

An Introduction to Lossless Audio Compression

Florin Ghido

Department of Signal Processing
Tampere University of Technology

SGN-2306 Signal Compression

Introduction and Definitions

- Digital Audio and Lossless Compression
- Definitions
- Types of Lossless Compressors

Data Preprocessing

- Unused Bits Removal
- Affine Transforms

Signal Prediction

- Fixed Predictors
- Forward Adaptive Prediction
- Quantization of Prediction Coefficients
- Backward Adaptive Prediction
- Stereo Prediction

Entropy Coding

- Entropy Coding and Integer Mappings
- Golomb-Rice Coding
- Adaptive Arithmetic Coding

Further Topics and Results

- Signal Segmentation for Compression
- Compression of Floating-Point Audio
- Compression Results and Comparison
- Further Reading and References

About digital audio

Digital audio is recorded by conversion of the physical sound signal into an electrical signal, which is measured $F = 1/T$ times a second using a discrete number of levels $L = 2^B$. F is called the sampling rate or sampling frequency and B is the number of data bits used to represent a value. Also, there is a number of synchronized audio channels C to recreate the spatial impression. The entire process is reversed when playing sound from a recording. Advantages of digital audio over analog audio:

- ▶ no degradation in time or generation degradation (for copies);
- ▶ advanced digital signal processing possible in digital domain;
- ▶ higher data density compared to analog recording methods;
- ▶ possibility of hierarchical coding, and online distribution.

Digital audio data types

Typical values for F , B , and C are:

- ▶ for F , 8000 Hz (telephony), 16000 Hz (wide band telephony), 22050 Hz (half of CD), 32000 Hz (DAT), 44100 Hz (CD, most used), 48000 Hz (DAT and DVD), 96000 Hz and 192000 Hz (DVD-Audio, HD-DVD, and Blu-ray);
- ▶ for B , 8 bit integer unsigned, 8 bit integer signed, 16 bit integer signed, 24 bit integer signed, 32 bit integer signed (rarely used), and 32 bit (floating-point). All signed values are in two's complement format;
- ▶ for C , 1 channel (mono), 2 channels (stereo), and multichannel which has a larger number of discrete channels (2.1, 3.0, 3.1, 4.0, 5.0, 5.1, 6.1, 7.1) - the suffix indicates if there is a discrete low frequency enhancement (LFE) channel.

Lossless audio compression

Main characteristics:

- ▶ Lossless signal reconstruction (bit identical reconstruction);
- ▶ Expected average compression is 2:1 to 3:1 (for Audio-CDs);
- ▶ Wide area of applications: archiving, transcoding, distribution, online music stores, portable music players, DVD-Audio.

Desired qualities:

- ▶ Low decoder complexity (w.r.t. hardware architecture, ISA);
- ▶ Configurable encoder complexity (from medium to very high);
- ▶ Support for most bit depths (including floating-point data), sampling rates, and number of channels;
- ▶ Scalable, streamable, embeddable in container formats.

Definition of audio signals

Audio signals are discrete time and discrete levels (integer or less commonly floating-point valued) unidimensional data. Typically, for compression the samples are grouped in independent blocks with length N from a fraction of a second up to several seconds; this is done for practical reasons in order to allow seeking to an arbitrary location and to limit damage in case of data corruption. We denote the audio signal, mono or stereo, by:

$$x_0, x_1, x_2, \dots, x_{N-1} \quad (1)$$

$$y_0, y_1, y_2, \dots, y_{N-1} \quad (2)$$

where, for a stereo signal, x_i is the left channel and y_i is the right channel. When processing data in order, x_i is processed before y_i .

Properties of a data compression algorithm

Depending on the type of processing involved in a compression algorithm (either for prediction or entropy coding), we can roughly categorize them into several classes:

- ▶ fixed, where a hard-coded predictor (for example polynomial predictor) or ad-hoc entropy coding parameters are used;
- ▶ semi-fixed, where the predictor or entropy coding parameters are hard-coded, but determined from a large training corpus;
- ▶ forward adaptive (asymmetrical), where the predictor or entropy coding parameters are computed for each small data block, and transmitted as side information;
- ▶ backward adaptive (symmetrical), where the predictor or entropy coding parameters are recomputed from time to time, but only using past data; lookahead may be used to select improved meta-parameters.

Why data preprocessing?

