LOSSLESS COMPRESSION OF G.711 SPEECH USING ONLY LOOK-UP TABLES

Stephen D. Voran

Institute for Telecommunication Sciences
325 Broadway, Boulder, Colorado, USA
svoran@its.bldrdoc.gov

ABSTRACT

The lossless compression algorithm specified in ITU-T Recommendation G.711.0 provides bit-exact G.711 speech coding at reduced bit-rates. We introduce two Look-Up Coders (LUCs) that also offer bit-exact G.711 speech coding at reduced rates but the LUCs do not use arithmetic operations and hence eliminate the need for a processor. Instead they read in eight G.711 symbols, reinterpret those 64 bits to form eight new symbols that carry temporal information, then look up Huffman codes for those new symbols. When compared to G.711.0, LUC rates are 9% to 40% higher and they require 2 to 8 kB additional ROM, but LUCs eliminate about one million weighted arithmetic operations per second. LUCs reduce the 8 b/smpl G.711 rate to 3.8 to 6.7 b/smpl, depending on speech and noise levels.

Index Terms— G.711, G.711.0, lossless compression, PCM, speech coding

1. INTRODUCTION

G.711 is the ubiquitous international toll-quality speech coding standard first established in 1972 [1]. It uses logarithmic quantization to achieve a signal-to-quantization noise ratio (SQR) that is largely independent of speech signal level. Due to masking effects in the human auditory system, this reduces the perceptibility of the quantization noise (compared to uniform quantization). Thus G.711 speech coding can be viewed as an early, simple, yet effective form of perceptual speech coding. Jayant and Noll [2] attribute logarithmic quantization to Holzwarth [3] and report that logarithmic quantization with 8 b/smpl (e.g., G.711 speech coding) provides about the same speech SQR as uniform quantization with 12 b/smpl. G.711 operates on speech sampled at 8000 smpl/s and converts each sample to an eight-bit symbol. The result is a bit-rate of 64 kb/s.

While G.711 is more efficient than uniform quantization (in the rate vs. quality sense), it is not particularly efficient compared to newer speech coders that offer slightly lower quality at rates well below 64 kb/s. This is because G.711 does not exploit temporal correlations found in speech signals, and thus it transmits information that is redundant. In 2009 an additional layer of encoding and decoding was specified in G.711.0 [4], [5]. This layer losslessly compresses the G.711 output, allowing bit-exact G.711 operation at reduced bit-rates. The G.711.0 encoder selects one of 12 different tools for each frame of speech [4]–[9].

These tools include two different forms of linear prediction and numerous tools targeted at low-level and/or sparse speech segments. Five frame sizes of 5, 10, 20, 30, or 40 ms are allowed. There is no look-ahead, so algorithmic delay is the same as frame size. On average, G.711.0 requires one million weighted arithmetic operations per second to encode and decode a single G.711 stream, just meeting the objective set out in the G.711.0 development process [5].

But significant bit-rate reduction can also be achieved without any arithmetic operations. Instead we can use look-up tables (LUTs) that exploit temporal correlations. In Section 2 we describe two Look-Up Coders (LUCs) that read in eight G.711 symbols, reinterpret those 64 bits to form eight new symbols that carry temporal information, then look-up Huffman codes for those symbols. Section 3 shows that this very simple procedure is also very effective—LUCs reduce the 8 b/smpl G.711 rate to 3.8 to 6.7 b/smpl in evaluations that cover different speech levels, background noise levels, transmission filters, and languages.

2. LOOK-UP CODERS

Both the A and μ-law symbols generated by G.711 form sign-magnitude representations for the (amplitude-compressed) speech samples. LUCs can be applied directly to the symbols produced by G.711. But for clarity of presentation it is desired that the magnitude portion of the symbol increase monotonically with the magnitude of the speech sample. This can be accomplished by inverting magnitude bits 1, 3, 5, and 7 of A-law symbols and all magnitude bits of μ-law symbols. This bit inversion is assumed in the following presentation.

2.1. Look-Up Coder 1 (LUC1)

The key to the LUCs is entropy-coding of symbols that carry temporal information. Toward that end LUC1 uses a speech coding frame of $N = 8$ symbols. This value of N is a compromise. Increasing N will slightly decrease bit-rate but will also cause exponential growth in LUT sizes. LUC1 uses no look-ahead, so the total algorithmic delay is 1 ms.

