Hybrid Low Bitrate Audio Coding Using Adaptive Gain Shape Vector Quantization

Sanjeev Mehrotra, Wei-ge Chen, Kazuhito Koishida, Naveen Thumpudi

Microsoft Corporation
One Microsoft Way, Redmond, WA 98052
{sanjeevm,wchen,kazukoi,naveent}@microsoft.com

***Abstract*—Audio coding at low bitrates typically suffers from artifacts caused by bandwidth truncation. In this paper we present a novel scheme to code audio signals at low bitrates which uses a traditional scalar quantization followed by entropy coding to code some portions of the spectrum (typically the lower portion). The other portions (typically the higher portions) of the spectrum are coded at a low bitrate using an adaptive gain shape vector quantizer where the codebook for vector quantization is formed by unmodified or modified versions of the portions of the spectrum which have already been coded. Fixed pre-trained codebooks are also available for use in certain cases. The use of such a scheme results in an audio codec which has been shown to be among the best audio codecs available at low bitrates. In addition, the decoder complexity of this audio codec is significantly lower than any other codec of equal quality at low bitrates.**

I. Introduction

With increasing interest in direct delivery of music to devices such as mobile phones and satellite radios over bandwidth constrained networks such as wireless, there is an increasing demand for better music quality at lower bitrates. It is well known that most existing audio codecs such as MPEG II - Layer 3 (MP3) and Advanced Audio Coding (AAC) perform poorly at bitrates below 128kbps for 16-bit/sample, 44.1kHz, stereo music. There are simply not enough bits to code the entire spectrum. Therefore the spectrum is typically truncated resulting in a muffled sounding signal.

To fix the problem of bandwidth truncation, existing codecs have been augmented with decoder side techniques which with the aid of some side information, try to extrapolate the missing high frequencies. One example of such a technique is the High-Efficiency Advanced Audio Coding (HE-AAC) codec where regular AAC is used to code a signal which has a lower sampling rate (and thus lower bandwidth) than the original signal. Then with the aid of small side information, the decoder uses the spectral band replication technique (SBR) [1], [2] to try to represent the missing high frequencies. Although the quality of these alternatives is far superior to prior schemes, it still suffers from synthetic sounds for some clips due to the fact that the reconstructed high frequencies most likely will not look similar to the original signal. It merely creates the sensation of higher frequencies. Also, the decoder complexity is very high which can be a big drain on battery life of devices such as mobile phones. This is because the processing happens in a different transform domain than the base coding requiring the decoder to do an additional forward and inverse transform. Techniques such as SBR do a full reconstruction to time domain, followed by an analysis filterbank, then do the processing in the subband domain, and then use a synthesis filterbank to do the reconstruction.

Therefore, it is useful to find alternatives to existing audio coding schemes which can more faithfully represent the audio signal as well as provide a lower decoder complexity. To that end we propose a hybrid scheme which utilizes an existing codec to code some portions of the spectrum as the spectral band replication technique does. However, instead of simply using some small side information parameters to try and guess the remaining portion of the spectrum at the decoder end, we propose to use a low bitrate coding of this portion of the spectrum using adaptive vector quantization.

Vector quantization (VQ) has also been proposed for use in audio coding such as the TwinVQ work [3], [4], [5] that is also part of the MPEG standard. However, existing vector quantization approaches have difficulty in that the codebooks are pre-trained and thus the coding quality is sometimes not as good. In addition, the dimension of the vectors that can be used is limited due to encoder complexity and memory constraints.

In this paper, we propose a scheme which uses gain-shape vector quantization (GSVQ) [6] to code a remaining portion of the audio spectrum, where the codebook that is used is adaptive and is dynamically formed from the portion of the spectrum which has already been coded. The remainder of the paper goes over our proposed scheme.