Data preprocessing is a bijective transform (typically from the forward adaptive class) applied to a data block before prediction and entropy coding in order to improve compressibility by taking advantage of special properties of the data:

- ▶ several unused least-significant bits in the values;
- ▶ smaller range of values than possible in the representing data type;
- ▶ some values are not possible, due to processing in the sensor hardware or previous data manipulation;
- ▶ for floating-point, convert to integers and also save additional information needed for reconstruction.

Unused least significant bits removal

This is the simplest and straightforward preprocessing method intended to remove the zero least significant bits common to all the sample values in the block. This situation would happen when the hardware sensor is, for example, 12 bits and data is packed into 16 bit values, or the hardware sensor is 18 or 20 bits and data is packed into 24 bit values.

1. **Set** $mask = 0$.
2. **For** $i \in \{0, \dots, N - 1\}$
 $mask = mask \text{ bitor } x_i$
3. **Set** $unused =$ number of zero least significant bits in $mask$

Affine integer transforms

A more advanced preprocessing method is intended to remove any integer scaling operation, combined with addition of an offset. We try to find the nonnegative integers α , β with $\alpha \geq 2$ and $\beta < \alpha$, such that $x_i = \alpha z_i + \beta$, where z_i is an integer sequence. This includes, as a particular case, the unused bits removal method.

1. **Set** $\alpha = x_1 - x_0$.
2. **For** $i \in \{1, \dots, N - 1\}$
 $\alpha = \text{cmmdc}(\alpha, x_i - x_0)$
3. **Set** $\beta = (x_1 - x_0) \text{ modulo } \alpha$

Prediction and the simplest case

Prediction is used in lossless audio compression in order to decrease the average amplitude of the audio signal, therefore reducing significantly its entropy. The most widely used class of predictors is that of linear predictors, which try to estimate the value of the current sample x_k by means of a linear combination of the M previous samples, $x_{k-1}, x_{k-2}, \dots, x_{k-M}$. M is called the prediction order, and the prediction is denoted by \hat{x}_k .

The simplest predictors are fixed polynomial predictors, which assume the signal can be modeled by a small order polynomial:

- ▶ order 1, $\hat{x}_k = x_{k-1}$
- ▶ order 2, $\hat{x}_k = 2x_{k-1} - x_{k-2}$
- ▶ order 3, $\hat{x}_k = 3x_{k-1} - 3x_{k-2} + x_{k-3}$

Forward adaptive prediction

If we assume the signal is stationary on an interval of 10-50 milliseconds (which corresponds to a number of samples from a few hundreds to a few thousands), we can compute the best predictor for that group of samples (called a frame); it is necessary, however, to save the prediction coefficients as side information (for the decoder), and moreover they must be saved with some limited precision (quantized), in order to minimize the size overhead. There is a tradeoff between prediction performance (sum of squared errors) and precision of the prediction coefficients; our target is to obtain the smallest possible overall compressed size.

Forward adaptive prediction (cont.)

If the frame extends from sample p to sample q , using a predictor \mathbf{w} of order M , the criterion to be minimized is:

$$J(\mathbf{w}) = \sum_{i=p}^q \left(x_i - \sum_{j=0}^{M-1} w_j \cdot x_{i-j-1} \right)^2 \quad (3)$$

The optimal predictor \mathbf{w} can be found using the Levinson-Durbin algorithm. The algorithm takes the autocorrelation vector \mathbf{r} of length $M + 1$, defined as

$$r_j = \sum_{i=p}^q x_i \cdot x_{i-j}, \text{ for } j \in \{0, \dots, M\} \quad (4)$$

where the values x_i outside the defined range are considered zero, and produces a vector of intermediary (reflection) coefficients \mathbf{k} and the prediction coefficients \mathbf{w} .

Quantization of prediction coefficients

The prediction coefficients \mathbf{w} must be quantized to a given number of fractional bits cb , before being saved as side information and used for prediction. We can write for the quantized scaled integer coefficients $\bar{\mathbf{w}}$, the relation

$$\bar{w}_j = \text{floor}(2^{cb} w_j + 0.5) \quad (5)$$

that is, scaling with the corresponding power of 2 and rounding to the nearest integer. A problem with this classical approach is that there is no simple range the values w_j can take, and no clear bound on the penalty in terms of the criterion $J(\mathbf{w})$ due to quantization.

Quantization of prediction coefficients (cont.)

The optimal number of fractional bits cb to be used varies typically between 6 and 12, depending on the frame content.