Once eight symbols are available, LUC1 simply reinterprets those 64 bits to form eight new symbols that carry temporal information. This reinterpretation step is most easily described graphically. Figure 1 shows eight G.711 symbols as columns in a time-bit plane and how these 64 bits are reinterpreted as eight new symbols $\{s_l\}_{l=0}^{7}$.

We arrived at this partition of the time-bit plane by evaluating the entropy of numerous different partitions, seeking to minimize total entropy, under the constraint of no more than eight bits per symbol (in order to keep LUT sizes modest). Each symbol is formed from eight bits, but the 2^8 possible symbol values are not equally likely, thus leading to lowered entropy, as reported in Table 1.

For example, s_0 encodes the time history of 8 sign bits. The temporally correlated nature of speech (specifically the low-pass nature) causes sign-bit patterns with fewer transitions to be much more likely than those with more transitions. Using our training database (see Section 3), the two sign-bit patterns with no transitions (all zeros and all ones) accounted for 26% of the probability mass and the 14

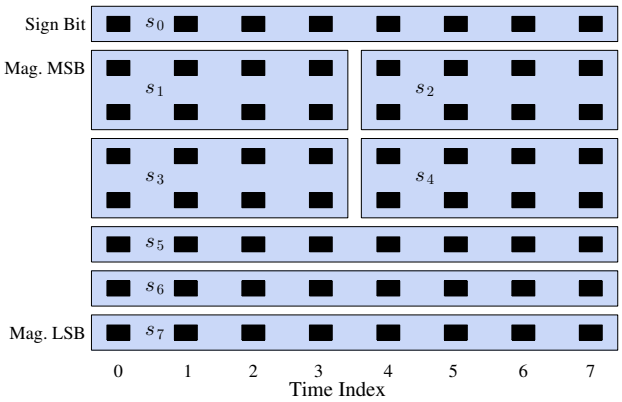

Fig. 1. LUC1 forms eight new symbols from eight G.711 symbols. LUC2 uses these same symbols when in Mode 0.

Symbol	Entropy (b/symbol)	
	A-law	μ-law
s_0	6.2	5.9
s_1	3.3	3.4
s_2	3.3	3.4
s_3	5.8	6.3
s_4	5.8	6.3
s_5	6.6	6.8
s_6	6.9	7.2
s_7	7.2	7.5

Table 1. Entropy for symbols defined in Figure 1 calculated using the training database. Totals are 45.1 or 46.8 bits per 8 symbols for A- and μ-laws respectively. This is 5.6 or 5.9 bits per speech sample. Uncompressed G.711 requires 8.0 b/smpl.

patterns with exactly one transition accounted for 37% of the probability mass. Thus 63% of the probability mass is attributed to only 6% ($(2+14)/2^8$) of the values of s_0.

In general, patterns with greater numbers of transitions are less likely, and this is a direct consequence of the fact that, on average, speech exhibits a low-pass nature. This trend is stronger for the more-significant magnitude bits and weaker for the less-significant magnitude bits. In addition, the more-significant magnitude bits become inactive when the speech level is low. So for those bits, the symbol associated with the all-zeros time history becomes even more likely. Together, these two factors drive the entropy patterns shown in Table 1.

Symbols s_5, s_6, and s_7 also encode time-histories of eight bits. Symbols s_1 through s_4 are different. In this region of the time-bit plane, entropy can be reduced by increasing the amplitude resolution to two bits, even though this requires reducing the time-history to four bits.

The histogram of each symbol leads to a Huffman code that allows the transmission of that symbol at a rate slightly above its entropy [10]. Symbols s_1 and s_2 have the same Huffman code (as do s_3 and s_4). Thus LUC1 requires six distinct Huffman codes, each containing 256 codewords. The Huffman codes are fixed, so the Huffman coding step can be implemented by treating each symbol $\{s_l\}_{l=0}^{7}$ as an eight-bit pointer into an LUT. The LUT returns a variable length codeword and these are concatenated to form the LUC1 output. Because Huffman codewords vary in length, the length of each codeword must also be stored so that the correct number of bits can be concatenated. All ROM sizes reported in this paper include tables of codewords and tables of codeword lengths.

