Adaptive Vector Quantization Using Generalized Threshold Replenishment

James E. Fowler Stanley C. Ahalt

Department of Electrical Engineering
The Ohio State University, Columbus, Ohio 43210 USA

Abstract

In this paper, we describe a new adaptive vector quantization (AVQ) algorithm designed for the coding of nonstationary sources. This new algorithm, generalized threshold replenishment (GTR), differs from prior AVQ algorithms in that it features an explicit, online consideration of both rate and distortion. Rate-distortion cost criteria are used in both the determination of nearest-neighbor codewords and the decision to update the codebook. Results presented indicate that, for the coding of an image sequence, 1) most AVQ algorithms achieve distortion much lower than that of nonadaptive VQ for the same rate (about 1.5 bits/pixel), and 2) the GTR algorithm achieves rate-distortion performance substantially superior to that of the prior AVQ algorithms for low-rate coding, being the only algorithm to achieve a rate below 1.0 bits/pixel.

Introduction

Over the last 20 years, vector quantization (VQ) has received significant attention as a powerful technique for data compression. VQ is theoretically attractive due to results from rate-distortion theory that show that VQ is asymptotically optimal for the coding of a data source whose statistics are stationary in time. Although VQ has been successfully applied to the coding of many types of data, most sources can rarely be assumed to be stationary in practice, leading to a gap between the performance predicted by theory and that actually obtained in real implementations. Indeed, the nonstationary nature of the sources common in practical applications has prompted a search for more general VQ algorithms that are capable of adapting to changing source statistics as the coding progresses. Such algorithms use what we call adaptive vector quantization (AVQ).

In this paper, we first present a brief mathematical definition of AVQ which accurately describes the operation of an AVQ communication system while being sufficiently general to apply to all previously reported AVQ algorithms. We follow with the key contribution of this work, a new AVQ algorithm called generalized threshold replenishment (GTR). The GTR algorithm differs from prior algorithms in that, 1) it is an online algorithm that does not rely on substantial buffering or iterative processing, and 2) it employs an explicit consideration of both rate and distortion, two quantities that measure the performance of a data-compression algorithm. We conclude with a sample of the experimental results that we have obtained for the GTR

Support for James E. Fowler was provided by an AT&T Ph.D. Scholarship
To appear in Proceedings of the 1997 IEEE Data Compression Conference, March, 1997

algorithm. In these results, we compare the rate-distortion performance of GTR to that of several prior AVQ algorithms for an image sequence. These results show that the GTR algorithm achieves rate-distortion performance superior to that of other reported AVQ algorithms, particularly for low-rate coding.

Adaptive Vector Quantization

VQ is the generalization of scalar quantization to higher dimensions. Briefly, non-adaptive VQ consists of a vector quantizer, Q, that maps vectors from N-dimensional space to a fixed finite set $\mathcal{C}$ of N-dimensional vectors; i.e., $Q : \Re^N \rightarrow \mathcal{C}$. Set $\mathcal{C}$ is called the *codebook* of the vector quantizer.

Rate-distortion theory states that, for a stationary, ergodic random process, there exists a rate-distortion function, $R(D)$, such that, for a given distortion D, $R(D)$ is the lower bound on the minimum achievable average rate for any coding method. In addition, the theory shows the existence of a vector quantizer that achieves this bound as the dimension of the quantizer becomes infinitely large.

This theoretic asymptotic optimality of VQ has inspired its use in many applications. However, most sources of practical interest are, in reality, nonstationary. A number of algorithms, known collectively as *adaptive vector quantization* (AVQ), have been introduced (e.g., [4–9]) to provide more efficient coding for these sources. These algorithms compensate for the changing source statistics associated with nonstationary sources by periodically updating the VQ codebook.