If we want bounded ranges for the prediction coefficients and a better representation (less prediction penalty for the same amount of side information), the alternative is to quantize and save the reflection coefficients as

$$\bar{k}_j = \text{floor}(2^{cb} k_j + 0.5) \quad (6)$$

There is a bijective mapping between reflection coefficients and prediction coefficients (in fact, the prediction coefficients are normally computed from the reflection coefficients), and we have that $|k_j| \leq 1$. The optimal number of fractional bits cb to be used for reflection coefficients varies between 5 and 8.

Backward adaptive prediction

We can also compute from time to time, or for each sample the best prediction coefficients for recent data and use those coefficients for prediction of the next sample. This approach has the advantage of not requiring to see in advance (no lookahead needed) - can be applied in an online setting, and also no coefficients must be saved as side information. As a downside, the decoder has also to mimic the same computations done by the encoder, which can make the entire process slower.

The most used methods of computing the coefficients are simple adaptive gradient methods (like Least Mean Squares or Normalized Least Mean Squares), or the (Recursive) Least Squares method.

Backward adaptive prediction (cont.)

In the case of adaptive gradient methods, we illustrate below the usage of the Least Mean Squares algorithm for prediction, where μ is a small constant such that $\mu < 1/(2Mr_0)$, as

1. **For** $j \in \{0, \dots, M - 1\}$ // initialize prediction coefficients
 - Set** $w_j = 0$
 - Set** $e_j = x_j$
2. **For** $i \in \{M, \dots, N - 1\}$
 - 2.1 **Set** $p = 0$
 - 2.2 **For** $j \in \{0, \dots, M - 1\}$ // compute prediction for x_i
 - Set** $p = p + w_j \cdot x_{i-j-1}$
 - 2.3 **Set** $e_i = x_i - p$ // e_i is the prediction error
 - 2.4 **For** $j \in \{0, \dots, M - 1\}$ // update prediction coefficients
 - Set** $w_j = w_j + \mu \cdot e_i \cdot x_{i-j-1}$

Backward adaptive prediction (cont.)

In the case of (Recursive) Least Squares method, the criterion

$$J_{LS}(\mathbf{w}) = \sum_{n=0}^i \alpha^{i-n} (x_n - \sum_{j=0}^{M-1} w_j \cdot x_{n-j-1})^2 \quad (7)$$

is used at each sample x_i or from time to time for recomputing the prediction coefficients \mathbf{w} . α is a positive constant with $\alpha < 1$, the exponential forgetting factor, which puts more emphasis in the criterion on the prediction performance in the recent past.

Efficient techniques exist (the Recursive formulation of the Least Squares is one of them) such that the new prediction coefficients are computed taking advantage of the previous coefficients, at much lower complexity than computing them from the start.

Stereo prediction

In the case of a stereo signal, we can obtain better predictions if we predict a sample using the previous values from the same channel and also the previous values (and the current value, if available) from the other channel. The prediction equations for stereo predictor composed of an intrachannel predictor of order M and an interchannel predictor of order L are, for the left channel

$$\hat{x}_i = \sum_{j=0}^{M-1} w_j \cdot x_{i-j-1} + \sum_{j=0}^{L-1} w_{j+M} \cdot y_{i-j-1} \quad (8)$$

and for the right channel (note that x_i can be used for prediction)

$$\hat{y}_i = \sum_{j=0}^{M-1} w'_j \cdot y_{i-j-1} + \sum_{j=0}^{L-1} w'_{j+M} \cdot x_{i-j} \quad (9)$$

Entropy Coding and Bijective Integer Mappings

Entropy coding is used to take advantage of the reduced amplitude of the prediction residuals from the prediction step and from the fact that the prediction residuals are generally distributed according to a symmetrical two sided Laplacian (or geometric) distribution. It is easy to convert the signed prediction residuals to unsigned values, by using the following bijective integer mapping

$$u(x) = \begin{cases} 2x & \text{for } x \geq 0 \\ -2x - 1 & \text{for } x < 0 \end{cases} \quad (10)$$

The signed values $\dots, -2, -1, 0, 1, 2, \dots$ get rearranged to $0, -1, 1, -2, 2, \dots$, and therefore we obtain a one-sided geometric distribution. If x is B bits signed, $u(x)$ is also B bits, but unsigned.

Golomb-Rice coding

Golomb and Golomb-Rice codes are two class of codes, described earlier in the Signal Compression course, which are well suited for coding one-sided geometric distributions.

