A Successive Approximation Vector Quantizer

for Wavelet Transform Image Coding

Eduardo A. B. da Silva, Demetrios G. Sampson, Member, IEEE, and Mohammad Ghanbari, Member, IEEE

Abstract—A coding method for wavelet coefficients of images using vector quantization, called successive approximation vector quantization (SA-W-VQ), is proposed. In this method, each vector is coded by a series of vectors of decreasing magnitudes until a certain distortion level is reached. The successive approximation using vectors is analyzed, and conditions for convergence are derived. It is shown that lattice codebooks are an efficient tool for meeting these conditions without the need for very large codebooks. Regular lattices offer the extra advantage of fast encoding algorithms. In SA-W-VQ, distortion equalization of the wavelet coefficients can be achieved together with high compression ratio and precise bit-rate control. The performance of SA-W-VQ for still image coding is compared against some of the most successful image coding systems reported in the literature. The comparison shows that SA-W-VQ performs remarkably well at several bit rates and in various test images.

I. INTRODUCTION

VECTOR quantization (VQ) is a generalization of scalar quantization (SQ), which enables us to move from the separate quantization of each individual sample to the joint quantization of a group of samples as one unit. Shannon’s original work on source coding with a fidelity criterion [1] suggests that rate-distortion bounds can be asymptotically approached by vector quantizers of arbitrarily large dimensions [2]. This theoretical advantage of vector quantization over scalar quantization has been the motivation for investigating VQ as a data compression method.

Since the early 1980’s, vector quantization has attracted the interest of both academia and the data compression industry. It is now considered one of the most powerful tools for audio, speech, image, and video compression [3]–[7]. The most important features of VQ can be summarized as follows:

1) simplicity of the decoder
2) potential of a fractional-bit allocation for the vector components
3) ability to exploit the statistical correlation between neighboring data in a straightforward manner.

An important part of the design of a vector quantizer is the construction of the VQ codebook, since it affects both coding efficiency and implementation complexity of the coder. Linde, Buzo, and Gray [8] have developed a constructive method for VQ codebook design that is a systematic generalization of Lloyd’s method I for designing optimum scalar quantizers [9]. Their method, the generalized Lloyd algorithm (GLA) or LBG algorithm, generates a locally optimal codebook by iteratively improving an initial codebook with respect to a given training sequence. Typically, the VQ codebook designed using GLA consists of quantization regions with nonregular shapes and sizes, so that the best available code vector can be typically selected after comparing a given input vector against all available code vectors in the codebook.

Encoding complexity is a major drawback in the real-time implementation of full-search VQ, since the number of computations required for the selection of the closest code vector increases exponentially with the vector dimensions and the coding rate. Several methods have been put forward in VQ literature to tackle the problem of VQ encoding complexity [3]–[6]. Typically, these methods involve imposing a certain structure to the VQ codebook so that unconfined access to all effective code vectors is restricted. Tree-structured VQ, multistage VQ, product-code VQ, classified VQ, and finite-state VQ fall into this category. They have proved to be very successful for both image and video coding applications [10]–[16].

An alternative method for constructing VQ codebooks has been proposed by Conway and Sloane [17]. According to this method, known as lattice vector quantization (LVQ), the VQ codebook is built based upon a regular lattice. A well-known result of high-rate quantization theory is that uniform quantizers with perfect entropy coding can achieve near-optimum performance for memoryless sources [2]. Gerstho [18] conjectured that these optimum entropy-constrained VQ’s are lattice quantizers, because they provide the best known space-filling properties [17]. In practice, the most promising feature of lattice vector quantization is the existence of simple and fast encoding/decoding methods based on the structural properties of regular lattices [19]. Thus, a lattice VQ can offer a significant reduction in the encoding complexity typically required by full-search GLA-designed VQ. Nevertheless, it is not always possible to design an efficient entropy coder for a very large codebook.

There is a plethora of applications of GLA-designed VQ’s on image coding. Nasrabadi and King [6] provide an excellent review of the subject. On the other hand, there are relatively few attempts to design lattice vector quantizers for image and video coding applications [20]–[26]. These techniques aim to
construct lattice codebooks that match the statistical properties of the memoryless sources with independent and identically distributed (i.i.d.) variables. To accomplish this, we employ the truncation and scaling of the original infinite regular lattice so that the selected lattice code vectors will be located in the areas of high source density. Hence, different truncation methods have been proposed, namely spherical [22], [25] and pyramidal [26]–[28] to match with the geometrical properties of Gaussian and Laplacian i.i.d. sources, respectively. After truncation, the selected lattice code vectors are scaled to minimize the distortion for the given input source. This is accomplished either experimentally [20], [22], [25] or analytically [28]. Application of these lattice quantizers in image coding requires a careful preprocessing of the input data, so that the assumption of i.i.d. sources is satisfied.

In this paper, we propose a novel image-coding method that employs successive approximation lattice vector quantization to code the wavelet coefficients of still images. The advantage of this method is that only a limited number of lattice code vectors are used so that they can be efficiently encoded using an adaptive arithmetic coder. Thus, instead of using a very large lattice codebook with the obvious difficulties of entropy coding, we suggest the successive approximation of input vectors using only a finite set of lattice code vectors, referred to as successive approximation vector quantization (SA-VQ). The paper is organized as follows. Section II gives the necessary definitions from lattice theory, emphasizing the properties of regular lattices, which are particularly useful in the proposed coding scheme. In Section III, after a short introduction to wavelet transforms, the most interesting features of the application of wavelet transforms in image coding are discussed, and design considerations of an efficient method for coding wavelet transform coefficients are outlined. In Section IV, we define the problem of successive approximation using a finite set of vectors and then derive conditions for convergence. Based on these conditions, it is shown that lattice codebooks are well-suited to this task. A wavelet transform image coder has been developed using vector successive approximation, referred to as successive approximation wavelet VQ (SA-WVQ). The main elements of this coder are described in Section V. In Section VI the performance of the proposed coder is tested for still image coding and our simulation results are compared with several other methods for various bit rates and different test images.

