# Vector Quantization using Artificial Neural Network Models

Aristides S. Galanopoulos
James E. Fowler, Jr.
Stanley C. Ahalt

ABSTRACT This paper describes our ongoing research into the construction of an Adaptive Vector Quantization (AVQ) image encode. We provide the motivations behind an AVQ encoder and report on our progress-to-date in realizing a Vector Quantization (VQ) encoder. We describe the hardware that has been built to compress video in real time using full-search vector quantization. This hardware implements a differential-vector-quantization (DVQ) algorithm which employs entropy-biased codebooks designed using an Artificial Neural Network (ANN). The theoretical properties of this codebook design method are discussed. We conclude with a description of the framework we are using to study various AVQ techniques with the goal of modifying our existing VQ hardware in order to realize an AVQ encoder.

## 1 Introduction

Ideally we would use *lossless* data compression for all compression tasks. Indeed, lossless compression is *required* for compression of many kinds of textual data, e.g., computer programs, documents and numerical data. Unfortunately, when applied to images lossless compression techniques achieve only modest compression rates (2:1 to 8:1). Furthermore, digital data has become more prevalent and the demand for real-time image-coding hardware, particularly for use with images, has increased dramatically.

Fortunately, for many image applications *lossy* data compression can be used as perfect reproduction is not necessary. Lossy compression techniques can achieve compression ratios can be as high as 100:1.

Vector quantization (VQ) has long been recognized as a useful technique for lossy data compression, attracting attention for its efficient compression of digitized image and speech data. However, the design of real-time vector quantizers for image coding has been challenging due to the inherent computational complexity of VQ encoders and the extremely fast speeds demanded by real-time video applications. Furthermore, the computational complexity of traditional codebook-design methods has also hindered real-time use of VQ. It has recently been shown that artificial neural networks

(ANN's) can be used to suggest real-time VLSI architectures as well as provide algorithms for the design of VQ codebooks which circumvent limitations of traditional algorithms [ACK89].

This paper describes our ongoing research into the construction of a Adaptive Vector Quantization (AVQ) encoder for images. We describe motivations behind an AVQ encoder and our progress to-date in realizing one vector-quantization encoder. In particular we describe hardware that has been built to compress video in real time using full-search vector quantization. This architecture implements a differential-vector-quantization (DVQ) algorithm which features codebooks designed using frequency-sensitive competitive learning (FSCL) [AKCM90], an ANN algorithm that attempts to maximize codebook entropy while minimizing distortion. We then describe some of the theoretical properties of this codebook design method. We conclude with a description of a framework we are currently using to study various AVQ techniques in order to modify our existing VQ hardware to realize an AVQ encoder.

## 2 Vector Quantization

A VQ system consists of an encoder, a decoder, and a transmission channel. The encoder and the decoder each have access to a fixed codebook, $\mathbf{Y}$. The codebook $\mathbf{Y}$ is a set of $Y$ codewords (or codevectors), $\mathbf{y}$, where each $\mathbf{y}$ is dimension $k$ and has a unique index, $j$, $0 \leq j \leq Y-1$. We describe our codebook design method in Section 4.3.

The image is broken into blocks of pixels called tiles. Each image tile of $n \times m$ pixels can be considered a vector, $\mathbf{u}$, of dimension $k = mn$. For each image tile, the encoder selects the codeword $\mathbf{y}$ that yields the lowest distortion by some distortion measure $d(\mathbf{u}, \mathbf{y})$. The index, $j$, of that codeword is sent through the transmission channel. If the channel is errorless, the decoder retrieves the codeword $\mathbf{y}$ associated with index $j$ and outputs $\mathbf{y}$ as the reconstructed image tile, $\hat{\mathbf{u}}$. A block diagram of a VQ system is shown in Fig. 1.

An extensive discussion of vector quantization techniques and applications has been given by Gersho and Gray[GG92], and the basic theory has been summarized in the context of image applications by others [FCA93].

