Additive Vector Decoding of Transform Coded Images

Siu-Wai Wu, *Associate Member, IEEE,* and Allen Gersho, *Fellow, IEEE*

***Abstract*— In a standard transform coding scheme of images or video, the decoder can be implemented by a table-lookup technique without the explicit use of an inverse transformation. In this new decoding method, each received code index of a transform coefficient addresses a particular codebook to fetch a component code vector that resembles the basis vector of the linear transformation. The output image is then reconstructed by summing a small number of nonzero component code vectors. With a set of well-designed codebooks, this new decoder can exploit the correlation among the quantized transform coefficients to achieve better rate-distortion performance than the conventional decoding method. An iterative algorithm for designing a set of locally optimal codebooks from a training set of images is presented. We demonstrate that this new idea can be applied to decode improved quality pictures from the bitstream generated from a standard encoding scheme of still images or video, while the complexity is low enough to justify practical implementation.**

***Index Terms*— Discrete cosine transform, image compression, transform coding, vector quantization.**

I. Introduction

IN THE PAST, most research studies on video compression have focused on the joint design of both the encoder and the decoder with the goal of optimizing the performance of the entire system subject only to a constraint on the available capacity of the channel. However, the expanding usage of video codecs coupled with the establishment of coding standards in recent years makes it less likely that today's system designer would have the freedom to jointly design the encoder and the decoder. From the perspective of decoder design, the performance of the system is constrained by an existing encoder which generates the bitstream and cannot be modified by the designer of the decoder. Therefore, effectively designing the decoder under these constraints is a distinctive and important task worthy of special consideration.

Manuscript received January 21, 1996; revised May 14, 1997. This work was supported by the University of California MICRO Program, Rockwell International Corporation, Hughes Aircraft Company, Eastman Kodak Company, and the Center for Advanced Television Studies. This paper was presented in part at the SPIE Conference on Image Processing Algorithms and Techniques III, San Jose, CA, Feb. 1992. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Michael W. Marcellin.

S.-W. Wu was with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA. He is now with General Instrument Corporation, San Diego, CA 92121 USA (e-mail: swu@gi.com).

A. Gersho is with the Center for Information Processing Research, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: gersho@ece.ucsb.edu).

Publisher Item Identifier S 1057-7149(98)03994-3.

To understand this decoding problem, we shall look at the role of the decoder in a compression system as the interface between the channel and the output video display device [1]. While the encoder *translates* the source video signal into a bitstream and transmits it over a channel, the decoder *interprets* the bitstream. Similar to the interpretation of human language, a particular bitstream could be interpreted by the decoder in a nonunique fashion, but some interpretations will result in perceptually good quality video, while others may not.

Suppose an encoder produces the binary codeword β for a finite duration input signal $\mathbf{x}$. The *optimal decoder* is one that generates the minimum distortion estimate of $\mathbf{x}$ given β. In other words, the objective of the decoder design is to solve the minimization problem

$$\min_{\hat{\mathbf{x}}} E\{d(\mathbf{x}, \hat{\mathbf{x}}) \mid \beta\} \tag{1}$$

where $d(\cdot, \cdot)$ is an appropriate distortion measure, and $\hat{\mathbf{x}}$ is the output signal reconstructed from the codeword β. For the well-known mean squared error distortion measure, the optimal decoder reproduces the video signal $\hat{\mathbf{x}}$ as the conditional expectation $E[\mathbf{x} \mid \beta]$.

Since the set of admissible values of β consists of those attainable permutations of zero's and one's that can be produced by the encoder and transmitted on the channel, the optimal decoder could in principle be implemented by a lookup table addressed by β, or equivalently, by an index corresponding to β. The content of the table is the set of best representative output video signals corresponding to the permissible values of β. Notice that such an optimal decoder is the decoder of a vector quantizer [2] whose codebook contains the representative video signals.

In practice, it is difficult or impossible to implement the optimal decoder because the size of the codebook grows exponentially with the bit rate of the codeword β. Therefore some structures are usually employed at the decoder to synthesize the output video signal from β instead of storing every possible output video signal in memory. These structures impose extra constraints on the attainable set of output video signals, resulting in suboptimal performance in general. An example is the structure consisting of inverse quantization followed by inverse transformation in transform coding.

In this paper, we investigate the decoding problem for transform coding. First, in the following section, we will examine the model of transform coding. Then we will introduce an additive vector decoding structure that is potentially capable of decoding enhanced quality images. An algorithm for designing the codebooks of the additive vector decoder will be developed

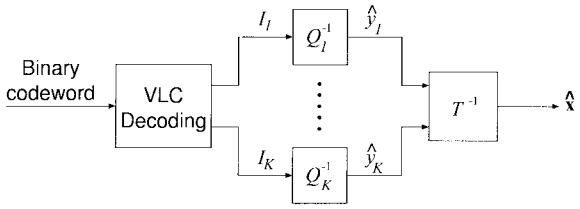

Fig. 1. Model of a conventional transform decoder.

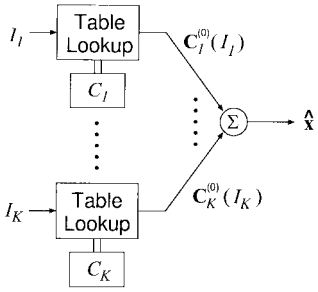

Fig. 2. Additive vector decoder for transform coding.

in Section IV. Finally, we apply the additive vector decoding scheme to an industrial coding standard based on the discrete cosine transform (DCT). Simulation results and analysis of the implementation complexity will be presented in Section V.