II. LATTICE QUANTIZATION

A. Definitions

A regular lattice is a discrete set of points in the $k$-dimensional Euclidean space $\mathbb{R}^k$, which can be generated by the integral linear combination of a given set of basis vectors. Hence, a $k$-dimensional lattice $L_k$ is defined as the subset of real space $\mathbb{R}^k$, such that

$$L_k = \{ \vec{y} \in \mathbb{R}^k : \vec{y} = u_1 \vec{a}_1 + u_2 \vec{a}_2 + \ldots + u_k \vec{a}_k \} \quad (1)$$

where $\{ \vec{a}_i \}$ is the set of linearly independent vectors that span $L_k$, called basis vectors of lattice $L_k$, and $\{ u_i \}$ is the set of integers that specify a particular point in lattice $L_k$, known as coefficients of the basis vectors.

Regular lattices have been originally investigated in the context of sphere packing. Sphere packing concerns the densest way of arranging $k$-dimensional, identical, nonoverlapping spheres in real space [29]. More formally, a sphere packing $L_k$ of radius $\rho$ consists of an infinite set of points $\{ \vec{y}_1, \vec{y}_2, \ldots \}$ in the Euclidean space $\mathbb{R}^k$ such that the minimum distance between any two points is not smaller than double the radius of the packing [30], as follows:

$$\text{dist}(\vec{y}_i, \vec{y}_j) = \sqrt{\sum_{m=1}^{k} (y_{im} - y_{jm})^2} \geq 2\rho, \forall i \neq j. \quad (2)$$

Thus, a sphere packing is described by specifying the centers $\{ \vec{y}_i \}, i = 1, 2, \ldots$ and the radius $\rho$ of the $k$-dimensional spheres. An important parameter in sphere packing is the "kissing" number $\tau(\vec{y})$ of a sphere with its center at the point $\vec{y}$ and radius $\rho$. The $\tau(\vec{y})$ counts the total number of spheres of the same packing $L_k$ that are in contact with the particular sphere. The maximum value of $\tau(\vec{y})$ for any $\vec{y} \in L_k$ is denoted by $\tau_{\text{max}}$. Lattice packing is a sphere packing where the sphere centers are points of a particular lattice. The most significant property in lattice packing is that, if there are spheres with centers $\vec{y}_i$ and $\vec{y}_j$, then there are also spheres with centers $\vec{y}_i \pm \vec{y}_j$. This is a result of the regular structure of lattices (regular lattices are an additive group). Hence, in lattice packing, $\tau(\vec{y}) = \tau_{\text{max}}$ always [29].

B. Important Regular Lattices

Root lattices comprise an important category of lattices, namely, $Z_k(k > 1)$, $A_k(k > 1)$, $D_k(k > 3)$, $E_k(k = 6, 7, 8)$, and the Barnes–Wall $\Lambda_{16}$ and Leech $\Lambda_{24}$, which have been shown to offer the best lattice packing of their space [29]. Definitions of the lattices used in this work are given below.

1) The Integer Lattice—$Z_k$: The integer or cubic lattice $Z_k(k > 1)$ is defined as the set of $k$-dimensional vectors with all integer components $Z_k = \{ \vec{y} = (y_1, y_2, \ldots, y_k) : y_i \in \mathbb{Z} \}$ where $\mathbb{Z}$ is the set of integer numbers. Lattice $Z_k$ gives the simplest structure of points in $\mathbb{R}^k$ and most regular lattices can be generated from $Z_k$.

2) The Lattice $D_k$: The $k$-dimensional lattice $D_k(k > 3)$ is defined by spanning the integer lattice $Z_k$ and retaining those points $\vec{y}$ in $Z_k$ that have coordinates with an even sum.

$$D_k = \{ \vec{y} : y_i \in \mathbb{Z} \wedge \sum_{i=1}^{k} y_i = 0 (\text{mod} 2) \}. \quad (3)$$

$D_k$ is very significant for quantization purposes since it is the backbone of other more complex lattices that give the most dense packing at high dimensions, namely the Gosset $E_8$ and the Barnes–Wall $\Lambda_{16}$.

3) The Lattice $E_k(k = 6, 7, 8)$: The most dense lattices in $k = 6, 7,$ and 8 dimensions are the members of the $E_k(k = 6, 7, 8)$ family. Among them the Gosset $E_8$ is particularly useful for quantization purposes due to its symmetrical structure. It is defined as the union of two subset of points, the lattice $D_8$ and the coset $\{ D_8 + \frac{1}{2} \}$, as follows:
\[ E_8 = D_6 \cup \{D_8 + \frac{\bar{c}}{2}\} \]

where

\[ \frac{\bar{c}}{2} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}) \tag{4} \]

4) The Barnes–Wall lattice \( \Lambda_{16} \): The Barnes–Wall lattice \( \Lambda_{16} \) is the most dense lattice at \( k = 16 \) dimensions. To exploit the fast quantization algorithm available for \( D_8 \), \( \Lambda_{16} \) can be conveniently defined as the union of 32 cosets of the lattice \( 2D_{16} \). The scaled lattice \( 2D_{16} \) is the set of even coordinates points in \( Z_{16} \) such that the sum of the coordinates is a multiple of four. Thus, \( \Lambda_{16} \) is defined as

\[ \Lambda_{16} = \bigcup_{i=1}^{32} (c_i + 2D_{16}) \tag{5} \]

where the coset representatives \( c_i \) are codewords of the rows of the Hadamard matrix \( H_{16} \) and its complimentary \( H_{16} \) after changing the 1’s to 0’s and the -1’s to 1’s.

C. Fast Lattice Quantization Algorithms

The main advantage of lattice vector quantization is the very fast encoding process. When the VQ codebook is built based on a finite set of lattice vectors, a nearest neighbor (NN) search is carried out only among a limited number of code vectors (depending on the properties of the particular lattice) as opposed to the exhaustive full codebook search of the conventional clustering VQ. Conway and Sloane [19] have developed fast and simple NN algorithms for all the important regular lattices. Their algorithms exploit the symmetry of the root lattices to find the closest lattice point for a given input vector with minimum computational effort, assuming an infinite lattice. However, the design of a lattice quantizer in practice demands the infinite root lattice to be truncated and scaled. Hence, certain modifications to the original algorithms are needed to deal with a scaled version of the lattices, as well as with the points outside the boundary regions of a truncated lattice. Such modifications can be found in [26], [28], [31] and they are not described here.