### *2.1 Why use VQ rather than MPEG?*

The MPEG standard has emerged as an effective standard for image compression. However, we believe there are four reasons why VQ should still be considered as a viable candidate for wide-spread use as an image coding method:

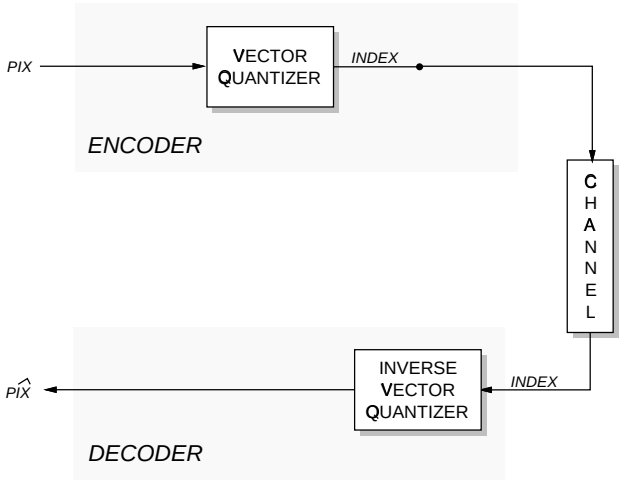


FIGURE 1. Block diagram of VQ system.

- **Errors**: because of run-length encoding and transmission ordering, nominal-MPEG is quite "error sensitive." VQ is relatively error-insensitive, and can be made more error-insensitive via codebook ordering techniques [CP93, PC95].
- **Regularity**: VQ design has been *inexpensively* implemented in real-time hardware.
- **Decoder cost**: MPEG is a symmetric algorithm, and thus both the encoder and decoder require equivalent computation. In contrast, VQ decoders are very simple.
- **Adaptive Coders**: MPEG does not provide an explicit adaptation mechanism, but Adaptive VQ (AVQ) has been under investigation for some time and adaptation mechanisms are easily incorporated into VQ.

These factors led us to build a prototype VQ encoder, which we describe below.

# 3 Differential Vector Quantization

Differential Vector Quantization (DVQ) is a combination of VQ and DPCM which replaces scalar quantization in DPCM framework with vector quantization. Consequently DVQ has many of the compression advantages of both VQ and DPCM. DVQ has been presented previously in [Rut86], where it was called vector DPCM, and in [GG92], where it was called predictive

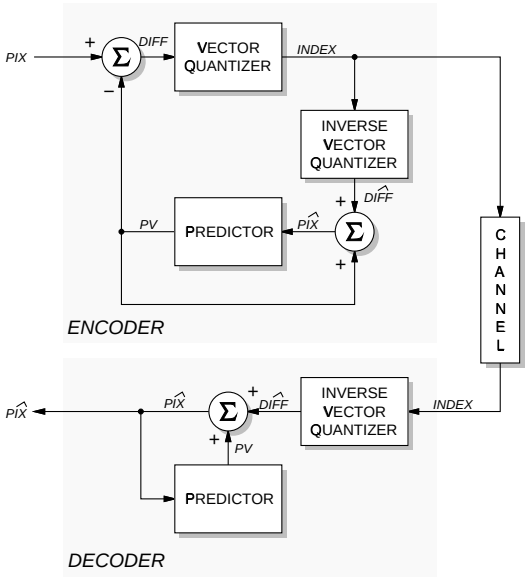


FIGURE 2. Block diagram of DVQ algorithm

VQ (PVQ). Our DVQ algorithm has been reported in detail previously [FCA93], so only a brief overview is given here.

Fig. 2 shows the general block diagram of our DVQ algorithm. In the encoding process, the predictor uses previously reconstructed tiles to predict the pixel values of the current tile. This predicted tile, $PV$, is subtracted pixel by pixel from the actual tile, $PIX$. The resulting difference tile, $DIFF$, is vector-quantized and the index, $INDEX$, is broadcast via the transmission channel to the decoder. The encoder inverse vector-quantizes $INDEX$, producing a reconstructed tile, $\widehat{PIX}$, to be used in later predictions. Note that, since the vector quantizer processes difference tiles, the VQ codebook must be appropriately derived from "difference images."