We have developed a mathematical definition to describe in general terms the operation of these AVQ algorithms. We briefly summarize this mathematical definition below; a more comprehensive discussion is given elsewhere [1, 3]. Assume that we have a N-dimensional random-vector process, $\mathbf{X}_t$. We define *adaptive vector quantizer*, Q_t, as follows. Let $\mathcal{C}^*$ denote a large *universal codebook*, $\mathcal{C}^* \subseteq \Re^N$, that is fixed for all time t. We define a sequence of *local codebooks*, $\mathcal{C}_t$, such that $\mathcal{C}_t \subset \mathcal{C}^*$ at each time t. We restrict each set $\mathcal{C}_t$ to be finite. Adaptive vector quantizer Q_t is a time-variant mapping from N-dimensional Euclidean space to the local codebook for time t; i.e., $Q_t : \Re^N \rightarrow \mathcal{C}_t$. The output of the adaptive vector quantizer is another random-vector process, $\hat{\mathbf{X}}_t = Q_t(\mathbf{X}_t)$. Note that, in an AVQ communication system, the encoder must transmit to the decoder not only codeword indices from the quantizer but also information describing the contents of the local codebook; this latter quantity is commonly known as *side information*. There are numerous other details of both a theoretic and practical nature involved in the construction of an AVQ system; for a more thorough investigation, consult [1, 3].

The Generalized Threshold Replenishment Algorithm

In this section, we describe our new AVQ algorithm called generalized threshold replenishment (GTR). GTR is an online algorithm that does not require large amounts of batch computation, and it employs cost criteria involving both rate and distortion measures. The GTR algorithm weighs the distortion performance against the cost in rate in both the coding of the current source vector and the updating of the local codebook. In general terms, GTR incorporates a rate-distortion-based cost function

similar to the one developed by Lightstone and Mitra [9] into a framework similar to the AVQ algorithm described by Paul [4] while offering substantial performance improvement over these two approaches.

The GTR algorithm selects a codeword from the current local codebook as the potential coding of the current source vector by considering both the distortion between the two vectors and the rate needed to specify the codeword to the decoder. This rate is estimated from the current codeword probabilities, assuming that variable-length entropy coding of the VQ indices follows the quantizer. Once the winning codeword is chosen, a decision rule is evaluated to see if a codebook update would result in a reduction in distortion outweighing the cost in rate associated with the update. If so, the algorithm replaces a codeword in the local codebook with the current source vector.

To simplify the discussion of this section, we present two versions of the GTR algorithm. First, we describe the basic algorithm which lays a foundation for the move-to-front variant of the algorithm to follow. The move-to-front variant has been observed to have a slight performance advantage over the basic algorithm while being only marginally more computationally complex [3].

The Basic Algorithm

The basic variant of the GTR algorithm is outlined in detail in Fig. 1. This algorithm operates as follows. The first sequence of steps determines the codeword "closest" to the current source vector in a rate-distortion sense. This rate-distortion-based nearest-neighbor mapping operates as follows. First, as a variable-length entropy coder follows the quantizer, the estimated lengths of the variable-length codewords are calculated from the estimated probabilities, $p_{t-1}(i)$, of the codewords $\mathbf{c}_i \in \mathcal{C}_{t-1}$ as

$$l(\gamma_t(\mathbf{c}_i)) = -\log_2 p_{t-1}(i), \tag{1}$$

where $\gamma_t(\cdot)$ represents the entropy coder. Then, the distortion, $\delta(\mathbf{c}_i)$, between the current source vector, $\mathbf{X}_t$, and each codeword, $\mathbf{c}_i \in \mathcal{C}_{t-1}$, is calculated. These distortions and the variable-length codeword lengths are combined into a cost function using a rate-distortion parameter, λ; i.e., the cost function, $J(\mathbf{c}_i)$, for each $\mathbf{c}_i \in \mathcal{C}_{t-1}$ is

$$J(\mathbf{c}_i) = \delta(\mathbf{c}_i) + \lambda \cdot l(\gamma_t(\mathbf{c}_i)). \tag{2}$$

The winning codeword, $\mathbf{c}^*$, is chosen to be the one with the lowest cost function. We denote the index of $\mathbf{c}^*$ be i^*. One should note that the steps to this point implement the modified nearest-neighbor rule of the well known entropy-constrained-vector-quantization (ECVQ) algorithm [10].