III. IMAGE CODING USING WAVELET DECOMPOSITION

A. Wavelet Transforms

A discrete biorthogonal wavelet transform of a function \( x(t) \), represented by the coefficients \( \hat{x}_{m,n} \), is its decomposition on expansions and translations of a mother function \( \psi(t) \) [32], such that

\[ x(t) = \sum_{m,n} \hat{x}_{m,n} 2^{-m/2} \psi(2^{-m}t - n) \tag{6} \]

\[ \hat{x}_{m,n} = \int_{-\infty}^{\infty} 2^{-m/2} \psi(2^{-m}t - n)x(t)dt. \tag{7} \]

It can be shown that a discrete wavelet transform can be computed via an octave band subband decomposition where the filter coefficients are derived from the wavelets \( \psi(t) \) and \( \psi(t) \) [33]. The Z transforms of the analysis lowpass and highpass filters \( H_0(z) \) and \( H_1(z) \) are related to those of the synthesis lowpass and highpass filters \( G_0(z) \) and \( G_1(z) \) [33], by

\[ H_0(z)H_1(-z) - H_0(-z)H_1(z) = cz^{-2m+1} \tag{8} \]

\[ G_0(z) = \frac{2}{c} 2^{-m-1} H_1(-z) \tag{9} \]

\[ G_1(z) = \frac{2}{c} 2^{-2m-1} H_0(-z) \tag{10} \]

where \( c \) is an arbitrary constant.

The set of wavelets \( \psi(2^{m}t - n) \) is orthogonal to the set of wavelets \( \psi(2^{-m}t - n) \), i.e., they comprise a biorthogonal set. An orthogonal wavelet transform is a special case of (6) and (7) when \( \psi(t) = \bar{\psi}(t) \). However, orthogonal wavelets cannot have linear phase [34], which makes them less suitable than the biorthogonal ones for image processing applications, 2-D wavelet transforms can be obtained by separable 1-D transforms applied to the rows and columns of an image [35], resulting in a decomposition as shown in Fig. 1(a).

B. Wavelet Transforms in Image Coding

Wavelet transforms have been attracting great interest in the image coding community in the last few years. They are mainly used to decorrelate image data so that the resulting coefficients can be efficiently coded using a technique like scalar quantization [35]–[39], vector quantization [40]–[42], or other methods [43], [44]. Among these, the vector quantization-based methods tend to be very successful. The structure of the wavelet coefficients, where many zero-valued coefficients are clustered through the bands, is well suited to the fractional bit allocation achievable with VQ. Also, the inherent property of VQ to exploit the statistical correlation among neighboring samples matches well with the spatial orientation of the wavelet bands. For example, horizontal bands can be coded mostly with horizontally oriented vectors, vertical bands with vertically oriented ones, and so on, so that codebooks for each band or orientation can be efficiently designed [40].

In addition to the decorrelation of the image data, 2-D wavelet transforms have another important property. Despite the low correlation among themselves, bands of same orientation look like scaled versions of each other. That is, their edges are approximately in the same corresponding positions, hence, their nonsignificant coefficients are approximately in the same corresponding locations. In Fig. 1(a), if the coefficient \( b_k(i,j) \) in band \( B_k \) is zero, it is likely that the coefficients \( b_{k+1}(2i,2j), b_{k+1}(2i+1,2j), b_{k+1}(2i,2j+1) \), and \( b_{k+1}(2i+1,2j+1) \) in band \( B_{k+1} \) will also be zero where \( B \) can be \( V \), \( H \) or \( D \) [35]. This resemblance can be seen in Fig. 1(b), where a three-stage wavelet transform of the Lena image is shown.

In terms of coding the wavelet coefficients, the above-mentioned similarity can be exploited to provide an efficient addressing of the nonzero coefficients. When \( b_k(i,j) \) and all its corresponding coefficients in bands \( B_r, r > k \) are zero (or have magnitudes below a certain threshold), instead of transmitting all the zero values, one can simply mark \( b_k(i,j) \) as a zero tree root. Since this is a likely event due to the similarity among the bands of same orientation, savings in terms of bit rate can be quite significant. Zero tree roots have been successfully applied to wavelet coefficients with scalar quantization [35],
relative levels of distortion among the bands in order for them to match human visual system (HVS) sensitivity for each band. An analytical description of the distortion equalization of orthogonal transform coefficients according to HVS frequency responses is given in [48]. This method can be easily extended to account for biorthogonal transforms.

Another property of wavelet transforms is that a quantization error \( e_{m,n} \) introduced to coefficient \( x_{m,n} \) will appear as a scaled version of the wavelet \( \psi(2^{-m}t - n) \) superimposed on the reconstructed signal. This can be deduced from (6). In image coding, this implies that a quantization error from a coefficient located, for example, at an edge, will not be just confined to that edge but will be spread through the reconstructed image with the shape of the corresponding wavelet, causing an annoying ringing. Therefore, edge-masking effects cannot easily be exploited when coding coefficients, and quantization distortion in every individual coefficient can be important to final image quality.

C. Design Considerations of a Wavelet Image Coder

To summarize the previous section, efficient coding of wavelet coefficients requires the following:

- (i) exploitation of the similarities among the bands of same orientation
- (ii) setting an arbitrary average level of distortion for each band (noise shaping)
- (iii) the quantization distortion in each coefficient should not exceed a certain maximum

Requirement (i) above can be achieved by using some of the techniques developed in either [35] or [39] for scalar quantization. One way to achieve requirements (ii) and (iii) is to code the wavelet coefficients in successive passes whereby, in each pass, the quantization error is further refined and a maximum error in each coefficient can be guaranteed. This clearly fulfills requirement (iii). Also, if at each pass we can guarantee a certain level of distortion for each coefficient, the overall level of distortion in each band can also be guaranteed. Therefore, if the bands are properly weighted prior to coding, noise shaping can be readily obtained.

Successive approximation of wavelet coefficients with scalar quantization has been efficiently implemented by Shapiro [39]. In this paper we propose a successive approximation vector quantization of wavelet coefficients. The proposed method satisfies requirements (ii) and (iii) while exploiting the advantages of vector quantization, described in Section I. In the following section, an analysis of the vector-successive approximation problem is made and sufficient conditions for its feasibility are derived.