Here is a formal description of this coding process. Consider eight G.711 symbols (covering 1 ms) and let $b_i(j) \in \{0,1\}$ with $0 \le i,j \le 7$ represent the value of the i^{th} bit at the j^{th} sample time ($b_7(j)$ is the value of the sign bit and $b_0(j)$ is value of the least-significant magnitude bit). Now define $f(\boldsymbol{x})$ to be the function that maps a length-eight vector to an integer, $s = f(\boldsymbol{x}) \in [0, 255]$:

$$s = f(\boldsymbol{x}) = \sum_{i=0}^{7} x_i 2^i. \tag{1}$$

Let $H_k(s), 0 \le k \le 5$ be the function that represents the k^{th} Huffman code. This function is defined for integers $s \in [0, 255]$ and it returns a codeword which is a string of M bits, $1 \le M$. Then LUC1 maps eight G.711 symbols to the following eight Huffman codewords, extracted from six distinct Huffman codes:

$$\begin{aligned}
&H_0(\ f(\ [b_7(0), b_7(1), b_7(2), b_7(3), b_7(4), b_7(5), b_7(6), b_7(7)]\ )),\\
&H_1(\ f(\ [b_6(0), b_6(1), b_6(2), b_6(3), b_5(0), b_5(1), b_5(2), b_5(3)]\ )),\\
&H_1(\ f(\ [b_6(4), b_6(5), b_6(6), b_6(7), b_5(4), b_5(5), b_5(6), b_5(7)]\ )),\\
&H_2(\ f(\ [b_4(0), b_4(1), b_4(2), b_4(3), b_3(0), b_3(1), b_3(2), b_3(3)]\ )),\\
&H_2(\ f(\ [b_4(4), b_4(5), b_4(6), b_4(7), b_3(4), b_3(5), b_3(6), b_3(7)]\ )),\\
&H_3(\ f(\ [b_2(0), b_2(1), b_2(2), b_2(3), b_2(4), b_2(5), b_2(6), b_2(7)]\ )),\\
&H_4(\ f(\ [b_1(0), b_1(1), b_1(2), b_1(3), b_1(4), b_1(5), b_1(6), b_1(7)]\ )),\\
&H_5(\ f(\ [b_0(0), b_0(1), b_0(2), b_0(3), b_0(4), b_0(5), b_0(6), b_0(7)]\ )).
\end{aligned} \tag{2}$$

The eight codewords are then concatenated and transmitted or stored in octets or otherwise, as appropriate for the application.

Decoding exactly inverts this encoding. Incoming bits are compared with the stored Huffman codewords in the appropriate Huffman code to identify a match. This can be accomplished using just look-ups. The location of the match is interpreted as an eight-bit symbol value. When values for all eight symbols have been determined, then 64 bits are available and the original G.711 symbols can be found by reinterpreting those 64 bits as shown in Figure 1.

2.2. Look-Up Coder 2 (LUC2)

LUC2 is a simple extension of LUC1. LUC2 codes $N = 8$ symbols at a time, but it can select a different coding mode every 5 ms. Thus we consider the LUC2 speech coding frame size and algorithmic delay to be 5 ms.

Once 5 ms of G.711 symbols (40 symbols) are available, the logical OR operation is applied to the most-significant magnitude bits of those 40 symbols ($b_6(0), b_6(1), ..., b_6(39)$). If the result is logical 0, then the OR operation is applied to the next most-significant magnitude bits ($b_5(0), b_5(1), ..., b_5(39)$). This process continues until either the result is logical 1 or the least-significant magnitude bits have been tested. Through this process LUC2 identifies the highest active bit b_A. Formally, b_A is the largest value of i for which at least one of the 40 magnitude bits ($b_i(0), b_i(1), ..., b_i(39)$) has the logical value 1. If none of the magnitude bits has logical value 1, then $b_A = -1$.

LUC2 compares b_A to two thresholds to select one of three coding modes:

$$\begin{aligned} b_A &= 6, \Rightarrow \text{Mode } 0, \\ 3 \leq b_A &\leq 5, \Rightarrow \text{Mode } 1, \\ b_A &\leq 2, \Rightarrow \text{Mode } 2. \end{aligned} \tag{3}$$

This process forms a very simple but effective speech classifier – Modes 0, 1, and 2 correspond to high, medium, and low-level speech segments respectively. This classification allows LUC2 to use Huffman codes optimized for each of these three classes, thus further reducing bit-rate, at the cost of some additional LUTs. The thresholds in (3) were selected to minimize total symbol entropy.

In Mode 0, LUC2 uses the same symbols as LUC1 (see Figure 1). The symbols used in Modes 1 and 2 are defined in Figures 2 and 3 respectively. In Modes 1 and 2 it is known a priori that some of the higher order bits are all zeros, and thus they do not require any coding.