II. Gain Shape Vector Quantization

A vector quantizer with an N element codebook $C = \{\boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{N-1}\}$ using the Euclidean distance as the distortion measure maps vectors $\boldsymbol{x}$ to an index i using an encoder $i = \arg\min_n \|\boldsymbol{c}_n - \boldsymbol{x}\|$ and then reconstructs using a decoder $\hat{\boldsymbol{x}} = \boldsymbol{c}_i$. If the rate of the vector quantizer is low, then typically the power of the codevectors will not match that in the original vector. Sometimes, as in the case when using VQ for audio coding to code the spectral coefficients, the energy of the codevectors needs to be close to that of the original. One could factor this into the distortion metric when designing the vector quantizer. However, this only works if the vector $\boldsymbol{x}$ is always going to be of the same magnitude which is usually not the case.

An alternative is to use GSVQ. GSVQ is a modified version of vector quantization (VQ) where the gain (or energy) is separated from the shape so that it can be more accurately represented. For example, in $\boldsymbol{x} = \|\boldsymbol{x}\|\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$, the first term $s = \|\boldsymbol{x}\|$ is the gain and the second term $\boldsymbol{y} = \boldsymbol{x}/\|\boldsymbol{x}\|$ is a unit norm vector. Now, the gain s can be quantized separately using some number of bits so that the gain is accurately represented. The shape $\boldsymbol{y}$ is a unit norm vector which can be coded using a codebook with unit norm vectors.

III. Audio Coding Using Gain Shape Vector Quantization

Vector quantization is often used in audio coding but the performance is usually on par or worse than codecs such as AAC. At low rates, the performance of codecs using VQ is often slightly worse than codecs such as HE-AAC. Coding the entire spectrum using VQ requires the use of pre-trained codebooks and thus the quality is not as good as it can be if the codebooks were somehow adaptive. In addition, by coding the entire spectrum using VQ can result in very high encoder complexity [3], [4].

Here we propose a hybrid coding to code the spectrum which utilizes a standard non-VQ based method to code some portions of the spectrum and a VQ based low bitrate scheme to code the remaining portions. By using a hybrid scheme, we take advantage of the low bitrate coding capabilities of VQ and avoid the disadvantages since we only use it to code some of the frequencies and use adaptive codebooks instead of fixed ones.

The VQ based coding takes only about 10% of the total bitrate. Since typically some portion of the spectrum (such as the higher portion) can be coded at a lower bitrate, we avoid the need for a large codebook while still being able to use large dimension vectors. By dynamically forming the codebook, we also avoid the need for decoder memory to actually store the codebooks. The codebook is not trained using algorithms such as the Generalized Lloyd Algorithm (GLA) [6], but rather simply formed from portions of the spectrum already coded as will be explained in later sections. This is similar to fractal coding used in image/video coding and exploits the self-similarity present in the waveform.

The portion of the spectrum that is not being coded by the traditional codec is split into vectors (or bands) of various sizes as shown in figure 1, the sizes being determined by various factors in the rate control and the frequency position of the vector. In figure 1a, the typical case where the lower portion of the spectrum is coded using a traditional codec is shown. The vectors formed to code the higher portion are shown in the figure. In figures 1b-c, some coefficients in the higher portion of the spectrum are also coded using the traditional codec. In these cases, the vector can be formed by taking just the zero valued (uncoded) coefficients over this region, as in 1b, or configuring the vectors so that this region does not overlap with any of the vectors, as in 1c.

The gain of each of the vectors is quantized and then coded using prediction followed by entropy (Huffman) coding. The

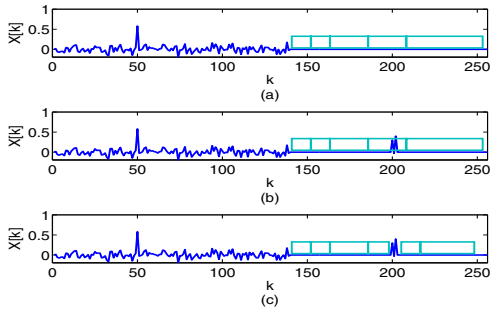

Fig. 1. Vector configuration for portion of spectrum coded using VQ.

shape (normalized) is quantized using a vector quantizer. The codebook for this vector consists of modified (using linear or non-linear transforms) or unmodified normalized vectors taken from the portion of the spectrum which has already been coded. This adaptive codebook design is meant to take advantage of the similarity between various components of the spectrum which is often present in audio signals due to harmonic components. The codebook formation is very simple and requires almost no computation at the decoder end. There also exist codebooks formed from a fixed pre-trained spectrum for vectors (typically noise-like vectors) which do not have a good match in the adaptive codebooks. Decoder side noise substitution can also be used for the shape if no good match is found or if the bitrate is very low. We also allow the use of two interleaved vector quantizers to be used for some vectors which have differing tonal and noise characteristics which cannot be accurately coded using a single vector.