DVQ has several advantages over both scalar DPCM and VQ. Primarily, the quantization of vectors yields better compression performance than that of scalars. Additionally, since the VQ is performed on difference values rather than on the image itself, the resulting image is less "blocky" [Rut86]. Finally, the codebooks for DVQ tend to be more robust and more representative of many images than codebooks designed for VQ because the difference tiles in a DVQ codebook are more generic than the image tiles in a VQ codebook [Rut86].

The Vector-quantizing Associative Memory Processor Implementing Real-time Encoding (VAMPIRE) is a special-purpose, digital associative memory designed for video-rate vector quantization. The details of the design and operation of this chip are found elsewhere [Adk93], so only a brief overview is given here. Table 1.1 presents a summary of the VAMPIRE

TABLE 1.1. Summary of the VAMPIRE Chip

| | |
|---|---|
| Die size | 4.6 × 6.8mm |
| Technology | 2$\mu$m CMOS n-well |
| Vector Rate | $3.57 \times 10^6$ vectors/sec |
| Encoding delay* | approx. 1$\mu$s |
| Codebook size | 32 codewords on one chip; expandable to 256 with 8 chips |
| Vector dimension | Four 8-bit components |
| Power supply | 5V |

*Encoding delay is for the VAMPIRE chip operating in the DVQ architecture

chip's characteristics.

The VAMPIRE chip is designed to quantize vectors at video rates. The input to the chip is 32 bits representing a 4-dimensional vector with each vector component having 8 bits of resolution. Since these vectors are composed of four video samples, the designed throughput is that of the NTSC colorburst (3.579545MHz, or one vector every 280ns). Each VAMPIRE chip holds 32 codewords. The chips can be operated alone (for codebooks of 32 or less codewords) or can be linked together to accommodate codebooks of greater than 32 codewords.

# 4 Codebook Design and Optimal Quantizers

Obviously, since the codebook holds the vectors used to replicate all images (or at least all of the images for some period of time), the design of the codebook for a VQ encoder is quite critical to the overall effectiveness of any VQ-based coding scheme.

It has been known for some time that the necessary conditions for minimizing the average distortion, assuming a convex distortion measure, are:

- Nearest neighbor selection rule

  $$\mathbf{y}(\mathbf{x}) = \{\mathbf{y}_i \mid \|\mathbf{y}_i - \mathbf{x}\| <= \|\mathbf{y}_j - \mathbf{x}\| \text{ for all } j\}, and$$

- Centroid condition

  $$\mathbf{y}_i = \frac{\int_{C_i} \mathbf{x} p(\mathbf{x}) d\mathbf{x}}{\int_{C_i} p(\mathbf{x}) d\mathbf{x}}.$$

Thus, analytical methods for VQ codebook design require knowledge of the data pdf, which are not known in realistic cases. For unknown data

pdf, a set of $M$ **training** data vectors must be used for the design of the $N$ quantizer codevectors, where $M >> N$.

The classic and most commonly used batch codebook design algorithm is the LBG (Linde-Buzo-Gray) algorithm[LBG80], a generalization of the Lloyd-Max algorithm. This is a descent algorithm, thus it converges to a local minimum depending on the initial state.

Unfortunately the computational complexity and memory requirements of such traditional VQ codebook design methods has restricted their use in real-time applications [LBG80]. It has been shown that ANN's can be used to design VQ codebooks and circumvent the limitations of traditional algorithms [ACK89], for the following reasons. First, ANN's consist of a large number of simple, interconnected computational units that can be operated in parallel. Second, ANN-codebook-design algorithms do not need access to the entire training data set at once during the training process. It should also be noted that these features make ANN algorithms ideally suited for the design of adaptive vector quantizers [FCA93], which are discussed in Section 5.

