amd.2, Information technology – Multimedia application format (MPEG-A) – Part 6: Professional archival application format, AMENDMENT 2: Support for large number of files" from ISO/IEC, Nov. 2012.

11. International Standardization Contribution Award from the Information Technology Standard Commission of Japan, May 2016.

References

[1] *ITU-T Rec. G.711 – Pulse Code Modulation (PCM) of Voice Frequencies*, ITU-T Std., 1989. (Cited on pages 1, 2, 5, 6, 7, 16, 47, 48 and 49.)

[2] *ITU-T Rec. G.726 – 40, 32, 24, 16 Kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*, ITU-T Std., 1990. (Cited on page 1.)

[3] S. Miki, K. Mano, T. Moriya, K. Oguchi, and H. Ohmuro, "A Pitch Synchronous Innovation CELP (PSI-CELP) Coder for 2 - 4 Kbit/s," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP1994)*, vol. II, Apr. 1994, pp. II–113–116. (Cited on page 1.)

[4] *ITU-T Rec. G.729 – Coding of Speech at 8 Kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP)*, ITU-T Std., Mar. 1996. (Cited on page 1.)

[5] R. Salami, C. Laflamme, J. P. Adoule, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, S. P. D. Massaloux, P. Kroon, and Y. Shoham, "Design and Description of CS-ACELP: Toll Quality 8 Kb/s Speech Coder," *IEEE Trans. Speech and Audio Proc.*, vol. 6, no. 2, pp. 116–130, Feb. 1998. (Cited on page 1.)

[6] *3GPP TS 26.071 – Mandatory Speech CODEC Speech Processing Functions; Adaptive Multi-Rate (AMR) Speech Codec; General Description*, 3GPP (3rd Generation Partnership Project) Std., 1999. (Cited on page 1.)

[7] *ISO/IEC 13818-7:1997, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information – Part 7: Advanced Audio Coding (AAC)*, ISO/IEC Std., Dec. 1997. (Cited on page 1.)

[8] *ISO/IEC 14496-3:1999, Information Technology – Coding of Audio-Visual Objects – Part 3: Audio*, ISO/IEC Std., Dec. 1999. (Cited on page 1.)

[9] K. Brandenburg, G. Stoll, Y. F. Dehery, J. D. Johnston, L. V. D. Kerkhof, and E. F. Schroeder, "The ISO/MPEG-Audio Codec: A Generic Standard for Coding of High Quality Digital Audio," in *Proc. 92nd Audio Engineering Society Convention*, no. 3336, Vienna, Itary, 1992, pp. 1–22. (Cited on page 1.)

[10] K. Brandenburg and R. Henke, "Near-Lossless Coding of High Quality Digital Audio: First Results," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP1993)*, vol. I, Apr. 27–30, 1993, pp. I–193–196. (Cited on page 1.)

[11] T. Painter and A. Spanias, "A Review of Algorithms for Perceptual Coding of Digital Audio Signals," in *Proc. 13th International COnference on Digital Signal Processing*, vol. 1, Jul. 1997, pp. 179–208 vol.1. (Cited on page 1.)

[12] K. Sayood, *Lossless Compression Handbook.* San Diego, California: Academic Press, 2003. (Cited on pages 1 and 8.)

[13] M. Hans and R. W. Schafer, "Lossless Compression of Digital Audio," *IEEE Signal Processing Magazine*, vol. 18, no. 4, pp. 21–32, Jul. 2001. (Cited on pages 1 and 8.)

[14] *ISO/IEC 14496-3:2005/Amd. 2:2006, Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, 3rd Ed. Amendment 2: Audio Lossless Coding (ALS), New Audio Profiles and BSAC Extensions*, ISO/IEC Std., 2006. (Cited on pages 1, 5, 8, 12 and 81.)

[15] *ISO/IEC 14496-3:2005/Amd. 3:2006, Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, 3rd Ed. Amendment 3: Scalable Lossless Coding (SLS)*, ISO/IEC Std., 2006. (Cited on page 1.)

[16] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI IEEE Std. 754, 1985. (Cited on pages 2, 27 and 28.)

[17] T. Liebchen, Y. A. Reznik, T. Moriya, and D. Yang, "MPEG-4 Audio Lossless Coding," in *Proc. 116th Audio Engineering Society Convention*, no. 6047, Berlin, Germany, May 2004, pp. 1–9. (Cited on page 2.)