The advantages of this algorithm over techniques such as spectral band replication is that since we actually code the spectrum, our reconstruction is a better representation and thus does not have as much of a synthetic sound. The rate used is slightly higher than the side information sent for SBR but is still very low when compared to the total bitrate. Since our coding of the higher spectral coefficients is in the same transform domain as the base coding, and since vector quantization decoder complexity is very low, our overall complexity is significantly lower than that of SBR.

In the remainder of the paper, we will assume that the portion of the spectrum being coded using a traditional audio codec is the lower portion and the portion which is coded using the adaptive GSVQ is the higher portion. However, this does not necessarily need to be the case and arbitrary partitioning between these two portions is possible as in figures 1b-c.

As shown in figure 2, the audio signal is first transformed into the frequency domain using an overlapping transform such as the modulated discrete cosine transform (MDCT). The signal is divided into barks (or critical bands), each band is weighted according to a weight derived from a psychoacoustic model, followed by a channel transform, then quantized and entropy coded. The channel transform is an orthogonal transform applied across the channels of the audio source and is used to remove correlation across the channels. A simple example of this for a stereo source is to code the sum and

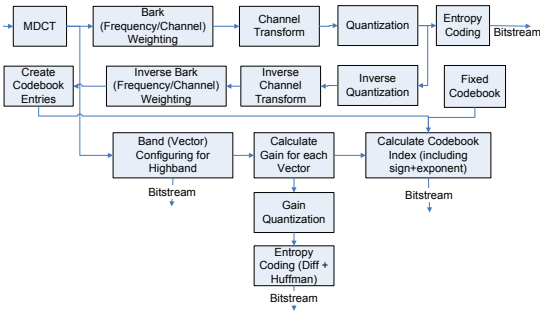

Fig. 2. Encoder Diagram.

difference of the right and left channels. This algorithm is used to code all frequency coefficients up to a certain point. The exact location for this is based upon various factors in the rate control, such as how many bits are available, and upon the signal complexity. This determination is dynamic and if the base coding is able to accurately code the signal up to a high frequency, it will do so. The remainder of the spectrum after this point is coded using adaptive GSVQ. The GSVQ is applied to the spectral coefficients prior to the channel transform which results in better quality than if done afterwards.

In the following subsections, we explain how to form the vectors and the adaptive codebook used to code them. We also go over what parameters we send for the gain and the shape and go over the case when a single block of coefficients needs to be split into two vectors.

A. Vector Configuration

Let $X[k]$ be the kth spectral coefficient after the time-to-frequency transform such as the MDCT as shown in figure 2. Let K_s be the starting frequency of the portion coded using the GSVQ. Let K_e-1 be the ending frequency. The frequency range of $K_e - K_s$ spectral coefficients is divided into M vectors (or bands). Let $m = 0, 1, \ldots, M-1$ be the index of the vector. Let $k_s[m]$ be the starting position of the mth vector, and let $k_e[m]-1$ be the ending position of the mth vector. Let $L[m] = k_e[m] - k_s[m]$ be the length of the mth vector. We assume that $k_e[m] = k_s[m+1]$ so that there are no holes between the vectors. The mth vector is then given by

$$\boldsymbol{x}_m = \begin{bmatrix} X[k_s[m]] \\ X[k_s[m]+1] \\ \vdots \\ X[k_s[m]+L[m]-1]. \end{bmatrix} \quad (1)$$