### *4.1 Competitive Learning*

The simplest ANN algorithm for VQ codebook design is Competitive Learning (CL). At time $t$ (discrete) a data vector $\mathbf{x}(t)$ is presented and the following steps are executed:

- find the *winner*, the codevector $\mathbf{y}_w$ which is closest to the input vector:

$$\|\mathbf{y}_w(t) - \mathbf{x}(t)\| <= \|\mathbf{y}_i(t) - \mathbf{x}(t)\| \text{ for all } i.$$

- update the winner as:

$$\mathbf{y}_w(t+1) = \mathbf{y}_w(t) + a(t) * (\mathbf{x}(t) - \mathbf{y}_w(t))$$

  where $a(t)$ is the *learning rate* at time $t$.

It can be shown that, for CL, the necessary conditions for convergence to a local equilibrium,

$$\lim_{t\to\infty} a(t) = 0, and \sum_{t=1}^{\infty} a(t) = \infty,$$

are sufficient for convergence to a local equilibrium if the initial state is inside the domain of attraction of some equilibrium (from stochastic approximation theory).

Unfortunately, this simple algorithm is easily trapped in local minima. In many realistic practical applications, these minima are the result of *codeword underutilization*. Codeword underutilization occurs when a codeword

is never designated as the closest codeword to an input vector,and is never updated. Two mechanisms have been proposed to overcome this problem: Kohonen's Self-Organizing Feature Maps (KSFM), and Conscience techniques including Frequency Sensitive Competitive Learning (FSCL). Both of these techniques are described below.

## *4.2 Kohonen Self-Organizing Feature Maps*

In a KSFM, a topology $\mathcal{B}$, (usually two-dimensional) is associated with the codevectors. Once again a data vector, $\mathbf{x}(t)$, is presented and the following steps are executed:

- the winner $\mathbf{y}_w(t)$ is selected as the codevector closest to the input vector $\mathbf{x}(t)$
- *All* codevectors inside a neighborhood of the winner, defined on $\mathcal{B}$, are updated towards the input vector as:

$$\mathbf{y}_n(t+1) = \mathbf{y}_n(t) + a(t) * h(n,w) * (\mathbf{x}(t) - \mathbf{y}_n(t)),$$

  where $h(n,w)$ is usually a decreasing function of the distance $\|\mathbf{y}_n - \mathbf{y}_w\|_{\mathcal{B}}$, and the size of the winner's neighborhood also decreases with time.

The KSFM technique has been shown to utilize all codevectors, and thus solves the underutilization problem. Further, it can be shown that necessary and sufficient conditions for convergence, as in the case of CL, are: [CF86, RS88]

$$\lim_{t\to\infty} a(t) = 0, and \sum_{t=1}^{\infty} a(t) = \infty.$$

Finally, equilibrium results for one dimensional data spaces have been derived [Rit91], and the equilibrium codevector density is proportional to a power of the data pdf which depends on the size of the winner's neighborhood.

Unfortunately, while KSFMs solve the underutilization problem and have been shown to have attractive theoretical properties, they exhibit certain problems. First, because of the use of neighborhoods, $\mathcal{B}$, they are computationally more expensive. Second, the selection of the neighborhood topology can be difficult for many dimensional data spaces. The FSCL algorithm circumvents these problems.

## *4.3 Frequency Sensitive Competitive Learning*

The Frequency Sensitive Competitive Learning (FSCL) method is an ANN method which features a modified distortion measure that ensures all code-

words in the codebook are updated equally frequently during iterations of the training process.