[18] N. Harada, T. Moriya, H. Sekigawa, and K. Shirayanagi, "Lossless Compression of IEEE Floating-point Audio Using Approximate Common Factor Coding and Masked-LZ Compression," in *Proc. 118th Audio Engineering Society Convention*, no. 6352, Barcelona, Spain, May 2005, pp. 1–8. (Cited on pages 2, 12 and 81.)

[19] T. Liebchen, T. Moriya, N. Harada, Y. Kamamoto, and Y. A. Reznik, "The MPEG-4 Audio Lossless Coding (ALS) Standard – Technology and Applications," in *Proc. 119th Audio Engineering Society Convention*, no. 6589, New York, NY, USA, Oct. 2005, pp. 1–14. (Cited on pages 2, 5, 8, 12 and 81.)

[20] D. Yang and T. Moriya, "Lossless Compression for Audio Data in the IEEE Floating-Point Format," in *Proc. 115th Audio Engineering Society Convention*, no. 5987, New York, NT, USA, Oct. 2003, pp. 1–5. (Cited on pages 2, 12, 27, 34 and 81.)

[21] D. Yang, T. Moriya, and T. Liebchen, "A Lossless Audio Compression Scheme with Random Access Propoerty," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2004)*, vol. III, Montreal, USA, May 2004, pp. III–1016–1019. (Cited on pages 2, 12, 27 and 34.)

[22] F. Ghido, "An Efficient Algorithm for Lossless Compression of IEEE Float Audio," in *Proc. IEEE Data Compression Conference (DCC2004)*, Utah, USA, Mar. 2004, pp. 429–438. (Cited on pages 2 and 27.)

[23] *ISO 14721:2003, Space Data and Information Transfer Systems – Open Archival Information System – Reference Model*, ISO Std., Mar. 2003. (Cited on pages 2, 3, 5, 24, 64, 65, 78 and 79.)

[24] *ITU-T Rec. G.711.0 – Lossless Compression of G.711 Pulse Code Modulation*, ITU-T Std., Oct. 2009. (Cited on pages 2, 5, 16, 18, 47, 94 and 97.)

[25] S. W. Golomb, "Run-Length Encodings," *IEEE Trans. Information Theory*, vol. 12, pp. 399–401, 1966. (Cited on pages 5, 8, 14 and 51.)

[26] R. F. Rice, "Some Practical Universal Noiseless Coding Techniques – Part I-III," *Jet Propulsion Laboratory Technical Report*, vol. JPL-79-22, JPL-83-17, JPL-91-3, 1979, 1983, 1991. (Cited on pages 5, 8, 14 and 51.)

[27] A. Kiely, "Selecting the Golomb Parameter in Rice Coding," *IPN(Interplanetary Network) Progress Report 42-159*, Nov. 2004. (Cited on pages 5 and 8.)

[28] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952. (Cited on pages 5, 9 and 10.)

[29] T. A. Welch, "A Technique for High-Performance Data Compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984. (Cited on pages 5, 8, 10 and 33.)

[30] H. C. Kotze and G. J. Kuhn, "An Evaluation of the Lempel-Ziv-Welch Data Compression Algorithm," in *Proc. Southern African Conference on Communications and Signal Processing (COMSIG1989)*, Jun. 1989, pp. 65–69. (Cited on pages 5 and 10.)

[31] D. Phillips, "LZW Data Compression," *The Computer Application Journal*, no. 27, pp. 36–48, Jun. – Jul. 1984. (Cited on pages 5, 8 and 10.)

[32] P. Deutsch, *RFC1951: DEFLATE Compressed Data Format Specification Version 1.3*, IETF Std., May 1996, (Category: Informational). (Cited on pages 5 and 10.)

[33] *ISO/IEC 14496-3:2009/Amd. 2:2010, Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, 4th Ed. Amendment 2: ALS Simple Profile and Transport of SAOC*, ISO/IEC Std., 2010. (Cited on pages 5, 12 and 81.)

[34] *ISO/IEC 14496-5:2001/Amd. 10:2007, Information Technology – Coding of Audio-Visual Objects – Part 5: Reference Software, Amendment 10: SSC, DST, ALS and SLS Reference Software*, ISO/IEC Std., 2010. (Cited on pages 5, 12 and 81.)

