L1-Norm-Based Coding for Lattice Vector Quantization

Wisarn Patchoo*, Thomas R. Fischer† and Curtis Maddex†
*School of Engineering, Bangkok University, Pathumthani, Thailand.
E-mail: wisarn.p@bu.ac.th
†School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA.
E-mail: fischer, cmaddex@eecs.wsu.edu

***Abstract*—A lattice vector quantization encoding method is developed based on ℓ_1-norm-based enumeration and bit-plane coding. The algorithm is implemented for the cubic lattice, can handle arbitrary vector dimension, and is suitable for transform, subband, or wavelet coding applications. Moreover, the algorithm can possibly extend to other binary lattices.**

I. Introduction

Lattice-based vector quantization (LVQ) is a form of structured vector quantization [1] with codevectors selected as a scaled or translated subset of a regular lattice. LVQ does not require storage of codevectors, may allow simple encoding and decoding implementations, and can offer up to 1.53 dB granular gain over scalar quantization (SQ) [2],[3]. In LVQ encoding, a source vector, $\mathbf{x}$, is first mapped to the minimum distortion lattice codevector, $\mathbf{y}$, and then the codevector is losslessly encoded as a binary string for transmission or storage. Lattice codevectors can be enumerated based on their ℓ_2 norm using the lattice theta function [4], or based on their ℓ_1 norm, using the lattice nu function [5]. Spheres and pyramids are convenient shaping regions that can be used to truncate the infinite lattice to a fixed-size codebook in fixed-length coding applications e.g., [6], [7], or used to partition lattice codevectors into spherical or pyramidal sets for enumeration coding in variable-length coding applications. We refer to the combination of lattice vector quantization and ℓ_2-norm-based lossless coding as lattice spherical vector quantization (LSVQ), and the combination of LVQ and ℓ_1-norm-based lossless coding as lattice pyramidal VQ (LPVQ).

Spherical and pyramidal LVQ have been used in a variety of studies and applications. For example, the AMR-WB+ audio coding standard [8] uses the RE_8 lattice VQ [9] to encode transform audio data. In [5] the lattice nu-function is used to encode codevectors formed from wavelet image data. In this work, we focus on LPVQ and the integer (cubic) lattice, $\mathbb{Z}^n$. The approach can be extended to any of the binary lattices [10] using the bit-plane coding approach in [11].

Let $\mathbf{y} \in \mathbb{Z}^n$ be a codevector in the n-dimensional integer lattice and denote $N(n,m)$ as the number of lattice vectors of ℓ_1-norm $m = \sum_i |y_i| = \|\mathbf{y}\|_1$. $N(n,m)$ satisfies the recursion [12]

$$N(n,m) = N(n-1,m) + N(n-1,m-1) + N(n,m-1)$$

with $N(n,1) = 2n$ for $n \geq 1$, and $N(1,m) = 2$ for $m \geq 1$. An enumeration encoding algorithm is developed in [12] for mapping $\mathbf{y}$ with $\|\mathbf{y}\|_1 = m$ into an integer index in the range $\{0, \ldots, N(n,m)-1\}$. This algorithm has been implemented for vector dimension as large as 100 [13], however, large tables are used for fast implementations and the size of $N(n,m)$ grows quickly with n and m (for example, $N(64,1000) \approx 9.5 \times 10^{120}$), so extended-length binary representations of integers are required for implementation.

The present paper develops a bit-plane approach to the coding, rather than the more complex enumeration coding in [12], [13]. The new approach has much less complexity, can handle arbitrary vector dimension, and can be straightforwardly extended to handle ℓ_1-norm-based coding of codevectors in any of the binary lattices [10]. The bit-plane coding is most efficiently implemented using adaptive arithmetic codes with probability models derived for the bit-plane weight. An interesting consequence of the approach is that all bit-planes of the same weight are encoded using the same codeword length, even though the encoding is done using an arithmetic code. This can be useful in data compression applications requiring rate control.