In the next steps, the algorithm decides whether the local codebook needs to be updated by first estimating the improvement in distortion to be gained by a codebook update. If the codebook is updated, the current source vector will be coded with zero distortion. However, if the codebook is not updated, the current source vector will be coded with a distortion of $\delta(\mathbf{c}^*)$. Thus, the estimated distortion improvement due

Given: initial local codebook, $\mathcal{C}_0$
initial codeword probabilities, $p_0(i)$, for each codeword $\mathbf{c}_i \in \mathcal{C}_0$
rate-distortion parameter, λ
windowing parameter, ω
initial time, $t = 1$

Step 1: Calculate initial codeword lengths of the VQ-index entropy coder:

$$l(\gamma_t(\mathbf{c}_i)) = -\log_2 p_{t-1}(i).$$

Step 2: Find the distortions between each codeword $\mathbf{c}_i \in \mathcal{C}_{t-1}$ and $\mathbf{X}_t$:

$$\delta(\mathbf{c}_i) = d(\mathbf{c}_i, \mathbf{X}_t).$$

Step 3: Calculate the cost function for each codeword

$$J(\mathbf{c}_i) = \delta(\mathbf{c}_i) + \lambda \cdot l(\gamma_t(\mathbf{c}_i)).$$

Step 4: Find the winning codeword:

$$\mathbf{c}^* = \arg \min_{\mathbf{c} \in \mathcal{C}_{t-1}} J(\mathbf{c}).$$

Let the index of $\mathbf{c}^*$ be denoted i^*.

Step 5: Calculate the distortion improvement and rate cost of a codebook update, as well as the rate-distortion cost function:

$$\Delta d = -\delta(\mathbf{c}^*), \qquad \Delta r = l(\mathbf{X}_t),$$
$$\Delta J = \Delta d + \lambda \cdot \Delta r.$$

Step 6: Set $\mathcal{C}_t = \mathcal{C}_{t-1}$. If $\Delta J < 0$, go to Step 6a. Else, go to Step 6b.

Step 6a: Set $\mathbf{c}^* = \mathbf{X}_t$ in $\mathcal{C}_t$. Send to the decoder $\mathbf{X}_t$, entropy-coded index i^*, and a flag indicating a codebook update. Go to Step 7.

Step 6b: Send the entropy-coded index i^* and a flag indicating no codebook update.

Step 7: Estimate the new codeword probabilities:

$$p_t(i) = \begin{cases} [\omega p_{t-1}(i)]/(\omega+1), & i \neq i^*, \\ [\omega p_{t-1}(i) + 1]/(\omega+1), & i = i^*. \end{cases}$$

Step 8: Set $t = t + 1$ and go to Step 1.

Figure 1: The basic GTR algorithm

to a codebook update is $\Delta d \triangleq -\delta(\mathbf{c}^*)$. The cost in rate of the codebook update is the amount of side information needed to send $\mathbf{X}_t$ to the decoder, $\Delta r \triangleq l(\mathbf{X}_t)$, where $l(\mathbf{X}_t)$ is the number of bits sent to the decoder.

The algorithm uses the expected distortion improvement and the expected rate cost in a rate-distortion-based cost function, defined as

$$\Delta J \triangleq \Delta d + \lambda \cdot \Delta r, \tag{3}$$

where λ is the rate-distortion parameter given to the GTR algorithm. If $\Delta J < 0$, then the expected improvement in distortion outweighs the expected cost in rate, and the algorithm inserts $\mathbf{X}_t$ into the local codebook by replacing $\mathbf{c}^*$ with $\mathbf{X}_t$. The algorithm transmits to the decoder a flag indicating whether the codebook was updated. Additionally, if the codebook was updated, the algorithm sends $\mathbf{X}_t$ as side information.

The final step of the algorithm is to estimate the new codeword probabilities using a time average. Our method of estimation assumes that the source possesses some degree of local stationarity; i.e., the statistics over a window of ω source vectors are approximately stationary. The algorithm estimates how many of the past ω source vectors have mapped to codeword $\mathbf{c}_i \in \mathcal{C}_{t-1}$ as $n_{t-1}(i) = \omega \cdot p_{t-1}(i)$. When the algorithm has determined the winning codeword, $\mathbf{c}^*$, for the current source vector, the new counts are calculated as

$$n_t(i) = \begin{cases} n_{t-1}(i), & i \neq i^*, \\ n_{t-1}(i) + 1, & i = i^*. \end{cases} \tag{4}$$

The algorithm estimates the new partition probabilities using the new codeword counts; i.e.,

$$p_t(i) = \frac{n_t(i)}{\sum_{\mathbf{c}_j \in \mathcal{C}_t} n_t(j)}, \tag{5}$$

for each $\mathbf{c}_i \in \mathcal{C}_t$. Plugging (4) into (5) yields

$$p_t(i) = \begin{cases} [\omega p_{t-1}(i)]/(\omega+1), & i \neq i^*, \\ [\omega p_{t-1}(i) + 1]/(\omega+1), & i = i^*. \end{cases} \tag{6}$$

After the codeword probabilities are updated, the index i^* is entropy coded and transmitted to the decoder. The algorithm repeats for the next source vector.