The value for K_s is typically the first coefficient after the ending position of the base coding. The base codes up to a point at which the quality of the coded coefficients is considered to be "good". The definition of good can be a noise-to-mask ratio (NMR) and can include other factors such as spectral holes (large regions of non-zero spectral coefficients being quantized to zero). Assume b_e-1 is the last coded spectral coefficient in the base. Then, typically $K_s = k_s[0] = b_e$ so that there is no spectral hole between the base and the portion coded using gain shape vector quantization, but does not have to be. The ending frequency, $K_e = k_e[M-1]$, is determined by the desired bandwidth of the reconstructed signal. For 64kbps, we can almost go up to full bandwidth (18-20kHz), so that the entire spectrum is coded. Thus K_e is typically the block size used in the time-to-frequency transform.

The value for M is determined by the current buffer fullness, bitrate, sampling rate, and transform block size. It is chosen by the rate control so that the total number of bits used by the GSVQ coding is within some desired range. The sizes of the M vectors is chosen so that coefficients with similar gain (or energy) or shape are in the same vector. A simple scheme is also available which takes advantage of the fact that the energy envelope of audio signals generally gets flatter with increasing frequency and the fact that higher frequency coefficients need less accurate representations of the actual values. Therefore, we can simply use increasing vector sizes to tile the frequency region to be coded. For example, one could approximately make the sizes $L, L, 2L, 2L, 4L, 4L, 4L, 4L, 8L, 8L, ...$, where L is chosen so that the entire frequency range is tiled. That is, we keep doubling the vector size after some number of vectors. A typical vector configuration can be as shown in figure 1a with smaller vector sizes for the lower frequencies.

B. Gain Coding

The gain of the mth vector is a scaled version of the norm of the vector (the root mean square (RMS) value) and can be computed as

$$s_m = \sqrt{\frac{1}{L[m]} \sum_{k=k_s[m]}^{k_s[m]+L[m]-1} X^2[k]}. \quad (2)$$

The norm is not used since otherwise different sized vectors with similar distribution would have different values. This is because the norm is proportional to the length of the vector and thus it would be harder to predict the gain across different sized vectors. The RMS value on the other hand can be predictable across different size vectors since it represents the average value for a single coefficient.

The gain is quantized using a non-linear quantizer by quantizing the gain in the log domain using a step size of Δ. The gain of the vector is differentially coded by subtracting the previous vector's quantized gain in the log domain, as $i_m = \text{round}((\log_{10}(s_m) - \hat{r}_{m-1})/\Delta)$, $\hat{r}_m = i_m\Delta + \hat{r}_{m-1}$, with $\hat{r}_{-1} = 0$ so that the first vector $m = 0$ is not predicted. The index i_m is sent in the bitstream for all vectors m. The decoder reconstructs the scale factor as $\hat{s}_m = 10^{\hat{r}_m}$.

C. Codebook Formation

To code the higher portion of the spectrum, the encoder first reconstructs the lower portion to create an adaptive codebook. This is done by first obtaining the quantized coefficients as shown in figure 2. Each normalized vector $\boldsymbol{y}_m = \frac{\boldsymbol{x}_m}{s_m}$ is coded using vector quantization with either the adaptive codebooks formed from the base spectrum or the codebooks formed from

a pre-trained random source spectrum. Since each vector is of a different length and norm $L[m]$ because of the normalization we have used, the codebook being used for each of the M vectors to be coded has to be different. Let there be P_a adaptive codebooks to choose from and let P_f be the number of fixed codebooks to choose from for each vector. Rather than letting these codebooks be arbitrary, the P_a adaptive codebooks are formed from a base codebook formed from the spectral coefficients which are already coded. The P_f fixed codebooks are formed from a single fixed pre-trained random source and is used for noise-like portions of the signal which are not represented well from the base spectral coefficients.