IV. SUCCESSIVE APPROXIMATION OF VECTORS

A. Definition of the Problem

In order to state the problem clearly, we first consider the process of successive approximation in the scalar case. This is equivalent to approximating a given length \( L \) by using at each pass yardsticks of increasingly smaller lengths until a certain level of error is obtained. Fig. 2 illustrates this process for the case where the length \( l \) of the yardstick is always
Fig. 2. Successive approximation of a coefficient for the scalar case.

halved after each pass. The process begins by choosing an initial yardstick length \( \ell \) such that \( \ell > L \). It can be inferred from this figure that after each pass the error is bounded by the yardstick length, which becomes smaller at each pass (for example, when the yardstick length is \( \frac{\ell}{2} \), the error is bounded by \( \frac{\ell}{2^2} \)). With reference to Fig. 2, the length \( L \) can be expressed as:

\[
L = +\ell - \frac{\ell}{2} + \frac{\ell}{4} - \frac{\ell}{8} + \frac{\ell}{16} + \frac{\ell}{32} \ldots
\]

Therefore, the length \( L \) can be represented as a string of “+” and “−” symbols. As each symbol “+” or “−” is added, precision in the representation of \( L \) increases, and distortion level decreases. Generalization of this process of successive approximation to \( k \)-dimensional space is not a straightforward task. In general, a \( k \)-dimensional vector is defined by two parameters: its length, which is a scalar value that corresponds to the norm of the vector, and its orientation, which is a \( k \)-dimensional vector with unit energy. Successive approximation of vectors should consider both of these components (length and orientation) rather than only length as it is in the scalar case (\( k = 1 \)).

Thus, instead of yardsticks of decreasing lengths, we deal with “vector yardsticks” having decreasing lengths and given orientations in a \( k \)-dimensional space. In practice, these vector yardsticks will be chosen from a finite codebook and, therefore, the set of possible orientations is finite. At each pass, the vector yardsticks will aim to approximate the residual vector formed as the difference between the original vector \( \vec{V} \) and its approximation so far. Fig. 3(a) illustrates an example in 2-D space, where \( \vec{V} \) is approximated by a series of vector yardsticks \( \vec{y}_i \) of decreasing lengths. From this example it is easy to realize that this process does not necessarily lead to convergence, as in the trivial scalar case. The question that naturally arises is how one can guarantee that the vector approximation process converges—that is, if the number of passes is sufficiently large, that the magnitude of the residual vector will be always smaller than an arbitrary value.

**B. Conditions for Convergence**

In this section we devise sufficient conditions for the convergence of the successive approximation of vectors using a finite set of orientation vectors of decreasing length. The following suppositions have been made.

1. The orientation codebook, \( \mathbf{Y} \) is a finite set of \( k \)-dimensional vectors with unit energy, such that

\[
\mathbf{Y} = \{ \vec{y}_i : \| \vec{y}_i \| = 1; i = 1, 2, \ldots, N \}.
\]

At each pass, a new vector yardstick is formed as the product of the current yardstick length and one of the unit orientation code vectors \( \vec{y}_i \).

2. The orientation codebook is built so that the angle\(^1\) between any possible vector and its closest orientation code vector is upper bounded by \( \theta_{\text{max}} \). Hence, at each pass the maximum error is introduced when the residual vector is approximated by a vector with orientation \( \theta_{\text{max}} \).

3. The yardstick length at each pass will be scaled by a constant factor \( \alpha \), the so-called approximation scaling factor, in the range of \( 0.5 \leq \alpha < 1.0 \).

4. For a given vector \( \vec{V} \), the approximation process is activated for the first time at a certain pass indexed by

\[\text{cos } \theta = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \| \| \vec{y} \|}.
\]

\(^1\) In \( k \)-dimensional space, the angle \( \theta \) between two vectors \( \vec{x} \) and \( \vec{y} \) is defined as \( \theta = \arccos \left( \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \| \| \vec{y} \|} \right) \).
\[ \ell_i < \| \vec{V} \| \leq \ell_i / \alpha \]  

(13)

is satisfied, where \( \ell_i \) denotes the yardstick length at pass \( i \) and \( \| \vec{V} \| \) is the norm of the vector \( \vec{V} \). Therefore, the maximum error introduced by the first pass will occur in the case that \( \| \vec{V} \| = \ell_{i-1} = \ell_i / \alpha \).

The right conditions for ensuring the convergence of the successive approximation by a finite set of orientation vectors of decreasing lengths can be derived by evaluating the worst case. Fig. 3(b) illustrates this process. From supposition (2), for each pass \( i \), the maximum error in the orientation is equal to \( \theta_{\text{max}} \), and from supposition (4) the maximum error in the length will occur if the initial yardstick length is set equal to \( \ell_i = \alpha \| \vec{V} \| \). Based on simple algebra, the error introduced after the first approximation has magnitude given by

\[ \| \vec{r}_1 \|^2 = \| \vec{V} \|^2 + \alpha^2 \| \vec{V} \|^2 - 2 \| \vec{V} \| \alpha \| \vec{V} \| \cos(\theta_{\text{max}}) \]  

(14)

as can be seen from Fig. 3(b). Accordingly, after \( n \) passes the magnitude of the residual vector is given by

\[ \| \vec{r}_n \|^2 = \| \vec{r}_{n-1} \|^2 + \alpha^{2n} \| \vec{V} \|^2 - 2 \| \vec{r}_{n-1} \| \alpha^n \| \vec{V} \| \cos(\theta_{\text{max}}) \]  

(15)

where

- \( \| \vec{r}_i \| \) norm of the residual vector \( \vec{r}_i \) at pass \( i \)
- \( \| \vec{V} \| \) norm of the input vector \( \vec{V} \)
- \( \alpha \) approximation scaling factor
- \( \theta_{\text{max}} \) maximum angle between any given vector and its closest available orientation code vector.