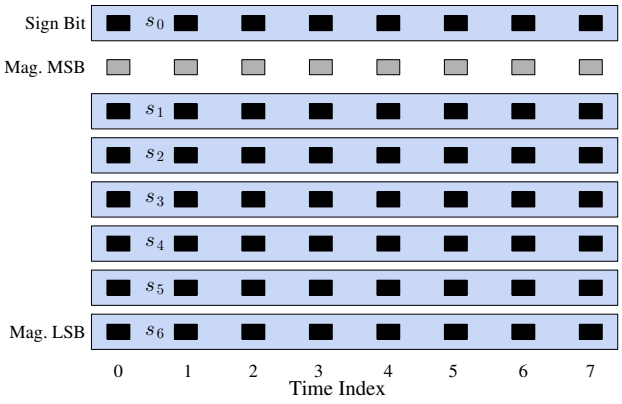

Fig. 2. Symbols used when LUC2 is in Mode 1.

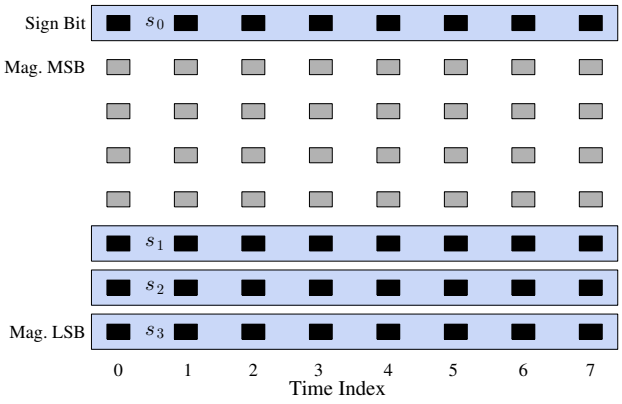

Fig. 3. Symbols used when LUC2 is in Mode 2.

LUC2 also employs a rate-cap feature. If the selected coding mode produces more than 320 bits for the 5 ms frame (8 bits per speech sample), then the original G.711 symbols are sent. This is called Mode 3. LUC2 requires two bits of side information per 5 ms to inform the decoder of the mode. Using our training database, Modes 0, 1, 2, and 3 were selected for 30%, 40%, 29%, and 1% of the 5 ms frames respectively (A-law) or 31%, 43%, 25%, and 1% of the frames respectively (μ-law).

In higher-level speech segments the less-significant magnitude bits do not contain significant temporal structure at the 1 ms timescale; the symbols based on these bits have entropies that are close to 8 bits. Thus in Mode 0, LUC2 sends s_5 - s_7 directly and uses Huffman coding for the other symbols. But in lower-level speech segments those same bits do contain significant temporal structure. Thus in Mode 1, s_5 and s_6 are sent directly and the other symbols are Huffman coded. In Mode 2 all symbols are Huffman coded.

3. EVALUATION

We designed LUC1 and LUC2 through analysis of a training database. This database holds 48 minutes of North American English speech and includes a mixture of different signal levels, signal-to-noise ratios, noise types, and transmission filters.

We used ten testing databases to evaluate LUC1 and LUC2 and there is no overlap between the training and testing databases. The testing databases contain a total of 71 minutes of speech, and various combinations of language, signal levels, signal-to-noise ratios, and transmission filters, as summarized in Table 2. Six noise types are included (bus, car, coffee shop, office, party, street) and a total of 13 female and 13 male speakers are represented. The speech activity factor ranges from 0.49 to 1.00, with a mean value of 0.75. This is higher than the value 0.45 used in the G.711.0 evaluations presented in [5], and this leads to higher bit-rates. Filtering and measurements were done with tools provided in [11].

Language	Filter	Level	SNR	Minutes
North American English	BP	-26 dB	10 dB	10
North American English	BP	-26 dB	20 dB	10
North American English	BP	-16 dB	NA	10
Portuguese	IRS	-26 dB	NA	3
North American English	IRS	-26 dB	NA	10
Italian	IRS	-26 dB	NA	2
North American English	BP	-26 dB	NA	10
German	IRS	-26 dB	NA	3
Japanese	IRS	-26 dB	NA	3
North American English	BP	-36 dB	NA	10

Table 2. Summary of testing databases. "BP" indicates a 200 to 3400 Hz bandpass filter, levels are active speech levels relative to overload, and "NA" indicates no added background noise.