II. Motivation

An integer vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{Z}^n$ can be expressed using bit-plane (sign-magnitude) representation as

$$\mathbf{y} = \begin{bmatrix} b_{s,1} & b_{s,2} & \cdots & b_{s,n} \\ b_{K,1} & b_{K,2} & \cdots & b_{K,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1,1} & b_{1,2} & \cdots & b_{1,n} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_s \\ \mathbf{b}_K \\ \vdots \\ \mathbf{b}_1 \end{bmatrix}, \qquad (1)$$

where the k^{th} column of the matrix is the binary representation of component y_k of $\mathbf{y}$, $\mathbf{b}_s$ is the vector of sign bits and $\mathbf{b}_k$ is the vector of magnitude bits for the k^{th} bit-plane. Define the number of ones in $\mathbf{b}_k$ as the weight of the k^{th} bit-plane, $w_k = wt(\mathbf{b}_k)$. Then, $\|\mathbf{y}\|_1 = \sum_{k=1}^{K} w_k 2^{k-1}$, where $K = 1 + \lfloor \log_2 \|\mathbf{y}\|_1 \rfloor$ is the highest level magnitude bit-plane necessary to represent $\mathbf{y}$.

Assume that random vector $\mathbf{Y}$ has probability mass function $p(\mathbf{y})$, with entropy

$$H(\mathbf{Y}) = -\sum_{y} p(\mathbf{y}) \log_2 p(\mathbf{y}).$$

Let $M = \|\mathbf{Y}\|_1$ and let $\mathbf{B}_k$ denote the k^{th} bit-plane of $\mathbf{Y}$. Since there is a one-to-one correspondence between $\mathbf{Y}$ and $(\mathbf{B}_s, \mathbf{B}_K, \ldots, \mathbf{B}_1)$, and since M is a function of $\mathbf{Y}$, it follows from the properties of conditional entropy [14] that

$$\begin{aligned} H(\mathbf{Y}) &= H(M, \mathbf{B}_s, \mathbf{B}_K, \ldots, \mathbf{B}_1) \\ &= H(M) + H(\mathbf{B}_K|M) + H(\mathbf{B}_{K-1}|M, \mathbf{B}_K) \\ &\quad + \ldots + H(\mathbf{B}_1|M, \mathbf{B}_K, \ldots, \mathbf{B}_2) \\ &\quad + H(\mathbf{B}_s|M, \mathbf{B}_K, \ldots, \mathbf{B}_2, \mathbf{B}_1). \end{aligned} \tag{2}$$

Hence, compared with encoding $\mathbf{Y}$ directly, nothing is lost, in principle, by successively encoding the ℓ_1 norm, M, followed by the bit-planes, provided that the proper conditioning is used to specify the context for the encoding.

Next, let the vector of magnitude bit-plane weights be denoted $\mathbf{w} = (w_1, \ldots, w_K)^T$, with $\mathbf{W}$ the weight vector for random lattice vector, $\mathbf{Y}$. Since the bit-planes weights are a deterministic function of the respective bit-planes, it follows that

$$\begin{aligned} H(\mathbf{Y}) &= H(M, \mathbf{W}, \mathbf{B}_s, \mathbf{B}_K, \ldots, \mathbf{B}_1) \\ &= H(M) + H(\mathbf{W}|M) \\ &\quad + H(\mathbf{B}_K|M, \mathbf{W}) + H(\mathbf{B}_{K-1}|M, \mathbf{W}, \mathbf{B}_K) + \ldots \\ &\quad \ldots + H(\mathbf{B}_1|M, \mathbf{W}, \mathbf{B}_K, \ldots, \mathbf{B}_2) \\ &\quad + H(\mathbf{B}_s|M, \mathbf{W}, \mathbf{B}_K, \ldots, \mathbf{B}_1). \end{aligned} \tag{3}$$

We use (3) to motivate an ℓ_1-norm-based bit-plane coding algorithm that is summarized in the following steps.

Encoding Algorithm

1) Encode $M = \|\mathbf{Y}\|_1$ using a (generally, variable-length) code, $\mathcal{C}_M$, with codeword $c_M(m)$.
2) Encode $\mathbf{W} = (W_1, \ldots, W_K)^T$, using a code, say $\mathcal{C}_W$, conditioned on M. Since
$$\begin{aligned} H(\mathbf{W}|M) = \quad & H(W_K|M) + H(W_{K-1}|M, W_K) + \\ & \cdots + H(W_1|M, W_K, \cdots, W_2), \end{aligned}$$
the coefficients of $\mathbf{W}$ can be efficiently encoded sequentially, provided the proper context is used.
3) Encode the magnitude bit-planes from $k = K$ down to $k = 1$, conditioned on M and W_k. A coefficient sign-bit is encoded immediately after the first binary one is encoded in the "bit stack" representing a coefficient magnitude.