The codebook formation is very simple and consists of taking normalized portions of the spectrum that has already been coded. The base codebook used for creating the P_a adaptive codebooks consists of N normalized vectors taken from the base spectral coefficients. Let $\boldsymbol{c}_n$ be the nth codevector in the base codebook used to code the mth vector, given by

$$\boldsymbol{c}_n = \frac{\boldsymbol{d}_n}{s_{c,n}}, \tag{3}$$

$$\boldsymbol{d}_n = \begin{bmatrix} \tilde{X}[l_s[n]] \\ \tilde{X}[l_s[n]+1] \\ \vdots \\ \tilde{X}[l_s[n]+L[m]-1] \end{bmatrix}, \tag{4}$$

$$s_{c,n} = \sqrt{\frac{1}{L[m]} \sum_{k=l_s[n]}^{l_s[n]+L[m]-1} \tilde{X}^2[k]}. \tag{5}$$

Now the norm of all the codevectors is also $L[m]$ as is for the vector being coded. $\tilde{X}[k]$ are the reconstructed values of $X[k]$ since they have been coded using the base coding and are different from the original values.

In determining the N codebook entries, we want to make sure that there is maximum separation between the entries as well as making sure that the position of the last coefficient in the last entry in the codebook does not exceed the position of the last coded spectral coefficient, $b_e - 1$. Therefore, we want $l_s[N-1] + L[m] - 1 \leq b_e - 1$. If $l_s[N-1] = b_e - L[m]$ is the starting position of the last codevector, then we will meet this constraint. To achieve approximately uniform spacing, we can use

$$l_s[n] = \left\lfloor \frac{n(b_e - L[m])}{N-1} \right\rfloor. \tag{6}$$

An example of codevectors from codebooks is shown in figure 3. In the figure, we show a transform with block size 256 with ending position $b_e = 150$ being the index of the last coded spectral coefficient. The four codevectors from this unmodified codebook are shown for two different vector sizes, $L[m]$. The codevectors consist of the spectral coefficients taken over the rectangular regions shown in the figure. Although the example shows only four codevectors in the codebook, typically there will be more. In the case of $L[m] = 25$, the four codevectors do not overlap since the width is small. In the case of $L[m] = 70$, the four codevectors are overlapping.

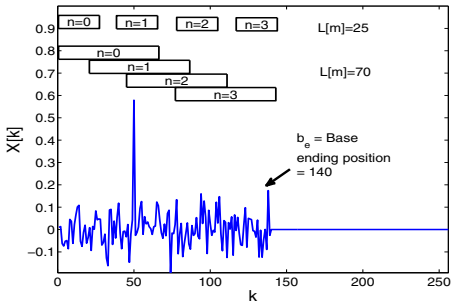

Fig. 3. Example spectrum and codevectors.

Since the computation of the scale factor, $s_{c,n}$, needed to normalize the unnormalized codevectors, $\boldsymbol{d}_n$, will require $\sum_{m=0}^{M-1} NL[m]$ computations, which can be very large, we propose a simple way to compute the scale factor. This involves computing a single cumulative energy sum of the base portion of the spectrum using high precision arithmetic. Let $e[k] = \sum_{j=0}^{k-1} \tilde{X}^2[j]$, be the cumulative energy, computed for $k = 0, 1, \ldots, b_e$. Then the scale factor for each of the codevectors can be computed as $s_{c,n} = \sqrt{(e[l_s[n] + L[m]] - e[l_s[n]])/L[m]}$.

The other adaptive codebooks are formed by modifying the codevectors of the base codebook and are there so that a better match to the vector being coded may be found. One such modification is to take care of the phase of the MDCT coefficients. The sign of the harmonic components in the highband may be flipped from the base due to the phase of the tonal component. Therefore, in the first modified adaptive codebook, the codevectors consist of N codevectors, $\boldsymbol{c}_{1,n}$, where $\boldsymbol{c}_{1,n} = -\boldsymbol{c}_n$.

Another modification is to take care of the fact that tones are often embedded in a noise floor. The noise floor is usually slowly varying over the spectrum when compared to the rate of decreasing amplitude of harmonics with increasing frequency. To compensate for this we introduce a non-linear transform to modify the codevectors, via exponentiation. This can be done by replacing $\tilde{X}[k]$ with $V[k]$ in equations (3)-(5), where $V[k] = \text{sgn}(\tilde{X}[k])|\tilde{X}[k]|^\alpha$, for some exponent α. The benefits of exponentiation can be seen in figure 4 which shows that using an exponent of $\alpha = 0.5$ lowers the tonal component without significantly affecting the noise floor. We create two additional adaptive codebooks in this manner by using $\alpha = 0.5$ and $\alpha = 2.0$. Another two are created by combining the sign with the exponentiation which gives a total of six adaptive codebooks.