Figure 4 shows the average bit-rate achieved for each of the ten testing databases (with A and μ-law) vs. millions of weighted (arithmetic) operations per second (WMOPS) [11] required for encoding and decoding. For reference, uncompressed G.711 has a bit-rate of 8 b/smpl. The figure provides a visual indication of how bit-rate can be traded-off against operations. The efficient frontier for all 20 cases in this bit-rate vs. operations plane is formed by (left to right) LUC2, then G.711.0 using 5, 10, 30, and 40 ms frames. G.711.0 with 20 ms frames is not on the efficient frontier because both its rate and operations count exceed those of the 40 ms mode. The figure shows wide rate variations between the databases. The database order (top-to-bottom) in Table 2 matches the LUC2 A-law rate order (top-to-bottom) in Figure 4. Databases with low signal and no noise can be coded at lower rates, but those with higher signal and/or noise levels require higher rates. This is true for both LUCs and G.711.0.

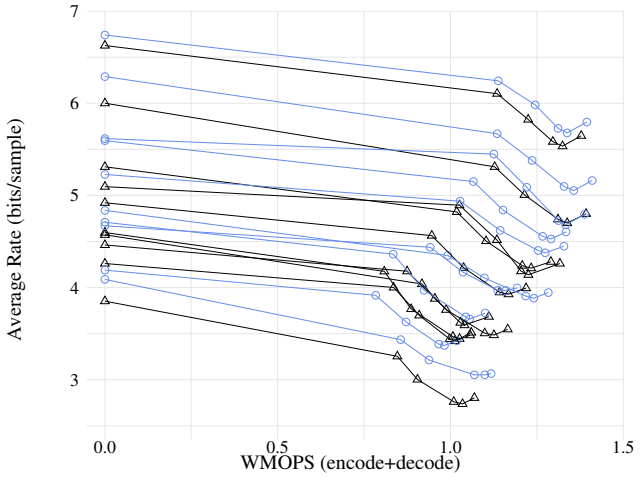

Fig. 4. Average bit-rate for ten databases using A-law (black triangles) and μ-law (blue circles) vs. WMOPS. Data points are (left to right) LUC2, then G.711.0 using 5, 10, 30, 40, and 20 ms frame sizes. Uncompressed G.711 rate is 8 b/smpl.

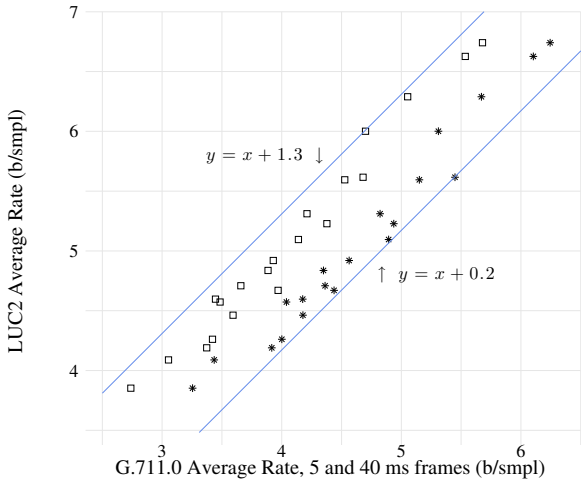

Fig. 5. LUC2 average bit-rate for each database and coding law (A / μ) vs. G.711.0 average bit-rate for 5 ms frames (asterisks) and 40 ms frames (squares). Lines show that LUC2 excess rate is bounded by 0.2 and 1.3 b/smpl.

Figure 5 emphasizes this point by showing LUC2 rates as a function of G.711.0 rates for each database and coding law (A / μ). G.711.0 frame sizes of 5 and 40 ms are used because they represent the highest and lowest average bit-rates for G.711.0. The increase in average bit-rate due to using LUC2 instead of G.711.0 is always between 0.2 and 1.3 bits per sample.

Tables 3 and 4 provide a summary for seven different G.711 lossless compression options using averages calculated across all the speech in the testing databases. G.711.0 uses 5 kB of RAM and 3.6 kB of program memory in addition to the 5.7 kB of ROM listed in the tables [5]. Finally, Table 5 compares both LUCs with G.711.0 at 5 ms (lowest WMOPS) and at 40 ms (lowest bit-rate.) LUCs eliminate one million or more weighted arithmetic operations each second. The cost is 2 or 8 kB of ROM, and a bit-rate increase that ranges from 9% to 40%.