Encoding using steps $1) - 3)$ can produce a bitstream that supports progressive decoding, since the magnitude bit-plane decoding can be truncated at any bit-plane, $\mathbf{b}_k, K \geq k \geq 1$, and a reduced fidelity decoded vector, $\hat{\mathbf{y}}$, generated.

III. ℓ_1-Norm-Based Coding Engine

Encoding using steps $1) - 3)$ requires specification of four codes: A code for encoding the ℓ_1-norm, $M = \|\mathbf{Y}\|_1$; a code for encoding the weight vector, $\mathbf{W}$ conditioned on M; a code for encoding bit-plane $\mathbf{b}_k$, conditioned on M and the bit-plane weight, W_k; and a code for encoding sign-bits. Suitable codes are described as follows.

A. ℓ_1-norm Encoding

To encode the ℓ_1-norm, $m = \|\mathbf{y}\|_1$, the non-negative integer m is mapped to the pair (i, r), where $i = \lfloor \log_2(m+1) \rfloor$ and $r = m - (2^i - 1) \in \{0, \ldots, 2^i - 1\}$, for $i = 0, 1, \ldots, I$, where I is non-negative integer. Based on the mapping, all possible values of m are partitioned into the disjoint non-negative integer sets $S_i = \{2^i - 1, \ldots, 2^{i+1} - 2\}$, and the "overload" set $S_{I+1} = \{2^{I+1} - 1, \ldots\}$, for $i = 0, 1, \ldots, I$. Then, for $m < 2^{I+1} - 1$, m is represented by the pair (i, r), where i is encoded using a variable-length code such as Golumb-type code [15], say $\mathcal{C}_I$ with codeword $c_I(i)$, and r is encoded using a fixed-length i-bit codeword, say $c(r|i)$. For $m \geq 2^{I+1} - 1$, m is represented by a pair $(I + 1, m_1)$, where the index $i = I + 1$ and the value $r = m_1 = m - (2^I - 1)$. A fixed-length L_1-bit code is used to encode the value of m_1, where $L_1 = 1 + \lfloor \log_2 m_1 \rfloor$. The value of L_1 is determined by the application, but $L_1 = 4$ or 8 should be adequate for most cases.

To obtain a better coding efficiency, the variable-length code $\mathcal{C}_I$ and $c(r|i)$ might be further encoded using a context-based code such as an adaptive arithmetic code. Since the value of m reflects the (ℓ_1) energy in $\mathbf{y}$, an adaptive arithmetic code is well suited to varying block energy, such as in a wavelet image [5] or transform audio coding application [8].

B. Weight Vector Encoding

Consider next the encoding of the weight vector $\mathbf{w} = (w_1, \ldots, w_K)$, where $0 \leq w_k \leq n$ and

$$\sum_{k=1}^{K} w_k 2^{k-1} = m = \|\mathbf{y}\|_1 \tag{4}$$

Following the development in [12], given value m, the number of possible weight vectors can be enumerated and the enumeration can be used to develop an encoding algorithm. Typically not all weight vectors are equally probable, so better coding efficiency is achieved by using an arithmetic code with adaptive context selected for each bit-plane level. For bit-plane k, using knowledge of m and the the weights of higher bit-planes $w_K, w_{K-1}, \ldots, w_{k+1}$, without requirement of any additional encoding bits it can be seen easily that w_k must lie between the upper and lower bounds $0 \leq L(k) \leq w_k \leq U(k)$, where

$$U(k) = \min\{n, \lfloor \frac{m - \sum_{i=k+1}^{K} w_i 2^{i-1}}{2^{k-1}} \rfloor\}, \tag{5}$$

$$L(k) = \max\{0, \lfloor \frac{m - \sum_{i=k+1}^{K} w_i 2^{i-1} - n \sum_{i=1}^{k-1} 2^{i-1}}{2^{k-1}} \rfloor\}. \tag{6}$$

Hence, there are three possible cases: 1) if $U(k) = 0$, clearly $w_k = 0$; 2) if $U(k) = L(k)$, then w_k is explicitly known; and 3) $L(k) \leq w_k \leq U(k)$ with $L(k) < U(k)$. Only for the last case does w_k need to be encoded. Bit plane weight w_1 never needs to be encoded, since, from (4), it can be computed from m and $w_2, \cdots, w_K$. Our implementation uses a binary arithmetic code to encode a binary representation of w_k, with context dependent on m, $U(k)$, $L(k)$, and k.