This is accomplished by assigning a counter $c_n(t)$ to each codevector $\mathbf{y}_n(t)$ that is incremented every time $\mathbf{y}_n(t)$ wins the competition. The winner, $\mathbf{y}_w(t)$, is selected as the codevector that minimizes the product of the distortion measure and a **fairness function** $F$, i.e.,

$$F(c_n(t))\|\mathbf{y}_n(t) - \mathbf{x}(t)\|.$$

The usual form of the fairness function can be expressed as:

$$F(c) = c^{\beta(t)}, \quad \text{with } \lim_{t\to\infty} \beta(t) = 0.$$

It has been shown that codebooks designed with FSCL yield mean squared errors and signal-to-noise ratios comparable to those of the locally optimal LBG algorithm [ACK89]. Also, FSCL yields codebooks with sufficient entropy so that Huffman coding of the VQ indices would not provide significant additional compression [FCA93]. Finally, FSCL overcomes underutilization problems, is relatively efficient, and is well-suited for adaptive applications.

Recently we have been able to establish the convergence and equilibrium properties of the FSCL algorithm [GA95, GMA95]. We outline those results below.

FSCL convergence

First, in order to establish FSCL convergence, we can model the FSCL network as a Markov process with state

$$\sigma(t) = \begin{bmatrix} \mathbf{y}_1(t) & \cdots & \mathbf{y}_N(t) \\ f_1(t) & \cdots & f_N(t) \end{bmatrix}$$

where $f_i(t) = c_i(t)/t$ is the update frequency, $c_i(t)$ is the update count, and $\mathbf{y}_i(t)$ is the position of codevector $i$ at time $t$. The winning codevector minimizes

$$F(f_i(t))\|\mathbf{y}_i(t) - \mathbf{x}(t)\|$$

where we consider fairness functions of the form

$$F(f_i) = f_i^{\beta}$$

with $\beta$ a positive parameter. The counter $c_w(t)$ of the winner is incremented and its position $\mathbf{y}_w(t)$ updated as

$$\mathbf{y}_w(t+1) = \mathbf{y}_w(t) + a(t)(\mathbf{x}(t) - \mathbf{y}_w(t))$$

where $\mathbf{x}(t)$ is the input data vector at time $t$, and $a(t)$ is the learning rate.

We then formulate the evolution of an ensemble of systems in terms of the probability $Q(\sigma, \sigma')$ of transition from state $\sigma$ to state $\sigma'$ and describe

the evolution of the process through a linear time-dependent Fokker-Plank equation.

With suitable approximations it can be shown that the mean and the covariance matrix of the codevectors' deviation from the equilibrium must vanish as $t \to \infty$ with necessary and sufficient conditions:

$$\lim_{t\to\infty} a(t) = 0, \qquad \text{and} \qquad \int_{t_0}^{\infty} a(t)dt = \infty.$$

Once again we note that these conditions are, satisfyingly, the same as those needed for CL and KSFM.

FSCL Equilibrium

In order to establish the equilibrium properties of the FSCL codebook design technique, we simplify the analysis to one-dimensional input data and a large number of codevectors. While we omit the details here, we can then show that, for fairness functions of the form $F(f_i) = f_i^{\beta}$, the equilibrium codeword density is proportional to a *power* of the input data pdf , i.e.,

$$(\text{codevector density}) \propto (\text{data density})^{\frac{3\beta+1}{3\beta+3}}.$$

and FSCL minimizes the $L_q$-norm distortion measure with

$$q = \frac{2}{3\beta+1} \in (0, 2].$$

These results show that the FSCL algorithm can be used to optimize a larger class of distortion measures compared with previous VQ clustering algorithms, at least for a one-dimensional input space. It should be noted that, as far as we know, only one-dimensional equilibrium results have been calculated for any ANN clustering algorithm.

# 5 Adaptive Vector Quantization

Our ultimate objective, which we are now investigating, is to modify our VQ encoder with a suitable adaptation mechanism such that an Adaptive Vector Quantizer (AVQ) is realized. More precisely, we wish to realize an encoder that codes a non-stationary source with unknown statistics at an average of $R$ bits per symbol, such that the average distortion per symbol is minimal. Among the open questions we hope to answer are 1) what is the optimal distortion that can be realized (by any coding), and 2) how close can AVQ come?