[35] J. Fennick, *Quality Measures and the Design of Telecommunications Systems.* Artech House, 1988. (Cited on page 6.)

[36] *ITU-T Rec. G.191 – Software Tool Library 2005 User's Manual*, ITU-T Std., Aug. 2005. (Cited on pages 6, 7, 52 and 56.)

[37] A. Moffat and A. Turpin, *Compression and Coding Algorithms.* Kluwer Academic Publishers, 2002. (Cited on pages 8, 9 and 10.)

[38] D. Salomon, *Variable-Length Codes for Data Compression.* Springer London, 2007. (Cited on pages 8, 9 and 10.)

[39] D. Salomon, G. Motta, and D. Bryant, *Data Compression: The Complete Reference*, 4th ed. Springer London, 2007. (Cited on pages 8 and 10.)

[40] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977. (Cited on pages 8, 10 and 33.)

[41] J. Ziv and A. Lempel, "Compression of Individual Sequences Via Variable-Rate Coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978. (Cited on pages 8, 10 and 33.)

[42] M. Rodeh, V. R. Pratt, and S. Even, "Linear Algorithm for Data Compression Via String Matching," *Journal of the Association for Computing Machinery*, vol. 28, no. 1, pp. 16–24, Jan. 1981. (Cited on pages 8 and 33.)

[43] FLAC. [Online]. Available: http://xiph.org/flac (Cited on pages 8 and 82.)

[44] T. Robinson, "SHORTEN: Simple Lossless and Near-lossless Waveform Compression," *Cambridge Univ. Eng. Dept. Tech. Rep. 156*, 1994. (Cited on page 8.)

[45] C. E. Shannon, "A Method for the Construction of Minimum-Redundancy Codes," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, Jul. – Oct. 1948. (Cited on page 9.)

[46] R. M. Fano, "The Transmission of Information," *MIT Technical Report, Research Laboratory for Electronics*, no. 65, 1949. (Cited on page 9.)

[47] P. Deutsch and J.-L. Gailly, *RFC1950: ZLIB Compressed Data Format Specification Version 3.3*, IETF Std., May 1996, (Category: Informational). (Cited on page 10.)

[48] Zlib. [Online]. Available: http://www.glib.org/ (Cited on pages 10 and 84.)

[49] P. Deutsch, *Informational RFC1952: GZIP File Format Specification Version 4.3*, IETF Std., May 1996, (Category: Informational). (Cited on page 10.)

[50] *GNU Zip Manual*, Free Software Foundation, Inc., Mar. 2016. [Online]. Available: https://www.gnu.org/software/gzip/manual/gzip.pdf (Cited on pages 10, 84 and 86.)

[51] Gzip. [Online]. Available: http://www.gzip.org/ (Cited on pages 10, 84 and 86.)

[52] T. Berners-Lee, R. Fielding, and H. Frystyk, *RFC1945: Hypertext Transfer Protocol – HTTP/1.0*, IETF Std., May 1996, (Category: Informational). (Cited on page 10.)

[53] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *RFC2616: Hypertext Transfer Protocol – HTTP/1.1*, IETF Std., May 1999, (Category: Standard Track). (Cited on page 10.)

[54] D. Rand, *RFC1962: The PPP Compression Control Protocol (CCP)*, IETF Std., Jun. 1996, (Category: Standard Track). (Cited on page 10.)

[55] J. Woods, *RFC1979: PPP Deflate Protocol*, IETF Std., Aug. 1996, (Category: Informational). (Cited on page 10.)

[56] T. Liebchen, "An Introduction to MPEG-4 Audio Lossless Coding," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2004)*, vol. III, Montreal, USA, May 2004, pp. III–1012–1015. (Cited on pages 12 and 14.)

[57] T. Liebchen and Y. A. Reznik, "MPEG-4 ALS: An Emerging Standard for Lossless Audio Coding," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2004)*, vol. III, Montreal, USA, May 2004, pp. III–1016–1019. (Cited on pages 12 and 14.)