C. Enumeration Bit-plane Coding

Based on the encoding algorithm 1) – 3) presented above, consider the enumeration encoding of magnitude bit-plane $\mathbf{b}_k$, conditioned on the bit-plane weight, w_k. There are exactly $\binom{n}{w_k}$ possible bit-plane vectors, and enumeration encoding simply uses a fixed-length code with at most $\lceil \log_2 \binom{n}{w_k} \rceil$ bits to encode $\mathbf{b}_k$. Clearly, if $w_k = 0$ or $w_k = n$, no bits are required to encode $\mathbf{b}_k$. So, assume that $1 \leq w_k < n$ and consider a binary arithmetic code to be used to encode $b_{k,i}$, for $i = 1, \ldots, n$. Without any conditions other than w_k, the probability model used to encode the first bit is $p(b_{k,1} = 1) = w_k/n$. As each successive bit is encoded, the probability model is updated to correspond to the encoded bits. Define $\mathbf{b}_k^i = (b_{k,1}, \ldots, b_{k,i})$ for $i = 1, \ldots, n$, with $\mathbf{b}_k^0$ empty. The probability model used to encode the i^{th} bit in $\mathbf{b}_k$ is

$$p(b_{k,i}|w_k, \mathbf{b}_k^{i-1}) = \begin{cases} \frac{w_k - wt(\mathbf{b}_k^{i-1})}{n+1-i}, & \text{if } b_{k,i} = 1; \\ \frac{n+1-i-[w_k - wt(\mathbf{b}_k^{i-1})]}{n+1-i}, & \text{if } b_{k,i} = 0; \end{cases} \tag{7}$$

with resulting probability of the bit-plane given by

$$p(\mathbf{b}_k|w_k) = \prod_{i=1}^{n} p(b_{k,i}|w_k, \mathbf{b}_k^{i-1}) = \frac{1}{\binom{n}{w_k}}. \tag{8}$$

As the bits are successively encoded, eventually the encoding reaches a termination condition where either i) all remaining bits in $\mathbf{b}_k$ are zero ($p(b_{k,i} = 1|\mathbf{b}_k^{i-1}) = 0$) or all remaining bits in $\mathbf{b}_k$ are one ($p(b_{k,i} = 1|\mathbf{b}_k^{i-1} = 1)$). This must occur before the final bit, $b_{k,n}$, is encoded, since $b_{k,n} = w_k - wt(\mathbf{b}_k^{n-1})$.

The probability model (7)-(8) implies that every binary n-tuple of weight $w_k = w$ has the same probability. Since an arithmetic code encodes a string of probability p using $-\log_2 p$ bits, then every bit-plane of the same weight is adaptive arithmetically encoded using a codeword of the same length, namely, $\log_2 \binom{n}{w_k}$ bits.

IV. Simulation results

The encoding method described in Section III is implemented using adaptive binary arithmetic codes (using software adapted from [16]) to encode the cubic lattice codevector ℓ_1 norm, the bit-plane weights, and the bit-planes. The sign bits are encoded as "raw," that is, uncoded, bits. The bit-stream produced is embedded, and so supports progressive decoding (although that feature is not used in the results to follow). In the remainder of the paper this encoding method is referred to $L1$-based coding or one-norm coding.

To evaluate the performance of the one-norm encoding algorithm, several sources were encoded, and the rate vs. signal-to-noise ratio (SNR) performance determined. For simplicity, as a reference, we select the entropy vs. SNR performance of a uniform scalar quantizer (USQ), where the entropy is computed as the zeroeth-order entropy of the quantization levels. We refer to this as the USQ performance. The USQ step size is matched to the step size of the cubic lattice VQ used in the one-norm coder, so in the simulation results the SNRs of the two encoding methods are identical. The USQ performance uses empirically computed entropy as the encoding rate, whereas the one-norm encoding results are based on actual encoded file sizes.

Fig. 1 compares USQ to the $L1$-based encoder for vector dimensions 4, 16, 64, and 256, and a memoryless Laplacian source. At low encoding rates the curves coalesce, but at high encoding rate there is a small gap between the ideal USQ rate and the one-norm encoder rate, with the gap larger for smaller dimensions. This gap is due to slight coding inefficiencies in the $L1$-based encoding of the vector norm, and bit-plane weights.