Using the initial condition of (14) and the recursive formula in (15), we can compute the residual vector magnitudes after each pass, \( \| \vec{r}_i \|, i = 1, 2, \ldots, n \) for any given pair \((\alpha, \theta_{\text{max}})\). The necessary and sufficient condition for the convergence of the vector successive approximation scheme is

\[ \| \vec{r}_n \| \rightarrow 0 \text{ as } n \rightarrow \infty. \]  

(16)

We assume that convergence is guaranteed when the improvement in the approximation after two subsequent passes is less than a small fraction of the magnitude of the original vector, as follows:

\[ \frac{\Delta}{\| \vec{V} \|} < \epsilon \]  

(17)

where \( \Delta = \| \vec{r}_n \| - \| \vec{r}_{n-1} \| \) and \( \epsilon \) takes an arbitrarily small value, which, in the graphs shown in Fig. 4, is \( 10^{-8} \).

The recursive formula is used to find the value of the convergence scaling factor \( \alpha \) for any \( \theta_{\text{max}} \) in the range \( 0^\circ \leq \theta_{\text{max}} < 90^\circ \) such that the scheme converges for any \( \alpha \geq \alpha_\hat{\alpha} \), where \( 0.5 \leq \alpha < 1.0 \). Fig. 4(a) gives the values of the convergence scaling factor \( \alpha \) for all angles \( \theta_{\text{max}} \) in the range \( 0^\circ \leq \theta_{\text{max}} < 90^\circ \). Fig. 4(b) shows the number of iterations \( \eta \) required for convergence when \( \alpha = \alpha_\hat{\alpha} \) for various values of \( \theta_{\text{max}} \). The results illustrated in Fig. 4 show that, for \( \theta_{\text{max}} \) values of up to \( 82^\circ \), this successive approximation scheme is guaranteed to converge provided that a suitable value of \( \alpha \) is chosen. For \( \theta_{\text{max}} = 0^\circ \), which is equivalent to the scalar case, convergence is guaranteed for \( \alpha = 0.5 \), as was demonstrated in Fig. 2. For this value, the number of iterations is minimum. As \( \theta_{\text{max}} \) increases, so do both \( \alpha \) and

the number of iterations. From a compression point of view, more iterations would require more bits to achieve a certain distortion. Hence, the selected orientation codebook must be such that \( \theta_{\text{max}} \) is as small as possible. Nevertheless, there is a compromise between the value of \( \theta_{\text{max}} \) and the resolution of the orientation codebook, determined by \( \log_2 N \), where \( N \) is the codebook population and \( k \) is the vector dimension. In one extreme, we have \( \theta_{\text{max}} = 0^\circ \) for the scalar case. However, if vectors of higher dimensions are used, there can be gains in bit rate despite the larger values of \( \theta_{\text{max}} \) and number of iterations due to savings from vector over scalar quantizers.

C. Selection of the Orientation Codebook

After determining the necessary conditions for convergence of the successive approximation scheme, we discuss the requirements for selection of a “good” orientation codebook. These requirements are naturally drawn from the suppositions made in the previous section. It is first assumed that orientation code vectors have unit energy. This is trivial, because any codebook can have this property when its code vectors are properly scaled. The second condition, however, is more
restrictive. We need to guarantee that the maximum error in the orientation introduced by approximation at any stage is bounded by a well-defined value $\theta_{\text{max}}$. Moreover, the graph in Fig. 4(b) indicates that $\theta_{\text{max}}$ should be as small as possible, such that the fastest possible convergence to an arbitrary error is achieved.

Therefore, the main requirement in the design of the orientation codebook is to provide a fairly low value of $\theta_{\text{max}}$ regardless of individual vector locations in $k$-dimensional space. This makes uniform codebooks a natural choice for our method. There is no apparent reason for designing the orientation codebook through the training process, since this would result in a codebook with nonregular structure. Hence, some orientation code vectors will be close to each other while others will be separated by a larger angle. As a result, a large $\theta_{\text{max}}$ value should be considered. Thus, it can be argued that there will always be a uniform codebook with the same $\theta_{\text{max}}$ and a smaller number of code vectors.

Lattice codebooks are a good choice for the orientation codebook of the proposed scheme. They can provide a good tradeoff between $\theta_{\text{max}}$ and the codebook population due to their space-packing properties. In addition to their well-known and well-defined structural properties, lattice codebooks offer the advantage of simple and fast encoding algorithms, as described in Section II. In the following section, properties of several known regular lattices are analyzed in order to verify the suitability of a particular lattice codebook to the application under consideration.

D. Lattice Codebooks for Successive Approximation of Vectors

Following the definitions given in Section II, it is obvious that different lattices lead to a different arrangement of points in real space $\mathbb{R}^k$. Nevertheless, they all share some common structural properties that can be exploited in the design of a “good” orientation codebook. In general, the points of a given regular lattice are distributed on the surface of successive, concentric, $k$-dimensional “hyperspheres” centered at the origin, so that all lattice points at the same shell have the same $l_2$-norm. Hence, the $m^{th}$ shell $S_m$ of a given lattice $L_k$ is the set of all $L_k$-points at the same distance from the origin $r(L_k, m)$, as follows:

$$S_m(L_k) : \{ \tilde{y} \in L_k : ||\tilde{y}||_r = r(L_k, m) \} \tag{18}$$

where $||\tilde{y}||_r = \left( \sum_{i=1}^{k} y_i^2 \right)^{1/2}$ is the $l_2$ norm of $\tilde{y}$.

The shells have pyramidal shape for $r = 1$ ($l_1$-norm) and spherical shape for $r = 2$ ($l_2$-norm). The exact number of $L_k$-lattice points at any shell, for the most important regular lattices can be calculated by using the theta functions [17] or the recently developed $N$u functions [26] for spherical and pyramidal shells, respectively.

The following parameters play a key role in the evaluation of the efficiency of a particular lattice orientation codebook:

1) the angle between the nearest neighbor code vectors $\theta_{NN}(L_k, S_m)$;
2) the population of the lattice points on the particular shell $N_m(L_k, S_m)$;
3) the maximum possible angle between any input vector and its closest code vector $\theta_{\text{max}}(L_k, S_m)$ assuming that the orientation codebook is built by the lattice points from shell $S_m$ of a given lattice $L_k$. Note that $\theta_{NN}(L_k, S_m)$ has the same value for any set of $NN$ code vectors at the same shell.