Usually tonal components are harmonic and have some good match that can be found in the base. However, noise-like components in the highband may not have a match in the base. For example, a noise-like component may be present in the entire frequency range, whereas tonal components may only be present in the lower frequencies. The use of a tonal component to represent noise-like components results in them sounding unnatural. Thus, if vector $\boldsymbol{x}_m$ is deemed to be noise-like, then in addition to the adaptive codebooks formed from the base

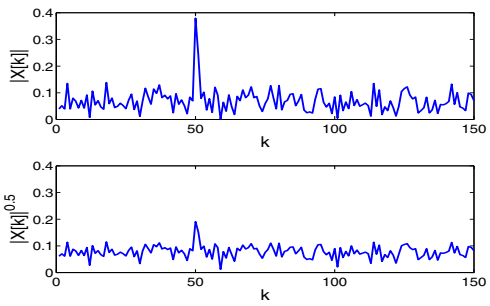

Fig. 4. Effect of Exponentiation.

spectral coefficients, we also search fixed noise codebooks which again are derived from a single base noise codebook.

As with the adaptive codebook formed from the base spectral coefficients, the noise codebook for each vector to be coded needs to be different since the length of the vector as well as its norm are varying. Therefore, instead of creating a true codebook with codevector entries, we create a pre-trained noisy spectrum of a certain length, $Z[k]$, $k = 0, 1, \ldots, n_e - 1$, where n_e is the length of the sequence and can be considered as the ending frequency of the noise spectrum. Instead of using the base spectral coefficients $\tilde{X}$ for creating the codebooks, they are instead created by using the pre-trained noisy spectrum Z. Similar to the base spectrum base codebook, we create a base noise codebook using equations (3), (4), (5), and (6), with $\tilde{X}[k]$ replaced by $Z[k]$ and b_e replaced by n_e. The base noise codebook can also be used to derive modified noise codebooks via negation or exponentiation, although the effects of exponentiation are simply used to increase diversity in the codebook instead of improving tone-to-noise ratio.

D. Finding the Best Shape Vector

For each vector coded using GSVQ, the index of the codebook being used (sign, exponent, base spectrum vs. pre-trained noise) is sent in the bitstream along with the index of the codevector within the codebook. The index entry within the codebook is between 0 and $N-1$ and is coded using a fixed length code using $\lceil \log_2(N) \rceil$ bits since there is little to be gained from prediction or entropy coding.

If both the base codebook and the pre-trained fixed noise codebook have N entries, then the number of total codevectors to choose from becomes $N(P_a + P_f)$ which can be very large. It may seem that to search all of the codebooks would be very costly at the encoder. To reduce computation complexity of the encoder we introduce an encoding algorithm which first searches to find the best index within original unmodified set taking only the sign into account. In addition, only the adaptive codebook from the base spectral coefficients is searched unless the vector is found to be noise-like in which case the codebooks formed from the noise sequence $Z[k]$ are also searched.

If the Euclidean distance is used as the distortion metric when finding the best codeword, then since both the codevectors $\boldsymbol{c}_n$ and the original vector $\boldsymbol{y}_m$ are of norm $L[m]$, the distortion calculation simply becomes $\| \pm \boldsymbol{c}_n - \boldsymbol{y}_m \|^2 = 2\left(L[m] \mp \sum_{j=0}^{L[m]-1} c_n[j] y_m[j]\right)$. So, with a single calculation of the dot product between the codevector and the original vector going over all the N entries, we can figure out the best index in the set of codevectors and the corresponding best sign of the codevector.

Once the best index is calculated, we then proceed to obtain the best exponent amongst the set $\alpha = 0.5, 1.0, 2.0$ for the given best index. The best exponent can also be calculated by trying all three exponents and simply picking the one which gives the minimal distortion in Euclidean distance. Although we don't search the entire codevector space due to complexity considerations, we still are able to find a good match.