Fig. 2 compares USQ entropy vs. SNR to one-norm encoder performance for a 4-class Gaussian mixture model. The vector dimension is 8 and the 4 classes are selected to model the vector energy distribution of audio transform coefficients, as constructed in [17]. Each class consists of independent and identically distributed Gaussian random variables with zero mean and, respectively, mixture class probabilities 0.4894, 0.2927, 0.0462, and 0.1717, and variances 1.1203, 0.2327, 109.38, and 7.7510. This 4-class model was found to well-model the block energy dependence observed in the 8-dimensional vectors of transform coefficients encoded (using the RE_8 lattice VQ [9]) in the AMR-WB+ audio coding standard [8]. Note that the one-norm coder provides a consistent gain of about 1 dB over the USQ entropy vs. SNR performance. Note also that the performance is lower-bounded by the Shannon lower bound to the rate-distortion function, and further that the gap between the $L1$-coder performance and the lower bound is 1.53 dB at large encoding rate, which reflects the granular gain limitation of cubic lattice quantization.

Fig. 3 compares the one-norm encoder performance for vector dimension 8 to the USQ entropy vs. SNR curve, for transform audio data derived from about one minute of music extracted from four difference audio segments (with sampling rate 48 kHz). The SNR is computed in the spectrally weighted transform domain (as in [17], and corresponds to the use of the RE_8 lattice VQ in the AMR-WB+ coder). The results consider the three different discrete Fourier transform block lengths used in [8] (namely, N =288, 576, and 1152 samples). The L1-coder results show a consistent improvement.

Fig. 4 is similar to Figure 3, but considers transform domain encoding of speech. In AMR-WB+ applications, the transform excitation mode is observed to be selected to encode roughly 40% of the speech segments [18]. As for the music source, the L1-coder provides consistent SNR improvment.

V. Conclusion

The one-norm-based encoding method described in the paper can handle arbitrary vector dimension and uses adaptive binary arithmetic codes for coding efficiency. The method is especially suitable for encoding transform, subband, or wavelet data that has little correlation, but displays significant block energy dependence, as is common in image or audio sources. Using the approach in [11], the proposed algorithm can also possibly be applied to other binary lattices.

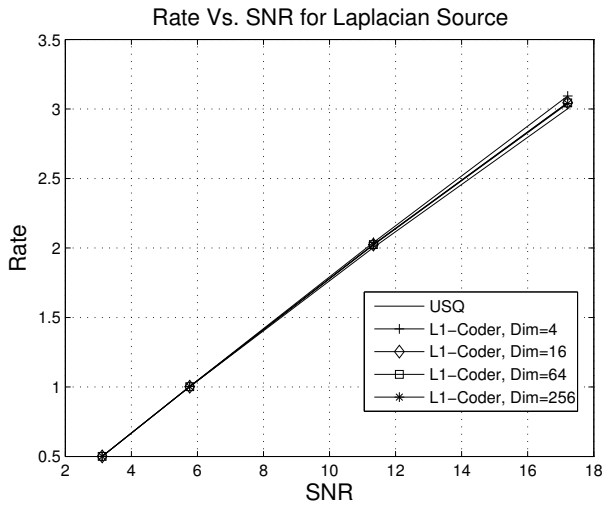

Fig. 1. Encoding of memoryless Laplacian source.

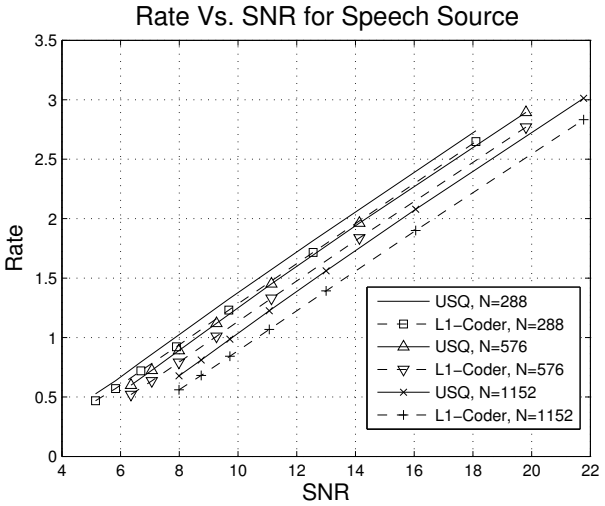

Fig. 4. Encoding of transform speech data.