[58] T. Liebchen and Y. A. Reznik, "MPEG-4 ALS: An Emerging Standard for Lossless Audio Coding," in *Proc. IEEE Data Compression Conference (DCC2004)*, Utah, USA, Mar. 2004, pp. 439–448. (Cited on pages 12, 14 and 81.)

[59] T. Liebchen and Y. A. Reznik, "Improved Forward-adaptive Prediction for MPEG-4 Audio Lossless Coding," in *Proc. 118th Audio Engineering Society Convention*, no. 6449, Barcelona, Spain, May 2005, pp. 1–10. (Cited on pages 12 and 81.)

[60] "Call for Proposals on MPEG-4 Lossless Audio Coding," ISO/IEC JTC 1/SC29/WG11 N5040, Klagenfurt, AT, Jul. 2002. (Cited on page 12.)

[61] "Revised Call for Proposals on MPEG-4 Lossless Audio Coding," ISO/IEC JTC 1/SC29/WG11 N5208, Shanghai, China, Oct. 2002. (Cited on page 12.)

[62] Y. A. Reznik, "Coding of Prediction Residual in MPEG-4 Standard for Lossless Audio Coding (MPEG-4 ALS)," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2004)*, vol. III, Montreal, USA, May 2004, pp. III–1024–1027. (Cited on page 14.)

[63] R. Q10/16, *ITU-T SG16 TD-33R1/WP3 Annex Q10.E, Terms of Reference (ToR) and Time Schedule for G.711 Lossless Compression (G.711-LLC), Study Period 2009-2012, Geneva, February 2009 (Source: Rapporteurs Q10/16)*, ITU-T Std., 2009. (Cited on pages 16, 53 and 56.)

[64] N. Harada, Y. Kamamoto, T. Moriya, Y. Hiwasaki, M. A. Ramalho, L. Netsch, J. Stachurski, L. Miao, H. Taddei, and F. Qi, "Emerging ITU-T Standard G.711.0 – Lossless Compression of G.711 Pulse Code Modulation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010)*, Mar. 14–20, 2010, pp. 4658–4661. (Cited on pages 17 and 47.)

[65] N. Harada, Y. Kamamoto, and T. Moriya, "Escaped-Huffman and Adaptive Reqursive Rice Coding for Lossless Compression of the Mapped Domain Linear Prediction Residual," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010)*, Mar. 14–20, 2010, pp. 4646–4649. (Cited on pages 17 and 47.)

[66] T. Moriya, Y. Kamamoto, and N. Harada, "Enhanced Lossless Coding Tools for Prediction Residual," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010)*, Mar. 14–20, 2010, pp. 4690–4693. (Cited on pages 17 and 47.)

[67] Y. Kamamoto, T. Moriya, and Y. Kamamoto, "Low-complexity PARCOR Coefficient Quantizer and Prediction Order Estimator for Lossless Coding," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010)*, Mar. 14–20, 2010, pp. 4678–4681. (Cited on pages 17 and 47.)

[68] J. Stachurski and L. Netsch, "Fractional-bit and Value-location Lossless Encoding in G.711.0 Coder," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2010)*, Mar. 14–20, 2010, pp. 4666–4669. (Cited on pages 17 and 47.)

[69] N. Harada, Y. Kamamoto, and T. Moriya, "Lossless Compression of Mapped Domain Linear Prediction Residual for ITU-T Recommendation G.711.0," in *Proc. IEEE Data Compression Conference (DCC2010)*, Mar. 24–26, 2010, pp. 532–532. (Cited on pages 17 and 47.)

[70] T. Moriya, Y. Kamamoto, and N. Harada, "Enhanced Lossless Coding Tools of LPC Residual for ITU-T G.711.0," in *Proc. IEEE Data Compression Conference (DCC2010)*, Mar. 24–26, 2010, pp. 546–546. (Cited on pages 17 and 47.)

[71] Y. Kamamoto, T. Moriya, and Y. Kamamoto, "Low-Complexity PARCOR Coefficient Quantizer and Prediction Order Estimator for G.711.0," in *Proc. IEEE Data Compression Conference (DCC2010)*, Mar. 24–26, 2010, pp. 4678–4681. (Cited on pages 17 and 47.)

[72] M. L. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2001. (Cited on page 29.)

[73] D. E. Knuth, *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed. Addison-Wesley Professional, Nov. 1997. (Cited on page 29.)