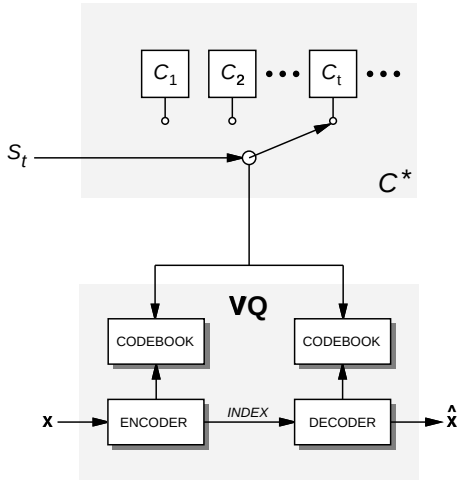


FIGURE 3. Block diagram of AVQ algorithms

To structure the discussion, assume a Large (possibly infinite) *universal* codebook, $\mathcal{C}^* \subseteq \Re^k$. [1] We'll assume that the AVQ encoding uses a *local* codebook, $\mathcal{C}_t \subset \mathcal{C}^*$ and that the AVQ encoding is time-variant:

$$Q_t : \Re^k \rightarrow \mathcal{C}_t$$

$$\hat{\mathbf{X}}_t = Q_t(\mathbf{X}_t) \in \mathcal{C}_t$$

This requires some local codebook selection mechanism:

$$s_t : \mathcal{C}^* \rightarrow \{0, 1\}$$

$$\text{such that} \quad \mathcal{C}_t = \{\mathbf{y} \mid s_t(\mathbf{y}) = 1,\ \mathbf{y} \in \mathcal{C}^*\}.$$

Now, assuming that $\mathcal{C}^*$ is finite, let $N = |\mathcal{C}^*|$ and $S_t = s_t(\mathbf{y}_1)s_t(\mathbf{y}_2)\cdots s_t(\mathbf{y}_N)$, where $\mathbf{y}_i \in \mathcal{C}^*$. Then $S_t$, the codebook-selection process, is an integer-valued random variable $0 < S_t \leq 2^N - 1$.

A pictorial view of this model is shown below in Figure 3.

We have found it useful to classify all of extant AVQ encoders into two classes of universal codebook encoders. We refer to these as:

- *A priori* codebooks in which:
  - Both encoder and decoder have complete knowledge of $C^*$, and
  - we assumes we know enough about the source to set $C^*$ before coding commences, or

[1] Please note that we have changed notation here. The codebook, $\mathcal{C}^*$, plays the same role as that denoted $\mathbf{Y}$ previously. We adopt the use of $\mathcal{C}^*$ to emphasize that this is an *adaptive* codebook.

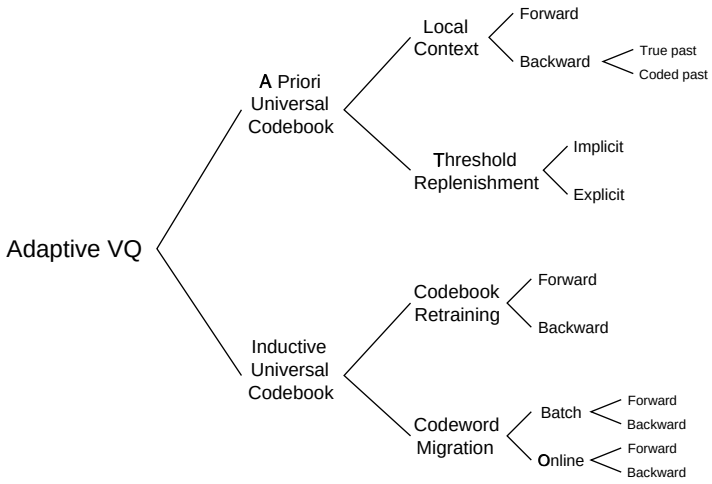


FIGURE 4. Taxonomy of AVQ algorithms

- *Inductive* codebooks in which:
  - Encoder and decoder only know initial local codebook $C_1$
  - The rest of $C^*$ is *induced* from observing the source.