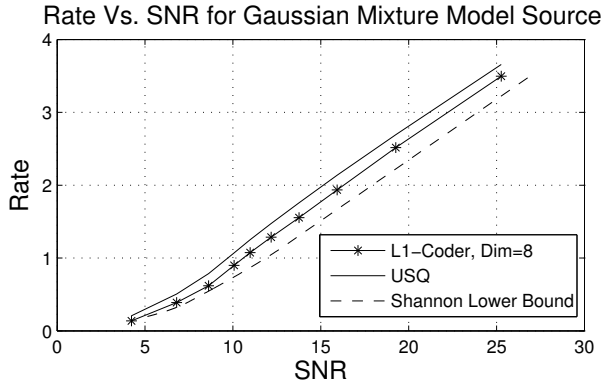

Fig. 2. Four-class Gaussian mixture model source.

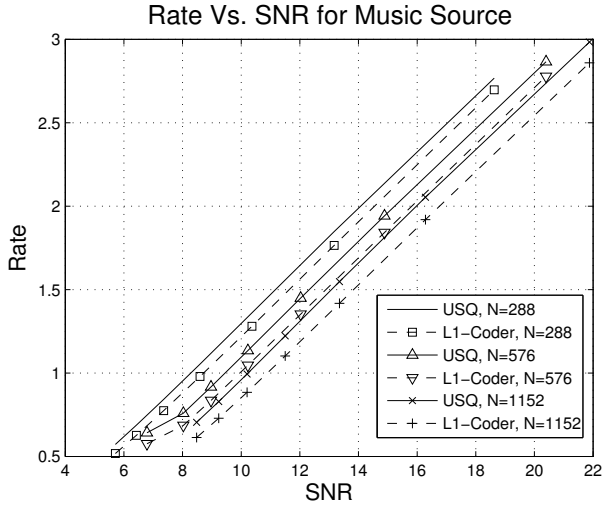

Fig. 3. Encoding of transform audio data.

REFERENCES

[1] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Springer, 1992.

[2] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trana. Inform. Theory,* vol. 25, pp. 373-380, 1979.

[3] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices, and groups*, Springer Verlag, 1999.

[4] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes,and quantization," *IEEE Trans. on Information Theory*, vol.28, no.2, pp.211-226, March 1982.

[5] M. Barlaud, P. Sole, T. Gaidon, M. Antonini, and P. Mathieu, "Pyramidal lattice vector quantization for multiscale image coding," *IEEE Trans. Image Processing*, vol. 3, no. 4, pp. 367-381, 1994.

[6] Rajiv Laroia and Nariman Farvardin, "A structured fixed-rate vector quantizer derived from a variable-length scalar quantizer – Part I: Memoryless sources," *IEEE Trans. Inform. Theory*, pp. 851-867, May 1993.

[7] D. G. Jeong and J. D. Gibson, "Image coding with uniform and piecewise-uniform vector quantizers," *IEEE Transactions on Image Processing,* vol. 4, no. 2, pp. 140-146, Feb. 1995.

[8] *Extend AMR Wideband Codec; Transcoding functions*, 3GPP TS 26.290, 2005.

[9] M. Xie. and J. P. Adoul (1996), "Embedded algebraic vector quantizers (EAVQ) with application to wideband speech coding," *Conf. Proceeding*, ICASSP, pp. 240-243, 1996.

[10] G. D. Forney, "Coset codes-II: Binary lattices and related codes," *IEEE Trans. on Inform. Theory*, vol.34, no.5, pp. 1152-1187, 1988.

[11] W. Patchoo and T. R. Fischer, "Bit-plane coding of lattice codevectors," to be appeared in *IEICE Transactions*, 2013.

[12] T. R. Fischer, "Pyramid vector quantizer," *IEEE Trans. Inform. Theory,* vol. 32, pp. 568-583, 1986.

[13] Z. Mohdyusof and T. R. Fischer, "An entropy-coded lattice vector quantizer for transform and subband image coding," *IEEE Trans. Image Processing*, vol. 5, pp. 289-298, Feb. 1996.

[14] T. M. Cover and J. A. Thomas, *Element of Information Theory*, 2nd ed., John Wiley & Son Inc., New Jersey, 2006.

[15] K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann, 2nd Edition, 2000.

[16] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun ACM*, vol. 30, pp. 520-540, June 1987.

[17] W. Patchoo and T. R. Fischer, "Gaussian mixture modeling of lattice-based spherical vector quantization performance in transform audio coding," *Conference Proceedings*, International Conference on Acoustics, Speech, and Signal Processing, pp. 373-376, March 2010.

[18] T. R. Fischer, H. Sung, J. Zhan, and E. Oh, "High-quality audio transform coded excitation using trellis codes," *Conference Proceedings*, International Conference on Acoustics, Speech, and Signal Processing, pp. 197-200, April 2008.