[74] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. A. K. Peters, Ltd., 2002. (Cited on page 29.)

[75] G. H. Hardy, E. M. Wright, D. R. Hearth-Brown, and J. H. Silverman, *An Introduction to the Theory of Numbers*, 6th ed. Oxford University Press, 2008. (Cited on page 32.)

[76] F. Ghido, "Combined Prediction and Residual Coding for Lossless Audio Compression," in *Proc. IEEE Data Compression Conference (DCC2006)*, Mar. 28–30, 2006, p. 449. (Cited on pages 47 and 53.)

[77] F. Ghido and I. Tabus, "Accounting for Companding Nonlinearities in Lossless Compression," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2007)*, vol. I, Honolulu, HI, USA, Apr. 16–20, 2007, pp. I–261–264. (Cited on pages 47 and 53.)

[78] *ITU-T Rec. P.501 – Test Signals for Use in Telephonometry*, ITU-T Std., May 2000. (Cited on page 53.)

[79] "Restricted Languages Multilingual Speech Database 2002," CD-ROM, NTT Advanced Technology Corporation, 2002. [Online]. Available: http://www.ntt-at.com/product/speech2002/ (Cited on page 53.)

[80] "Ambient Nose Database," CD-ROM, NTT Advanced Technology Corporation. [Online]. Available: http://www.ntt-at.com/product/noise-DB/ (Cited on page 53.)

[81] *ITU-T WP3/16 Document AC-0809-Q10-14 – Proposed Processing Plan for the Corpus 11-B of G.711 LLC*, ITU-T Std., Sep. – Oct. 2008. (Cited on pages 53 and 56.)

[82] *GNU Tar Manual*, Free Software Foundation, Inc., May 2016. [Online]. Available: https://www.gnu.org/software/tar/manual/tar.pdf (Cited on pages 63, 77 and 86.)

[83] Apple Disk Image (DMG). [Online]. Available: http://en.wikipedia.org/Apple_Disk_Image/ (Cited on pages 63, 77 and 86.)

[84] *Metadata Encoding & Transmission Standard*, The Library of Congress Std., Apr. 2010. [Online]. Available: http://www.loc.gov/standards/mets/ (Cited on pages 63 and 78.)

[85] *IEEE Recommended Practice for Learning Technology – Metadata Encoding and Transmission Standard (METS) Mapping to the Conceptual Model for Resource Aggregation*, IEEE Std., Dec. 2013. (Cited on pages 63 and 78.)

[86] J. Wilkinson and B. Devlin, "The Material Exchange Format (MXF) and Its Application," *SMPTE Journal*, vol. 111, no. 9, pp. 378–384, Sep. 2002. (Cited on page 63.)

[87] *ISO/IEC 21000-9:2005, Information Technology – Multimedia Framework (MPEG-21) – Part 9: File Format*, ISO/IEC Std., Apr. 2004. (Cited on page 72.)

[88] *ISO/IEC 21000-2:2005, Information Technology – Multimedia Framework (MPEG-21) – Part 2: Digital Item Declaration*, ISO/IEC Std., Oct. 2005. (Cited on pages 72 and 81.)

[89] *ISO/IEC 21000-3:2003, Information Technology – Multimedia Framework (MPEG-21) – Part 3: Digital Item Identification*, ISO/IEC Std., Apr. 2003. (Cited on page 72.)

[90] *ISO/IEC 15938-5:2003, Information Technology – Multimedia Content Description Interface – Part 5: Multimedia Description Schemes*, ISO/IEC Std., May. 2003. (Cited on page 72.)

[91] *ISO/IEC 21000-4:2006, Informatione Technology – Multimedia Framework (MPEG-21) – Part 4: Intellectual Property Management and Protection Components*, ISO/IEC Std., Apr. 2006. (Cited on page 72.)

[92] *ISO/IEC 21000-5:2004, Information Technology – Multimedia Framework (MPEG-21) – Part 5: Rights Expression Language*, ISO/IEC Std., Apr. 2004. (Cited on page 72.)

[93] K. H. Lee, O. Slattery, R. Lu, X. Tang, and V. McCrary, "The State of the Art and Practice in Digital Preservation," *Journal of Research of the National Institute of Standards and Technology*, vol. 107, no. 1, pp. 93–106, Jan. – Feb. 2002. (Cited on page 77.)