Maximum bit-rate is also a consideration. For G.711.0 using the 5 ms frame size the maximum bit-rate is 8.2 b/smpl. When measured over a 5 ms window, the LUC1 bit-rate is less than 8 b/smpl 95% of the time, less than 8.5 b/smpl 99% of the time, and the absolute maximum observed was 10.1 b/smpl. LUC2 includes the rate-cap feature and thus has a maximum rate (5 ms window) of 8.05 b/smpl.

In packetized speech communication the choice of packet size involves trade-offs. Shorter packets decrease packetization delay, but increase packet header overhead. Multiple speech coding frames can always be combined into a single packet. This means that LUC1 allows any packet size that is a multiple of 1 ms. With LUC2 and G.711.0, packet size can be any multiple of 5 ms. By design, both G.711.0 and LUCs are stateless—a frame can be decoded without reference to any previous frame. This property minimizes the effects of channel losses.

We realize that processors are powerful, inexpensive, and widespread; eliminating one million weighted arithmetic operations per second may not seem important in many G.711 applications. Yet the simplicity and effectiveness of the LUC approach are noteworthy. LUCs can be implemented with a modest amount of simple hardware and thus they offer the opportunity to losslessly reduce G.711 bit-rate in environments without processors.

Coder	Frame Size (ms)	ROM (enc.+dec.) (bytes)	WMOPS		Average Rate (b/smpl)
			enc.	dec.	
G.711.0	5	5721	0.61	0.37	4.63
G.711.0	10	5721	0.68	0.38	4.32
G.711.0	20	5721	0.80	0.43	4.11
G.711.0	30	5721	0.74	0.42	4.06
G.711.0	40	5721	0.76	0.43	4.03
LUC1	1	7680	0.00	0.00	5.63
LUC2	5	13,824	0.00	0.00	5.12

Table 3. Lossless G.711 compression summary for A-law.

Coder	Frame Size (ms)	ROM (enc.+dec.) (bytes)	WMOPS		Average Rate (b/smpl)
			enc.	dec.	
G.711.0	5	5721	0.64	0.38	4.90
G.711.0	10	5721	0.72	0.40	4.61
G.711.0	20	5721	0.84	0.45	4.42
G.711.0	30	5721	0.78	0.44	4.38
G.711.0	40	5721	0.79	0.45	4.35
LUC1	1	7680	0.00	0.00	5.86
LUC2	5	13,824	0.00	0.00	5.36

Table 4. Lossless G.711 compression summary for μ-law.

Comparison	WMOPS Decrease	ROM Increase (bytes)	Rate Increase (b/smpl)
LUC1 cf. G.711.0 at 5 ms	1.0	1959	1.00 / 0.96
LUC1 cf. G.711.0 at 40 ms	1.2	1959	1.60 / 1.51
LUC2 cf. G.711.0 at 5 ms	1.0	8103	0.49 / 0.46
LUC2 cf. G.711.0 at 40 ms	1.2	8103	1.09 / 1.01

Table 5. LUCs compared with G.711.0. Rate increases are for A / μ-law.

4. REFERENCES

[1] *Pulse code modulation (PCM) of voice frequencies*, ITU-T Recommendation G.711, 1988.

[2] N. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice Hall, 1984.

[3] H. Holzwarth, "Pulse code modulation und ihre verzerrungen bie logarithmischer amplitudenquantelung," *Archiv der elektrischen Übertragung*, pp. 277–285, 1949.

[4] *Lossless compression of G.711 pulse code modulation*, ITU-T Recommendation G.711.0, 2009.

[5] N. Harada, Y. Kamamoto, T. Moriya, Y. Hiwasaki, M. 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 *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 2010, pp. 4658 –4661.

[6] J. Stachurski and L. Netsch, "Fractional-bit and value-location lossless encoding in G.711.0 coder," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 2010, pp. 4666 –4669.

[7] T. Moriya, Y. Kamamoto, and N. Harada, "Enhanced lossless coding tools for prediction residual," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 2010, pp. 4690 –4693.

[8] 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 *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 2010, pp. 4646 –4649.

[9] Y. Kamamoto, T. Moriya, and N. Harada, "Low-complexity parcor coefficient quantizer and prediction order estimator for lossless speech coding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Mar. 2010, pp. 4678 –4681.

[10] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[11] *Software tools for speech and audio coding standardization*, ITU-T Recommendation G.191, 2005.