By examining the definition of the kissing number in lattice packing given in Section II, it is easy to see that $\tau_{\text{max}}(L_k)$ determines the maximum number of points that can be located on the surface of a $k$-dimensional sphere in $\mathbb{R}^k$, so that the angle between any nearest neighbor vectors is $\theta_{NN}(L_k, S_1) = 60^\circ$. Fig. 5(a) illustrates this in a 2-D space. The centers of any three identical, nonoverlapping spheres of arbitrary radius $\rho$ in touch with each other form an equilateral triangle, so that $\theta_{NN} = 60^\circ$. Hence, the lattice points on the first spherical shell $S_1(L_k)$ form a codebook with $\theta_{NN}(L_k, S_1) = 60^\circ$. Then, the best lattice packing in $\mathbb{R}^k$ gives the maximum number of points at $\theta_{NN} = 60^\circ$. That is, for $k = 2$, $\tau_{\text{max}}(A_2) = 6$; for $k = 4$, $\tau_{\text{max}}(A_4) = 24$; for $k = 8$, $\tau_{\text{max}}(E_8) = 240$; for $k = 16$, $\tau_{\text{max}}(A_{16}) = 4320$; for $k = 24$, $\tau_{\text{max}}(A_{24}) = 195,560$.

In general, the nearest neighbor angle $\theta_{NN}(L_k, S_m)$ for a given lattice $L_k$ and a spherical shell $S_m$ can be calculated only from the radius of the lattice-packing $\rho(L_k)$ and the energy of the particular shell $r(L_k, S_m)$. Fig. 5(b) shows this process. Assuming that $\tilde{y}_i$, $\tilde{y}_j$ are two neighboring points on shell $S_m(L_k)$ at distance $r(L_k, S_m)$ from the origin, the angular separation between $\tilde{y}_i$ and $\tilde{y}_j$ is calculated as

$$\cos(\theta_{NN}) = 1 - 2 \sin^2(\theta) \Rightarrow \theta_{NN} = \cos^{-1} \left[ 1 - \frac{2\rho^2(L_k)}{r^2(L_k, S_m)} \right]. \tag{19}$$

Table I summarizes the parameters of regular lattices that give the best lattice packing at dimensions $k = 4$, 8, and 16. Note that the nearest neighbor angle $\theta_{NN}(L_k, S_m)$ is calculated using (19), and the maximum possible angle $\theta_{\text{max}}$ between any input vector and its closest code vector has been computed exhaustively. The orientation codebooks in our coding scheme are based on these lattices.

V. A SUCCESSIVE APPROXIMATION VECTOR WAVELET CODER

In this section we introduce a new method for wavelet image coding based on successive approximation vector quantization, described in Section IV. We refer to this method as successive approximation wavelet vector quantization (SA-W-VQ). Since
<table>
<thead>
<tr>
<th>lattice type, $L_k$</th>
<th>shell index, $m$</th>
<th>population, $N_m$</th>
<th>nearest neighbor, $\theta_{NN}$</th>
<th>maximum actual angle, $\theta_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_4$</td>
<td>1</td>
<td>24</td>
<td>60°</td>
<td>45°</td>
</tr>
<tr>
<td>$D_4$</td>
<td>2</td>
<td>24</td>
<td>60°</td>
<td>45°</td>
</tr>
<tr>
<td>$D_4$</td>
<td>1+2</td>
<td>48</td>
<td>45°</td>
<td>32°</td>
</tr>
<tr>
<td>$E_8$</td>
<td>2</td>
<td>2160</td>
<td>41°</td>
<td>45°</td>
</tr>
<tr>
<td>$E_8$</td>
<td>3</td>
<td>6720</td>
<td>38°</td>
<td>35°</td>
</tr>
<tr>
<td>$E_8$</td>
<td>1+2</td>
<td>2400</td>
<td>41°/38°</td>
<td>32°</td>
</tr>
<tr>
<td>$E_8$</td>
<td>1+2+3</td>
<td>9120</td>
<td>30°/33°</td>
<td>29°</td>
</tr>
<tr>
<td>$A_{16}$</td>
<td>2</td>
<td>4320</td>
<td>60°</td>
<td>55°</td>
</tr>
</tbody>
</table>

**TABLE I**

Parameters of the regular lattices with best packing in dimensions $k = 4, 8, 16$

SA-W-VQ uses a successive approximation process, it satisfies requirements (ii) and (iii) discussed in Section III-C. According to requirement (i), a good wavelet coder should exploit the similarities among the bands of the same orientation to generate zero tree roots. The SA-W-VQ coder also makes use of this fact.

In coding of images with SA-W-VQ, first the image mean is extracted. This removes the DC value of the lowest frequency band and hence increases the number of zero-valued coefficients. These zero-valued coefficients can be the roots of zero-trees, which would indicate that all corresponding coefficients in all bands are zero. This tends to increase coding efficiency. An $R$-stage biorthogonal wavelet transform is then applied to the zero-mean image yielding an image decomposition like the one shown in Fig. 1(a). Since the filter banks are not orthogonal—but are biorthogonal—each band has to be normalized so that a quantization error $\epsilon$ in one coefficient will contribute with the same amount of distortion in the final image, no matter which coefficient is considered. This is equivalent to a noise shaping according to a flat HVS response. In a case where a particular HVS response should be used, the scaling of the coefficients is modified according to the desired response [48]. It must be pointed out that this is because the successive approximation method ensures that each coefficient has a guaranteed maximum error.

The image is then divided into $M \times N$ blocks forming vectors of dimension $MN$. Depending on the band considered, scanning of the blocks to form a vector is different. The scanning is vertical in the vertical bands, horizontal in the horizontal bands, and zig-zag in the diagonal bands [49]. The maximum magnitude $V$ of all the vectors is then computed. Initially, the yardstick length $\ell$ is set to $\alpha V$ where the value of $\alpha$ is chosen according to the $\theta_{max}$ value of the selected lattice codebook. All the vectors are scanned, and the ones with magnitude smaller than $\ell$ are set to zero. Each of the remaining vectors is replaced by its closest orientation code vector scaled with magnitude $\ell$. After this pass, the locations of the zero vectors are transmitted. This is done via three symbols: zero (Z); zero tree root (ZT); and coded value (C).