Golomb codes use a parameter s to divide each value to be coded into bins of equal width s , splitting a value u into $m = \text{floor}(u/s)$ and $l = u \bmod s$. The value of m is coded in unary (i.e. m one bits followed by a zero bit), and l is coded using a special code which uses either $\text{floor}(\log_2(s))$ or $\text{ceil}(\log_2(s))$ bits.

Golomb-Rice codes further limit the value of parameter s to be a power of two $s = 2^p$, eliminating the need for an integer division, and coding l raw using exactly p bits.

Golomb-Rice coding (cont.)

There are two ways of computing s and p , forward-adaptive and backward-adaptive.

The first method is to compute the optimum s or p for each short block of a few hundred samples and save it as side information.

The suggested values are $s \approx \mathbf{E}(u(x))$ and $p \approx \log_2(\mathbf{E}(u(x)))$, but we can also check for values around the estimated integer value, and select the one which provides the best compression.

The second method is to estimate s_i or p_i recursively, by keeping an exponentially weighted running average U_i , as

$$U_{i+1} = \beta U_i + (1 - \beta)u(x_i) \quad (11)$$

where β is a positive constant, the exponential forgetting factor, with $\beta < 1$, and computing $s_i = U_i$ or $p_i = \text{floor}(\log_2(U_i))$.

Adaptive arithmetic coding

This improved method is based on the general principles of Golomb coding. The parameter s_i is estimated recursively, as in Golomb coding. The difference is that l (the least significant part) is coded with equal probability $1/s$, and m (the most significant part) is not coded in unary, instead it is using an adaptive technique.

A counter table with T entries is maintained (typically, $T = 32$). If a value for $m < T - 1$ is encountered, the symbol m from the table is encoded and its counter incremented, followed by raw encoding of l . However, if a value for $m \geq T - 1$ is encountered, the symbol $T - 1$ from the table (escape indicator) is encoded and its counter incremented, followed by raw encoding of u (the entire value).

Adaptive signal segmentation for compression

In the case of forward adaptive prediction, where prediction coefficients are saved as side information, it is important that each frame to be nearly stationary, otherwise the prediction coefficients will be suboptimal. We want to adaptively split the signal into nonequal frames so that the overall compressed size will be minimized (the minimum description length principle).

One method starts from frames of size $2^R G$ (G is the grid size), and checks if compressing the two halves of size $2^{R-1} G$ leads to better compression. If yes, the same process is applied recursively for each half, as long as the resulting frame size is at least G .

Another method considers frames of arbitrary size kG , with $k \leq k_{max}$, the maximum allowed frame size, and uses a dynamic programming algorithm (evaluating a function which gives the size of a continuous frame) to compute the optimal segmentation.

Compression of floating-point audio

Floating-point audio is used in the professional audio workflow to keep intermediate processing results, and also for some audio restoration projects, where no added distortion is wanted. The most used format is the 32 bit IEEE floating-point format, which consists of 1 sign bit s , 8 bits of exponent e , and 23 mantissa bits s . The real value of a finite normalized (not denormalized, infinity, or NaN) floating-point number is $(-1)^s \cdot 1.s \cdot 2^{e-128}$.

The integer based lossless audio compression methods cannot be applied directly to floating-point numbers. One efficient and simple method to be able to use the integer based compression methods is to approximate the floating-point values with integers and also save some additional information needed to correctly restore the values.

Standardization status and state of the art

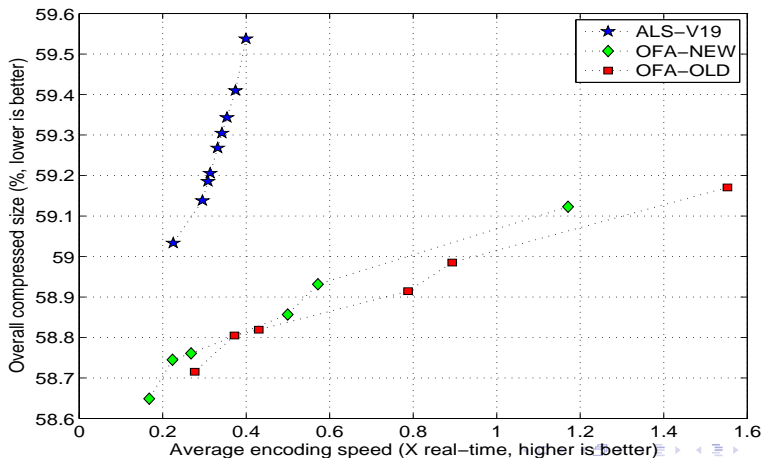
Existing MPEG-4 standards

- ▶ Audio Lossless Coding (ALS) - lossless only
- ▶ Scalable to Lossless (SLS) - AAC core (optional) to lossless
- ▶ Lossless Coding of 1-bit Oversampled Audio (DST) - for DSD