E. Reconstruction

The decoder takes the coded scale factor and recreates the coefficients by taking coefficients from the optimal codebook. In the case when the codevector is taken from the codebook created from the base spectral coefficients, if the optimal sign is given by $v_{opt} = \pm 1$, the optimal exponent given by α_{opt}, and the optimal index into the codebook set given by n_{opt}, then the reconstruction can be given by

$$\hat{\boldsymbol{x}}_m = \frac{\hat{s}_m}{s_{w,m}} \begin{bmatrix} W[l_s[n_{opt}]] \\ W[l_s[n_{opt}] + 1] \\ \vdots \\ W[l_s[n_{opt}] + L[m] - 1], \end{bmatrix} \quad (7)$$

where $W[k] = v_{opt} \text{sgn}(\tilde{X}[k]) |\tilde{X}[k]|^{\alpha}$, and $s_{w,m}$ is the RMS value for the vector with components $W[k]$ and can be computed similar to equation (5). If the pre-trained fixed noise codebook is used instead, then $W[k] = v_{opt} \text{sgn}(Z[k]) |Z[k]|^{\alpha}$.

F. Vector Partitioning

Sometimes when tones are embedded in noise, the exponent modeling is still not sufficient to find a good match for the shape. In these cases, we split the original vector $\boldsymbol{x}_m$ into two components, one for the tones, another for the noise floor. Instead of sending a bit mask to specify what is noise and what is tone, which will take more bits than are available, we instead use an implicit classification into tone and noise for the vector $\boldsymbol{x}_m$ based upon a threshold, T, sent in the bitstream and the codevector being used to represent the vector. The classification uses the following to classify,

$$\tau = \{k : |c_{n_{opt}}[k]| \geq T\} \quad (8)$$

$$\eta = \{k : |c_{n_{opt}}[k]| < T\}, \quad (9)$$

where τ is the set of tonal components, and η is the set of noise components in the vector. The gain for the entire vector is still coded and sent as in equation (2), with the an additional scale factor representing the noise-to-total ratio as

$$\gamma = \frac{1}{s_m} \sqrt{\frac{1}{|\eta|} \sum_{k \in \eta} X^2[k]}, \quad (10)$$

where $|\eta|$ is the cardinality of the set of noise components. The codebook shape index is coded separately for each of the two sets. The shape for the coefficients in set τ is coded

TABLE I
LISTENING TEST RESULTS. VALUES SHOWN ARE PERCENTAGE OF LISTENERS WITH THE GIVEN PREFERENCE FOR EACH SONG.

Song	Ours much more	Ours more	Ours slightly	Identical	HE-AAC slightly	HE-AAC more	HE-AAC much more
Blank baby	4.00	11.00	19.67	38.33	12.67	11.33	3.00
Sun Is Shining (Island Mix)	3.67	17.33	25.67	25.00	14.33	10.00	4.00
Walk of Life	5.00	12.67	16.33	28.00	15.67	16.33	6.00
Clocks	5.00	15.00	19.67	34.00	12.67	12.00	1.67
Eclipse	3.67	10.00	16.67	37.33	21.00	8.00	3.33
Boulevard Of Broken Dreams	6.00	19.67	19.67	21.33	17.00	12.33	4.00
Open Your Eyes	1.67	10.00	16.33	46.00	16.00	9.67	0.33
Take A Look Around	4.67	11.67	18.00	29.33	16.33	15.67	4.33
Castanets	20.00	25.33	17.00	26.67	5.67	4.67	0.67
Dont You (Forget about me)	4.00	11.33	21.33	38.33	15.67	8.33	1.00
Bitter Sweet Symphony	2.33	11.00	15.67	34.33	21.00	12.67	3.00
This Love	2.00	10.33	13.67	48.00	14.33	8.67	3.00
Total	5.17	13.78	18.31	33.89	15.19	10.81	2.86

using the adaptive codebooks from the base portion of the spectrum. The shape for the coefficients in set η is coded using the codebooks derived from the pre-trained noise sequence. The decision whether to use two vector approach is made at the encoder by looking at the difference between the peak and median values. If the vector is partitioned into two sets, then to save bits, the codevector exponent is assumed to be 1.0 since the tone-to-noise ratio should be correct. However, the additional scale factor and shape coding still approximately doubles the number of bits used to code such vectors.