Before continuing to the move-to-front variant, several comments on the parameters of the basic GTR algorithm are in order. The parameter λ given to the algorithm is called the rate-distortion parameter as it controls the tradeoff between rate and distortion within the algorithm. The value of λ affects not only the nearest-neighbor mapping to the current source vector but also the codebook-update decision. Consequently, the value of λ ultimately determines the performance of the algorithm, in terms of rate and distortion. Indeed, varying the value of λ traces out the rate-distortion performance curve of the algorithm. Larger values of λ will focus the efforts

of the algorithm on minimizing rate over distortion, whereas smaller values of λ will result in performance with a lower distortion and a higher rate. The windowing parameter, ω, controls the relative weighting of the past versus the present in the time-average estimates of the current codeword probabilities. The value of ω has been determined to be noncritical in the performance of the algorithm [3]; we will use $\omega = 100$ throughout the experimental results of this paper. Finally, $l(\mathbf{X}_t)$ is length of the representation of current source vector $\mathbf{X}_t$ which is sent to the decoder when the codebook is updated. For the results reported in this paper, which involve image data, we assume that a uniform scalar quantizer is used to describe $\mathbf{X}_t$ to 8 bits per vector component. In this case, $l(\mathbf{X}_t) = 8N$ bits. More complicated coding schemes are discussed in [3].

The Move-to-Front Variant of the Algorithm

In the case of a codebook update in the basic GTR algorithm presented above, the algorithm adds the current source vector to the local codebook by replacing the codeword $\mathbf{c}^*$ that was determined to be the closest in a rate-distortion sense. In this section, we describe the move-to-front GTR algorithm. In this variant of the algorithm, codewords are added to the local codebook in a move-to-front fashion, effectively replacing least-recently-used (LRU) codewords. The move-to-front GTR algorithm replaces Steps 6 and 7 of the basic algorithm (Fig. 1) with the steps shown in Fig. 2 and operates as follows.

If the codebook is not updated, the algorithm simply moves $\mathbf{c}^*$ to the front (index number 1) of the codebook. However, if the codebook is updated, i.e., if $\Delta J < 0$, the algorithm places current source vector $\mathbf{X}_t$ in the front of $\mathcal{C}_t$. In this case, all the other codewords are shifted to the next highest index and the one with the highest index (the LRU codeword) is deleted.

The estimation of the new codeword probabilities is the same as in the basic algorithm, if the codebook was not updated. However, in the case of codebook update, the move-to-front insertion of $\mathbf{X}_t$ necessitates a modified calculation of the codeword probabilities. In this case, the algorithm "splits" the probability of partition i^* between the new codeword, $\mathbf{X}_t$, and $\mathbf{c}^*$. Fig. 2 gives the details of how this probability "split" is accomplished. After the new codeword probabilities are calculated, they are rearranged so as to match the move-to-front rearrangement of $\mathcal{C}_t$.

The additional computational cost of the move-to-front variant over that of the basic algorithm is negligible as the move-to-front process involves mainly simple index shuffling. However, the LRU replacement strategy of the move-to-front GTR algorithm has an advantage in that the index of the LRU codeword does not need to be transmitted to the decoder, which contributes to a slight improvement in rate-distortion performance over the basic algorithm. Consequently, we will use the move-to-front variant of the GTR algorithm exclusively in the experimental evaluation presented in the next section.

Step 6: Set $\mathcal{C}_t = \mathcal{C}_{t-1}$. If $\Delta J < 0$, go to Step 6a. Else, go to Step 6b.

Step 6a: Insert $\mathbf{X}_t$ in the front of $\mathcal{C}_t$. Increment the indices of all the other codewords. Delete the codeword with the highest index. Send to the decoder $\mathbf{X}_t$ and a flag indicating a codebook update. Go to Step 7.

Step 6b: Send the entropy-coded index i^* and a flag indicating no codebook update. Move $\mathbf{c}^*$ to the front of $\mathcal{C}_t$.

Step 7: Estimate the new codeword probabilities. If $\Delta J \geq 0$, go to Step 7a. Else, go to Step 7b.