If a vector is zero and all of its corresponding vectors in the higher bands of the same orientation are also zero, this vector is replaced by a ZT, so that it is not necessary to transmit its corresponding vectors. For the lowest frequency band, a ZT implies that the corresponding vectors in all bands are zero. In case a vector is zero but not a ZT, it is marked as Z, and no information can be inferred about its corresponding vectors. On the other hand, a nonzero vector is replaced by a coded value symbol (C).

The string generated by the three symbols ZT, Z, and C is then coded by the arithmetic coder described in [50] with an adaptive model. In the higher frequency bands, since there are no ZT’s, the arithmetic coder uses a model with only two symbols (Z and C). After encoding this string, which indicates the location of the zero vectors, the orientation code vectors of the nonzero vectors (marked as C) are encoded. For this purpose, the model of the arithmetic coder is reinitialized to have as many symbols as the population of the orientation codebook. The yardstick length $\ell$ is then updated through multiplication by $\alpha$. The difference between the original and the nonzero reconstructed vectors is coded using the new yardstick length. The new orientation code vectors are also encoded into the bitstream via the arithmetic coder.

In the next pass the vectors that were previously found to be zero are scanned again. A new string of Z’s, ZT’s, and C’s is encoded into the bitstream. In order to reduce the number of symbols in this string, it is beneficial to obtain as many ZT’s as possible. To achieve this, the vectors that have been found nonzero so far are assumed to be zero during the ZT generation, although they are not encoded as Z. As in the previous pass, the indices of the C vectors are encoded and the whole process is repeated until a certain bit rate is achieved.

The generated bitstream has a header, which informs the decoder about the number of stages of the decomposition, the image dimensions, the format of the image, the image means (luminance and both chrominances), the value of $\alpha$, and the initial value of the yardstick length. In our implementation, 12 bytes are used for monochrome and 14 bytes for color images.

It should be pointed out that there are some similarities between SA-W-VQ and a multistage gain/shape VQ [3]. The gain corresponds to yardstick length and the shape to its orientation. However, SA-W-VQ has the advantage in that there is no need to code the values of the gain because the yardstick length is fixed and known at each pass. Also note that SA-W-VQ has a very simple encoding algorithm due to the fast NN algorithms of the lattice code vectors.

**VI. EXPERIMENTS AND SIMULATION RESULTS**

In this section, the performance of SA-W-VQ with various lattice codebooks is compared against the standard JPEG [51] and some of the most successful image coding techniques based on vector quantization. The wavelet transform used throughout this work is a five-stage octave band decomposition implemented by the filter bank described in Table II, which is shown to give very good subjective performance [52].

The test images are Lena at resolutions $256 \times 256 \times 8$ and $512 \times 512 \times 8$; and the set of ISO/CCITT test images, namely, Barbara, Boats, Girl, Gold, and Zelda, with resolution $720 \times 576 \times 8$. Note that results refer only to the monochrome versions of these images; however, extension to color is straightforward.
A. Comparison Between Different Lattice Codebooks

In the first experiment, we evaluate the performance of different orientation codebooks. These codebooks are built by using the lattices with the the best known space-packing properties in dimensions $k = 4, 8$ and $16$. The parameters of these lattice codebooks are tabulated in Table I. The test image in this experiment is Lena $256 \times 256 \times 8$.

First, the best value of the approximation scaling factor $\alpha$ is estimated for various lattice codebooks. Fig. 6(a)–(c) plot the peak signal-to-noise ratio (PSNR) performance against $\alpha$, obtained by orientation codebooks based on the spherical shells of lattices $D_4$, $E_8$ and $\Lambda_{16}$, assuming 0.5 b/pixel.

For $k = 4$, three different $D_4$ codebooks are compared: (i) shell 1, $N(D_4, S_1) = 24, \theta_{\text{max}} = 45^\circ$; (ii) shell 2, $N(D_4, S_2) = 24, \theta_{\text{max}} = 45^\circ$; and (iii) shell $1 + 2, N(D_4, S_{1+2}) = 48, \theta_{\text{max}} = 32^\circ$. Fig. 6(a) summarizes these results. Note that in all cases, the PSNR curve reaches its peak for a value of $\alpha$, which is well inside the prediction drawn from Fig. 4(a), for the $\theta_{\text{max}}(D_4, S_m)$ values given in Table I. This result is consistent for all the test codebooks. Also, the best performance is achieved by $D_4$-shell 1 codebook. Although $D_4$-shell 1 $+ 2$ has a smaller $\theta_{\text{max}}$ value than shell 1, its coding performance is poorer. This is due to the fact that the codebook population of $D_4$-1 $+ 2$ is double that of $D_4$-1.

For $k = 8$, five different $E_8$ codebooks have been tested. These are: (i) shell 1, $N(E_8, S_1) = 240, \theta_{\text{max}} = 45^\circ$; (ii) shell 2, $N(E_8, S_2) = 2160, \theta_{\text{max}} = 45^\circ$; (iii) shell 3, $N(E_8, S_3) = 6720, \theta_{\text{max}} = 35^\circ$; (iv) shell $1 + 2, N(E_8, S_{1+2}) = 2400, \theta_{\text{max}} = 32^\circ$; and (v) shell $1 + 2 + 3, N(E_8, S_{1+2+3}) = 9120, \theta_{\text{max}} = 29^\circ$. Fig. 6(b) illustrates the PSNR results of the $E_8$ codebooks. The best $\alpha$ for all cases is near 0.53, which corresponds to a maximum angle $\theta_{\text{max}} = 15^\circ$ except for shell 1, where the best $\alpha$ takes a higher value ($\alpha = 0.60$; hence $\theta_{\text{max}} = 35^\circ$).