[94] "Usage Guideline for Storage Media According to Pro Tools (in Japanese)," Oct. 2008. [Online]. Available: http://www.japrs.or.jp/protools/pdf/unyoukijun300.pdf (Cited on page 77.)

[95] "Folder Structure Guideline of Victor Entertainment Corporation (in Japanese)." [Online]. Available: http://www.japrs.or.jp/protools/pdf/victor&jvc.pdf (Cited on page 77.)

[96] "Folder Structure Guideline of EMI Music Japan Inc. (in Japanese)." [Online]. Available: http://www.japrs.or.jp/protools/pdf/victor&jvc.pdf (Cited on page 77.)

[97] *IEC 62702-1-1:2016, Audio Archive System – Part 1-1: DVD Disk and Data Migration for Long Term Audio Data Storage*, IEC Std., May 2016. (Cited on page 77.)

[98] K. Diepold, F. Pereira, and W. Chang, "MPEG-A: Multimedia Application Formats," *IEEE Multimedia Magazine*, vol. 12, no. 4, pp. 34–41, Oct. – Dec. 2005. (Cited on page 78.)

[99] *ISO/IEC JTC1/SC29/WG11/N10233, MAF Overview*, ISO/IEC Std., Oct. 2008. (Cited on page 78.)

[100] *ISO/IEC 23000-6:2009, Information Technology – Multimedia Application Format (MPEG-A) – Part 6: Professional Archival Application Format*, ISO/IEC Std., Apr. 2009. (Cited on page 78.)

[101] N. Harada, Hendry, H. Sabirin, M. Kim, Y. Kamamoto, and T. Moriya, "Archive and Preservation of Media Contents Using MPEG-A," *IEEE Multimedia Magazine*, vol. 17, no. 4, pp. 94–98, Oct. – Dec. 2010. (Cited on page 78.)

[102] N. Harada, T. Moriya, and Y. Kamamoto, "An Audio Archiving Format Based on the MPEG-4 Audio Lossless Coding (ALS)," in *Proc. 121st Audio Engineering Society Convention*, no. 6895, San Francisco, CA, USA, Oct. 5–8, 2006, pp. 1–6. (Cited on page 79.)

[103] N. Harada, Y. Kamamoto, and T. Moriya, "MPEG-A Professional Archival Multimedia Application Format (MAF) Under Development," in *Proc. 31st Audio Engineering Society International Conference*, no. 20, London, UK, Jun. 25–27, 2007, pp. 1–8. (Cited on page 79.)

[104] N. Harada, T. Moriya, and Y. Kamamoto, "MPEG-A Professional Archival Application Format and Its Application for Audio Archiving Combined with MPEG-4 Audio Lossless Coding (ALS)," in *Proc. Conference Unlocking Audio 2*, London, UK, Mar. 16–17, 2009. (Cited on page 79.)

[105] N. Harada, Y. Kamamoto, T. Moriya, and M. Otsuka, "MPEG-A Professional Archival Application Format and Its Application for Audio Data Archiving," in *Proc. 129th Audio Engineering Society Convention*, no. 8207, San Francisco, CA, USA, Nov. 4–7, 2010, pp. 1–6. (Cited on page 79.)

[106] T. Moriya, N. Harada, Y. Kamamoto, and H. Sekigawa, "MPEG-4 ALS International Standard for Lossless Audio Coding," in *NTT Technical Review*, vol. 4, no. 8, 2006, pp. 40–45. (Cited on page 81.)

[107] Y. Kamamoto, T. Moriya, T.Nishimoto, and S. Sagayama, "Lossless Compression of Multi-Channel Signals Using Inter-Channel Correlation," *IPSJ Jounal*, vol. 46, pp. 1118–1128, May 2005. (Cited on page 81.)

[108] Y. Kamamoto, T. Moriya, T.Nishimoto, and S. Sagayama, “Intra- and Inter-Channel Long-Term Prediction in ISO/IEC MPEG-4 Audio Lossless Coding (ALS),” *IEICE Trans. Communications*, vol. J89-B, no. 2, pp. 214–222, Feb. 2006. (Cited on page 81.)