Step 7a: Estimate the new codeword probabilities as:

$$p_t(i) = \begin{cases} [\omega p_{t-1}(i)]\,/(\omega+1), & i \neq i^*, \\ [\omega p_{t-1}(i)+1]\,/(\omega+1), & i = i^*. \end{cases}$$

Rearrange the indices of the probabilities to match the move-to-front rearrangement of $\mathcal{C}_t$; i.e., move $p_t(i^*)$ to index 1, shifting all the other probabilities. Go to Step 8.

Step 7b: Form the new codeword counts as:

$$n_t(i) = \begin{cases} \omega p_{t-1}(i), & i \neq i^* \\ \omega p_{t-1}(i)/2, & i = i^*. \end{cases}$$

Let K be the size of the local codebook; i.e., $K = |\mathcal{C}_t|$. Set $n_t(K) = n_t(i^*)$. Calculate the new codeword probabilities as:

$$p_t(i) = \frac{n_t(i)}{\sum_{1 \leq j \leq K} n_t(j)}.$$

Rearrange the indices of the probabilities to match the move-to-front rearrangement of $\mathcal{C}_t$; i.e., move $p_t(K)$ to index 1, shifting all the other probabilities.

Figure 2: The move-to-front GTR algorithm. These steps replace the corresponding steps in the basic GTR algorithm of Fig. 1.

Figure 3: Original image sequence

Experimental Results

In this section, we present the results obtained for the coding of the image sequence shown in Fig. 3 which consists of 8 image frames: 4 frames from the image sequence "Miss America" followed by 4 frames from the "Garden" sequence. Each image of the sequence is grayscale with 256 levels and has a resolution of 352×240 pixels. To produce an initial local codebook, we use an additional frame from the "Miss America" sequence as a training data set to the generalized Lloyd algorithm.

In Fig. 4, we plot the rate-distortion performance of various AVQ algorithms. We show also the performance of nonadaptive VQ. The nonadaptive vector quantizer uses the initial codebook for the entire image sequence, and its rate is the first-order entropy of the VQ indices. We present the final image of the quantized sequence for both the nonadaptive vector quantizer and the GTR algorithm in Fig. 5. For Fig. 5(b), we choose λ so that the GTR algorithm operates at approximately the same rate as the nonadaptive vector quantizer.

Conclusions

For low compression ratios (a rate of 1.5 bits/pixel or greater), most of the AVQ algorithms have distortion performance significantly better than that of nonadaptive VQ. For example, at a rate equal to that of the nonadaptive vector quantizer (about 1.5 bits/pixel), most of the AVQ algorithms achieve an MSE of around 50, which results in very little visual distinction between their quantized images. However, the AVQ algorithms achieve substantially less distortion than nonadaptive VQ at this rate. This improvement in image quality due to AVQ is illustrated in Fig. 5, where the quantized image of the GTR algorithm has much less visual distortion, particularly for edges and other areas of high detail, than the corresponding image for the nonadaptive vector quantizer.

More distinction between the AVQ algorithms is observed as the compression ratio increases. Particularly, several algorithms were unable to achieve rates below about 1.5 bits/pixel. Of those algorithms that were able to produce a coding at a rate below