### Table II: Coefficients of the Used Filter Bank

<table>
<thead>
<tr>
<th>$H_0(z)$</th>
<th>$G_0(z)$</th>
<th>$H_1(z)$</th>
<th>$G_1(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z^1$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^2$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^3$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^4$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^5$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^6$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^7$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>$z^8$</td>
<td>0.0000000</td>
<td>-0.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>

Fig. 6. Performance of SA-W-VQ versus $\alpha$ for Lena $256 \times 256$ image at 0.5 b/pixel and different lattices: (a) $D_4$ lattice, shells 1, 2, and 1&2; (b) $E_8$ lattice, shells 1, 2, 3, 1&2 and 1&2&3; (c) $D_4$, $E_8$ and $\Lambda_{16}$ lattices; (d) rate distortion performance of SA-W-VQ for Lena $256 \times 256$ and with $D_4$, $E_8$ and $\Lambda_{16}$ lattices.
TABLE III

<table>
<thead>
<tr>
<th>Test Image</th>
<th>$D_4$</th>
<th>$E_8$</th>
<th>$A_{16}$</th>
<th>EZW</th>
<th>JPEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARBARA</td>
<td>29.36</td>
<td>30.60</td>
<td>30.90</td>
<td>29.03</td>
<td>27.27</td>
</tr>
<tr>
<td>BOATS</td>
<td>34.19</td>
<td>34.78</td>
<td>35.24</td>
<td>34.29</td>
<td>32.63</td>
</tr>
<tr>
<td>GIRL</td>
<td>35.27</td>
<td>35.91</td>
<td>36.12</td>
<td>35.14</td>
<td>33.98</td>
</tr>
<tr>
<td>GOLD</td>
<td>31.01</td>
<td>32.76</td>
<td>32.61</td>
<td>32.48</td>
<td>31.38</td>
</tr>
<tr>
<td>ZELDA</td>
<td>38.43</td>
<td>39.36</td>
<td>39.44</td>
<td>39.08</td>
<td>37.16</td>
</tr>
<tr>
<td>LENA 256</td>
<td>30.13</td>
<td>30.15</td>
<td>30.29</td>
<td>30.06</td>
<td>28.07</td>
</tr>
<tr>
<td>LENA 512</td>
<td>35.17</td>
<td>35.86</td>
<td>36.09</td>
<td>35.02</td>
<td>33.42</td>
</tr>
</tbody>
</table>

lattice codebooks when the optimum $\alpha$ value is selected. Note that $A_{16}$ codebook always performs better compared to both $E_8$ and $D_4$. In general, although there are no substantial differences in the performance of the three codebooks, higher dimensional codebooks result in better PSNR values, and this is consistent for all bit rates.

B. Rate-Distortion Performance of SA-W-VQ

After selecting the best scaling factor $\alpha$ for the three lattice-based orientation codebooks, the coding performance of the proposed method is compared against the standard JPEG coder [51] and the embedded wavelet image coder proposed by Shapiro [39]. Table III summarizes PSNR results obtained by these methods for coding the test image Lena $256 \times 256 \times 8$ and $512 \times 512 \times 8$ and the set of five ISO/CCITT test images at 0.4 b/pixel. Our simulation results demonstrate that the proposed coding scheme achieves considerably better R-D performance compared with the JPEG coder. The improvement in PSNR over the JPEG-coded images is constantly around 2.50 dB for $A_{16}$ and 1.50 dB for the $D_4$ codebook. This improvement in PSNR is reflected in the picture quality of the coded images, as demonstrated in Fig. 7.

Table III also compares SA-W-VQ with the embedded zero tree wavelet (EZW) image coder proposed by Shapiro [39], which is a very efficient coder employing successive approximation scalar quantization of wavelet coefficients. Fig. 7 shows the test images Lena $256 \times 256$ and Barbara coded at a compression ratio of 20:1 (0.4 b/pixel) using the proposed method with lattice $A_{16}$.

C. Comparison with Other Coders

Finally, we have compared the rate-distortion performance of our coding system with some of the most successful image coding methods reported in the literature. For this experiment, we have used the test image Lena $512 \times 512 \times 8$, since simulation results from other methods are readily available for this image. Table IV summarizes the results of this comparison. The methods that we have included are referred to in the table. Simulation results are presented for different categories of low bit rates. From these results, it can be seen that SA-W-VQ with $A_{16}$ lattice gives the comparable PSNR results to the best methods for all the test bit rates.

VII. CONCLUSIONS

A method of performing successive approximation of wavelet coefficients using vector quantization was introduced. We refer to this method as successive approximation

---

Fig. 7. Test images coded at 0.4 b/pixel using SA-W-VQ with the lattice $A_{16}$: (a) Lena $256 \times 256$; (b) Barbara.

Nevertheless, the best rate-distortion performance for all the bit rates tested was obtained by $E_8$-shell 1 codebook, despite the fact that more passes are typically required to achieve the same distortion (since $\alpha(E_8$-shell 1) > $\alpha(E_8$-others)). Again, this is due to the smaller codebook population in $E_8$-shell 1, at least 1/10 of the population of the other $E_8$ codebooks.

Fig. 6(c) compares PSNR performance against the scaling factor $\alpha$ for the best tested lattice codebooks at $k = 4, 8$, and 16, namely $D_4$-shell 1, $E_8$-shell 1, and $A_{16}$-shell 2. Finally, Fig. 6(d) shows the rate-distortion performance of these three

---

(continued on next page)
wavelet vector quantization (SA-W-VQ). An analysis of the convergence of successive approximation vector quantization (SA-VQ) was made. It was found that the most important feature of a SA-VQ codebook is the maximum possible error in orientation when an input vector is represented by its closest code vector. Orientation codebooks based on regular lattices are suitable for SA-VQ due to their well known structural properties and the existence of fast encoding algorithms.

SA-W-VQ was implemented taking advantage of the similarities among the frequency bands of same spatial orientation. Several lattice codebooks were compared and the first shells of the lattices were shown to perform best. Among these, the Barnes–Wall lattice $A_{16}$ offered the best rate-distortion results.

The results also show that the performance of SA-W-VQ compares favorably with most of the well-established image coding methods reported in the literature, and particularly striking gains were obtained at low bit rates (less than 0.3 b/pixel). Since the successive approximation of vectors guarantees levels of distortion in each pass, noise shaping can be easily implemented by simple weighting of the coefficients prior to coding. Moreover, SA-W-VQ has the advantage of a very simple encoding process due to the fast NN algorithms of the lattice codebooks.

### References