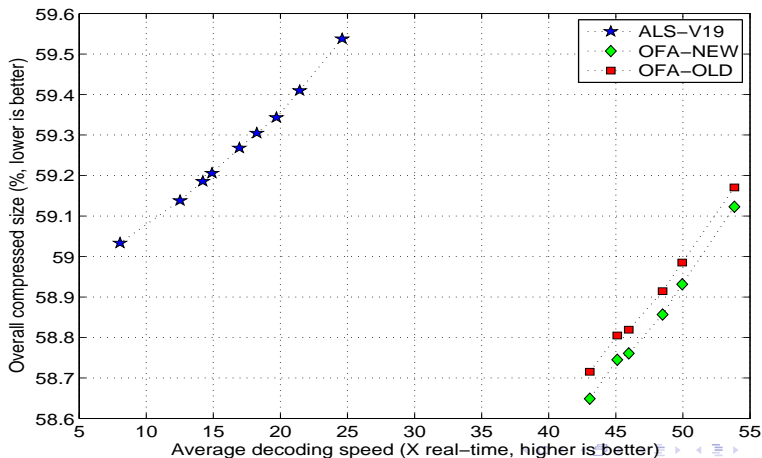
Compression and speed performance of the top two compressors, OptimFROG 4.600ex and MPEG-4 ALS V19, on a private 51.6 GB CD-Audio corpus (consisting of 82 Audio-CDs):

- ▶ mode maximumcompression vs. RLMS best mode, BGMC
 - ▶ 1.04% better compression (52.64% vs. 53.68%)
 - ▶ same encoding speed (0.33 vs. 0.31 real-time)
 - ▶ 6.7 x decoding speed (2.03 vs. 0.30 real-time)
- ▶ mode normal vs. opt. comp., LTP, MCC, max. order 1023
 - ▶ 0.14% better compression (54.37% vs. 54.50%)
 - ▶ 30.5 x encoding speed (7.34 vs. 0.24 real-time)
 - ▶ 2.8 x decoding speed (20.91 vs. 7.38 real-time)

Overall compressed size vs. average **encoding** speed for asymmetrical ALS-V19 and OFR-AS OLD/NEW v0.300






Overall compressed size vs. average **decoding** speed for asymmetrical ALS-V19 and OFR-AS OLD/NEW v0.300



Further reading and references I

-  T. Robinson. SHORTEN: Simple lossless and near-lossless waveform compression. In *Technical Report CUED/F-INFENG/TR.156*, Cambridge University Engineering Department, Cambridge, UK, December 1994.
-  M. Hans and R.W. Schafer, “Lossless Compression of Digital Audio,” *IEEE Signal Processing Magazine*, vol. 18, issue 4, pp. 21-32, July 2001.
-  Simon Haykin, “Adaptive Filter Theory,” Prentice Hall, 4th edition, September 2001.

Further reading and references II

-  F. Ghido, “An Asymptotically Optimal Predictor for Stereo Lossless Audio Compression,” in Proceedings of the Data Compression Conference, pp. 429, Snowbird, Utah, March 2003.
-  T. Liebchen and Y.A. Reznik, “MPEG-4 ALS: an Emerging Standard for Lossless Audio Coding,” in Proceedings of the Data Compression Conference, pp. 439-448, Snowbird, Utah, March 2004.
-  F. Ghido, “OptimFROG Lossless Audio Compressor (symmetrical),” available on Internet at <http://www.LosslessAudio.org/>, July 2006, version 4.600ex.

Further reading and references III

-  F. Ghido and I. Tabus, "Adaptive Design of the Preprocessing Stage for Stereo Lossless Audio Compression," in Proceedings of 122th Audio Engineering Society Convention (AES 122), Convention Paper 7085, Vienna, Austria, May 2007.
-  ISO/IEC, "ISO/IEC 14496-3:2005/Amd 2: 2006, Audio Lossless Coding (ALS), new audio profiles and BSAC extensions," available on Internet at <http://www.nue.tu-berlin.de/mp4als>, November 2007, reference software version RM19.