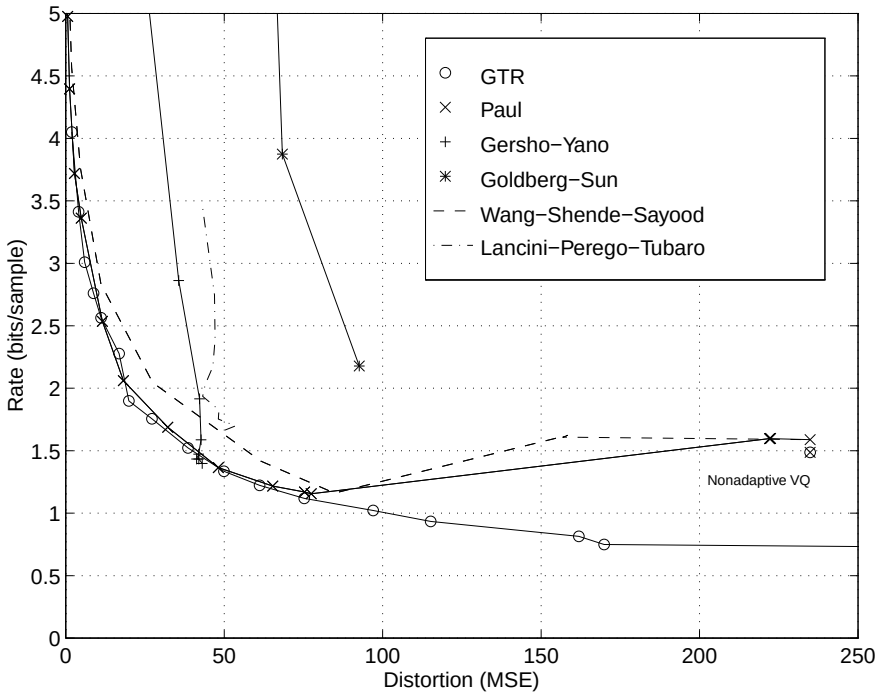

Figure 4: The rate-distortion performance of various AVQ algorithms [4–8] for the image sequence using 4-dimensional vectors and a local codebook of 256 codewords. The circled "×" represents the operation point of the nonadaptive vector quantizer for the same data.

(a) (b)

Figure 5: (a) The final frame of the image sequence coded with nonadaptive VQ (4-dimensional vectors, 256 codewords, MSE = 234.9, 1.487 bits/pixel). (b) The final frame of the image sequence coded with the GTR algorithm (4-dimensional vectors, 256 codewords, $\lambda = 16$, MSE = 38.6, 1.522 bits/pixel).

1.25 bits/pixel, only the GTR algorithm was able to maintain a monotonic decrease in rate for increasing distortion. As a consequence, GTR was the only algorithm to achieve a coding at a rate less than 1.0 bits/pixel.

In this paper, we have shown that our GTR algorithm features rate-distortion performance superior to not only that of nonadaptive VQ but also that of other AVQ algorithms for the low-rate coding of an image sequence. Although beyond the scope of this paper, similar results have been obtained for other data sources [2, 3]. These results show that the GTR algorithm consistently achieves low-rate-coding performance superior to that of other AVQ algorithms. Since many applications of current interest, such as network and wireless communications, have severe rate limitations, low-rate coding techniques will play a vital role in the successful development of systems for such applications. Our GTR algorithm will be instrumental in the incorporation of AVQ into practical, low-rate coding techniques at the heart of future communication systems designed for these demanding applications.

References

[1] J. E. Fowler, "A Survey of Adaptive Vector Quantization—Part I: A Unifying Structure," *IEEE Transactions on Communications*, November 1996. submitted.

[2] J. E. Fowler and S. C. Ahalt, "A Survey of Adaptive Vector Quantization—Part II: Classification and Comparison of Algorithms," *IEEE Transactions on Communications*, November 1996. submitted.

[3] J. E. Fowler, *Adaptive Vector Quantization for the Coding of Nonstationary Sources*. Ph.D. dissertation, The Ohio State University, 1996.

[4] D. B. Paul, "A 500-800 bps Adaptive Vector Quantization Vocoder Using a Perceptually Motivated Distance Measure," in *Conference Record, IEEE Globecom*, pp. 1079–1082, 1982.

[5] A. Gersho and M. Yano, "Adaptive Vector Quantization by Progressive Codevector Replacement," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 133–136, May 1985.

[6] M. Goldberg and H. Sun, "Image Sequence Coding Using Vector Quantization," *IEEE Transactions on Communications*, vol. COM-34, pp. 703–710, July 1986.

[7] R. Lancini, F. Perego, and S. Tubaro, "Neural Network Approach for Adaptive Vector Quantization of Images," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, (San Francisco, CA), pp. 389–392, March 1992.

[8] X. Wang, S. Shende, and K. Sayood, "Online Compression of Video Sequences Using Adaptive VQ Codebooks," in *Proceedings of the IEEE Data Compression Conference* (J. A. Storer and M. Cohn, eds.), (Snowbird, UT), pp. 185–194, IEEE Computer Society Press, 1994.

[9] M. Lightstone and S. K. Mitra, "Adaptive Vector Quantization for Image Coding in an Entropy-Constrained Framework," in *Proceedings of the International Conference on Image Processing*, vol. 1, (Austin, TX), pp. 618–622, November 1994.

[10] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-Constrained Vector Quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 31–42, January 1989.