With this fundamental differentiation, we have established a taxonomy of current AVQ encoders, as shown in Fig. 4. We are currently reviewing the literature to determine if this taxonomy is sufficient for all previously proposed AVQ techniques. We anticipate that this taxonomy will prove useful for our subsequent efforts at implementing an AVQ system consistent with our current DVQ algorithm.

# 6 Conclusions

In this paper, we have described our DVQ algorithm and presented a hardware architecture implementing the algorithm. We have shown that the codebook design technique we employ, FSCL, has attractive theoretical properties, and appears to be suitable for use with non-stationary sources. Finally, we have shown that previous AVQ investigations can be described in a convenient mathematical framework, and classified according to a simple taxonomy.

Our ongoing work is concentrated on determining what AVQ mechanisms are needed to approach optimal AVQ, and determining if these mechanisms are computationally feasible in real-time, inexpensive hardware.

# 7 References

[ACK89] Stanley C. Ahalt, Prakoon Chen, and Ashok K. Krishnamurthy. Performance Analysis of Two Image Vector Quantization Techniques. In *IJCNN*, volume I, pages 169–175, Washington, D.C., June 18–22, 1989.

[Adk93] Kenneth C. Adkins. *The VAMPIRE Chip: A Vector-quantizer Associative Memory Processor Implementing Real-time Encoding.* PhD thesis, The Ohio State University, 1993.

[AKCM90] Stanley C. Ahalt, Ashok K. Krishnamurthy, Prakoon Chen, and Douglas E. Melton. Competitive Learning Algorithms for Vector Quantization. *Neural Networks*, 3:277–290, 1990.

[CF86] M. Cottrell and J. C. Fort. A stochastic model of retinotopy: a self organizing process. *Biological Cybernetics*, 53:405–411, 1986.

[CP93] Da-Ming Chiang and Lee C. Potter. Minimax Non-Redundant Channel Coding for Vector Quantization. In *ICASSP*, pages V–617–V–620, Minneapolis, MN, April 1993.

[FCA93] James E. Fowler, Matthew R. Carbonara, and Stanley C. Ahalt. Image Coding Using Differential Vector Quantization. *IEEE Trans. on Circuits and Systems for Video Tech.*, 3(5):350–367, October 1993.

[GA95] Aristides S. Galanopoulos and Stanley C. Ahalt. Codeword Distribution for Frequency Sensitive Competitive Learning with One Dimensional Input Data. *IEEE Trans. on Neural Networks*, 1995.

[GG92] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression.* Kluwer international series in engineering and computer science. Kluwer Academic Publishers, Norwell, MA, 1992.

[GMA95] Aristides S. Galanopoulos, Randolph M. Moses, and Stanley C. Ahalt. Convergence Conditions for Frequency Sensitive Competitive Learning. *IEEE Trans. on Neural Networks*, 1995.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Trans. on Comm.*, COM-28(1):84–95, January 1980.

[PC95] Lee C. Potter and Da-Ming Chiang. Minimax Nonredundant Channel Coding. *IEEE Trans. on Comm.*, 43(2/3/4):804–811, February/March/April 1995.

[Rit91] Helge Ritter. Asymptotic level density for a class of vector quantization processes. *IEEE Trans. on Neural Networks*, 2(1), January 1991.

[RS88] J. Ramanujam and P. Sadayappan. Parameter Identification for Constrainted Optimization using Neural Networks. In *Proc. 1988 Connnectionist Models*, pages 154–161, Carnegie Mellon Univ., June 1988.

[Rut86] Charles W. Rutledge. Vector DPCM: Vector Predictive Coding of Color Images. In *Proc.IEEE Global Telecomm. Conf.*, pages 1158–1164, September 1986.