[109] N. Harada, T. Moriya, H. Sekigawa, K. Shirayanagi, and Y. Kamamoto, “Lossless Compression of IEEE754 Floating-Point Signal in ISO/IEC MPEG-4 Audio Lossless Coding (ALS),” *IEICE Trans. Communications*, vol. J89-B, no. 2, pp. 204–213, Feb. 2006, (in Japanese). (Cited on page 81.)

[110] MPEG-4 Audio Lossless Coding (ALS) Reference Software RM23. [Online]. Available: http://www.nue.tu-berlin.de/fileadmin/fg97/Forschung/Projekte/Beendete_Projekte/MPEG4_ALS/mp4alsRM23.zip (Cited on pages 81, 85 and 86.)

[111] T. Moriya, N. Harada, and Y. Kamamoto, “An Enhanced Encoder for the MPEG-4 ALS Lossless Coding Standard,” in *Proc. 121st Audio Engineering Society Convention*, no. 6869, San Francisco, CA, USA, Oct. 5–8, 2006, pp. 1–7. (Cited on page 81.)

[112] N. Harada, T. Moriya, and Y. Kamamoto, “An Implementation of MPEG-4 ALS Standard Compliant Decoder on ARM Core CPUs,” in *Proc. 125th Audio Engineering Society Convention*, no. 7625, San Francisco, CA, USA, Oct. 2–5, 2008, pp. 1–6. (Cited on page 81.)

[113] N. Harada, T. Moriya, and Y. Kamamoto, “MPEG-4 ALS: Performance, Applications, and Related Standardization Activities,” in *NTT Technical Review*, vol. 5, no. 12, 2007. (Cited on page 81.)

[114] Y. Kamamoto, T. Moriya, N. Harada, and C. Kós, “Enhancement of MPEG-4 ALS Lossless Audio Coding,” in *NTT Technical Review*, vol. 5, no. 12, 2007. (Cited on page 81.)

[115] WavPack Hybrid Lossless Audio Compressor Win32. [Online]. Available: http://www.wavpack.com/ (Cited on pages 82 and 85.)

[116] H. Huang, H. Shu, and R. Yu, “Lossless Audio Compression in the New IEEE Standard for Advanced Audio Coding,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2014)*, Florence, Italy, May 4–9, 2014, pp. 6934–6938. (Cited on page 82.)

[117] F. Ghido and I. Tabus, “Sparse Modeling for Lossless Audio Compression,” *IEEE Trans. ASLP*, vol. 21, no. 1, pp. 14–28, Jan. 2013. (Cited on page 82.)

[118] N. P. Sgouros, I. P. Constantinou, G. K. Loudos, and S. A. Kossida, “Use of the MPEG-4 ALS Architecture and Inter-Channel Prediction for Multi-Channel ECG Coding,” in *Proc. IEEE International Symposium on Signal Processing and Information Technology*, 2007, pp. 754–759. (Cited on page 82.)

[119] "PA-AF Development Website," NTT. [Online]. Available: http://www.brl.ntt.co.jp/cs/mrl/paaf/index.html (Cited on page 83.)

[120] OpenSSL. [Online]. Available: http://www.openssl.org/ (Cited on page 84.)

[121] R. Rivest, *RFC1321: The MD5 Message-Digest Algorithm*, IETF Std., Apr. 1992, (Category: Informational). (Cited on page 84.)

[122] D. E. 3rd and P. Jones, *RFC3174: US Secure Hash Algorithm 1 (SHA1)*, IETF Std., Apr. 1992, (Category: Informational). (Cited on page 84.)

[123] Xerces. [Online]. Available: http://xerces.apache.org/ (Cited on page 85.)

[124] *ISO/IEC 23000-6:2009/Amd.1:2010, Information Technology – Multimedia Application Format (MPEG-A) – Part 6: Conformance and Reference Software for Professional Archival Application Format*, ISO/IEC Std., Apr. 2010. (Cited on pages 85 and 86.)

[125] WinZip. [Online]. Available: http://www.winzip.org/ (Cited on page 85.)

[126] M. Ramalho, P. Jones, N. Harada, M. Perumal, and L. Miao, *RFC7655: RTP Payload Format for G.711.0*, IETF Std., Nov. 2015, (Category: Standard Track). (Cited on page 95.)