IV. EXPERIMENTAL RESULTS

A listening test was done using the hybrid audio coding scheme presented in this paper with coding using HE-AAC at 64kbps CBR (constant bit rate with a buffer) coding of a 44.1kHz stereo source. The scale factors were quantized with a 1 dB quantization step size, and $N = 64$ was used as the number of codevectors in each of the codebooks. After prediction and entropy coding, the scale factors took about 3.5 bits to code, and the codebook index took 6 bits to code. The sign takes 1 bit, and the exponent takes approximately 1.5 bits and is coded using a simple code. In total we take about 12 bits per vector to code. We try to use approximately 10-16 vectors per transform block depending upon transform block size and bits available in the buffer. This gives an approximate rate of about 3kbps per channel. So at 64kbps, we end up using about 6kbps for GSVQ for stereo coding which is 10% of the total bitrate. The bandwidth at which the GSVQ starts coding varies depending upon content but is usually around 10-12kHz.

A blind listening test was conducted using 12 audio samples presented to three hundred listeners. The results of the listening test are shown in table I which shows the percentage of listeners for a given preference. For example, column 1 is the percentage of listeners who preferred our algorithm much more than HE-AAC. Approximately 37% of the listeners preferred our algorithm, 34% said the two were identical, 29% liked HE-AAC better. This proves that this codec is identical or better than HE-AAC proving it to be amongst the best audio codecs at this bitrate. However, the decoder complexity is much lower than HE-AAC since the decoder operates in the same transform domain as the base. We are able to decode using only 30-35 MIPS on an ARM processor, whereas typical HE-AAC decoder implementations require at least 50 MIPS on a similar processor for 44.1kHz stereo source [7].

It should be noted that the syntax described above is very rich, only a simple encoder was used in this study. For example, a number of ad hoc decisions were made such as the number of vectors, bits used to code the scale factor (the step size Δ), and the codebook size chosen.

V. CONCLUSION

We have presented a hybrid audio coding scheme which uses a gain shape vector quantizer to code some portions of the spectrum at a low bitrate. The gain shape vector quantizer consists of adaptive codebooks formed from spectral coefficients which have already been coded using standard audio coding techniques. The algorithm provides audio quality which is as good or better than any other audio codec at very low bitrates such as 64kbps. However, the decoder complexity is significantly lower than other audio codecs with similar quality at these low bitrates.

REFERENCES

[1] M. Dietz, L. Liljeryd, K. Kjorling, and O. Kunz, "Spectral band replication, a novel approach in audio coding," in *AES Convention 112*. Audio Engineering Society, Apr. 2002, paper 5553.

[2] M. Schug, A. Groschel, M. Beer, and F. Henn, "Enhancing audio coding efficiency of MPEG Layer-2 with spectral band replication (SBR) for DigitalRadio (EUREKA 147/DAB) in a backwards compatible way," in *AES Convention 114*. Audio Engineering Society, Mar. 2003, paper 5850.

[3] N. Iwakami and T. Moriya, "Transform-domain weighted interleave vector quantization (TwinVQ)," in *AES Convention 101*. Audio Engineering Society, Nov. 1996, paper 4377.

[4] N. Iwakami, T. Moriya, A. Jin, T. Mori, and K. Chikira, "Fast encoding algorithms for MPEG-4 TwinVQ audio tool," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 5. IEEE, May 2001, pp. 3253–3256.

[5] T. Moriya, N. Iwakami, K. Ikeda, and S. Miki, "Extension and complexity reduction of TwinVQ audio coder," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, May 1996, pp. 1029–1032.

[6] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers, 1992.

[7] Helix fixed-point HE-AAC (aacplus) decoder. [Online]. Available: https://datatype.helixcommunity.org/2005/aacfixptdec