II. Transform Coding

Transform coding has recently become the heart of several industry standards for image and video compression. In particular, the DCT is adopted in the Joint Photographers Expert Group (JPEG) image coding standard, the Motion Picture Expert Group (MPEG) video coding standard, the ITU-T (formerly CCITT) H.261 and H.263 recommendations for visual telephony, and many other commercially available compression systems based on some variations of these standard transform coding schemes. The purpose of this work is to investigate a vector decoding technique for reconstructing enhanced quality pictures from the compressed bitstream produced by a standard transform encoder. This enhanced decoding technique inherits the table-lookup philosophy of the optimal block decoding scheme while employing a summation structure to ease the codebook storage complexity. Details of this enhanced decoding technique are given in the following sections. First of all, let us examine the model of conventional transform coding.

Transform coding is a block-based image compression technique in which the input image is partitioned into fixed-size small blocks and each block of pixels is quantized independently. In a typical transform encoder, an input image block, $\mathbf{x}$, is transformed by a linear operation T at the encoder into a set of transform coefficients. Each transform coefficient, y_k $(k = 1, \cdots, K)$, is encoded independently by a distinct scalar quantizer to generate an index I_k. The indices are then losslessly entropy encoded to generate a binary string of bits that are transmitted to the decoder. In this paper, we shall assume that the channel is error-free so that the decoder receives the same string of bits produced by the encoder.

Fig. 1 shows the model of a conventional decoder of transform coding. For each image block encoded by the encoder, an ordered set of indices $[I_1, \cdots, I_K]$ are extracted from the corresponding binary codeword received by the decoder, and each index I_k in this set is independently decoded into a quantized transform coefficient $\hat{y}_k$. The set of quantized transform coefficients are then converted into the output image block $\hat{\mathbf{x}}$ by the inverse transformation T^{-1}. The image is reconstructed by tiling the output image blocks on the same grid as the block partitioning at the encoder.

To achieve a high compression ratio, most of the transform coefficients are coarsely quantized at the encoder. Coarse quantization of the transform coefficients results in various artifacts in the decoded image, including loss of detail, introduction of noise, and blockiness. Methods for reducing the block-artifacts in transform coding of images have been extensively studied [3]–[5]. However, these enhancement techniques proposed for transform decoders mitigate the block-artifacts only in smooth areas of the image, and are based on the conventional transform decoding model. We have not found any literature prior to our work on changing the basic structure of the decoder to minimize the amount of quantization noise in the decoded image.

III. Vector Decoding with a Summation Structure

From the theory of transform coding, we know that the image block reconstructed by a conventional transform decoder can be formulated as the weighted sum of the basis vectors of the transformation (see for example [6]), which is

$$\begin{aligned}\hat{\mathbf{x}} &= \sum_{k=1}^{K} \hat{y}_k \mathbf{T}_k \\ &= \sum_{k=1}^{K} q_k(I_k)\mathbf{T}_k \end{aligned} \tag{2}$$

where $\{\mathbf{T}_1, \cdots, \mathbf{T}_K\}$ are the basis vectors of the transformation, and $\{q_k(i) : i = 1, \cdots, M_k\}$ denotes the set of possible quantization levels of the quantizer Q_k. Notice that each term in (2) is a vector valued function of a discrete variable I_k. Therefore, $\hat{\mathbf{x}}$ can be generated as a sum of *code vectors* by the construction

$$\hat{\mathbf{x}} = \sum_{k=1}^{K} \mathbf{C}_k^{(0)}(I_k) \tag{3}$$

where

$$\mathbf{C}_k^{(0)}(I_k) = q_k(I_k)\mathbf{T}_k. \tag{4}$$

This formulation of transform coding belongs to the class of *summation product codes* in the literature of vector quantization [7], [8]. Instead of using an inverse transformation with scalar decoding, the conventional transform decoder can be equivalently implemented by a bank of K lookup tables and a summation as shown in the schematic of Fig. 2, where each index I_k of an image block addresses a particular table (*codebook*) C_k to fetch the code vector $\mathbf{C}_k^{(0)}(I_k)$ to be summed.

Compared to the conventional decoding structure that decodes the quantized transform coefficients and performs inverse transformation, the table-lookup structure of Fig. 2 is a one-step process that skips the transform domain and generates the spatial domain components of the output image block directly from the indices, hence allows more freedom in the design of the decoder. The equivalent conventional decoder formulated above is only a special case in which a set of trivial codebooks defined by (4) are employed. In general, we may use an arbitrary set of codebooks. Of course, if we pick the codebooks randomly, there is no guarantee on the performance of the decoder, and it is likely that the resulting decoder will produce random noise rather than a good quality reproduction of the original image. However, with a set of well-designed codebooks that fit the statistics of the input images, such an additive vector decoding structure is potentially capable of decoding images with lower distortion than that decoded by the inverse transformation method. Conceptually, the inverse transform in the conventional decoder is a linear operation on the quantized transform coefficients, whereas the vector decoder with a set of nontrivial codebooks is a nonlinear operation in which the output image block is nonlinearly "interpolated" from a small number of nonzero quantized transform coefficients. For this reason, such an enhanced decoder is sometimes referred as a *nonlinear interpolative* decoder [9], [10]. Combined decoding and estimation has been studied more extensively in the context of *generalized vector quantization* [11]. An overlapped block decoding technique [12] can be easily applied to this vector decoding scheme to improve its performance by exploiting the statistical dependency among the indices in neighboring blocks. A schematic of this lapped vector decoder is shown in Fig. 3. The first part of this lapped vector decoder is the same as the summation vector decoding scheme described above, except that the code vectors $\mathbf{C}_k(I_k)$ have higher dimension than the basis vectors of the transformation so that the output block

$$\hat{\mathbf{x}} = \sum_{k=1}^{K} \mathbf{C}_k(I_k) \tag{5}$$

covers an area that extends beyond the input image block $\mathbf{x}$ into its neighborhood. The image is reconstructed by overlapping the buffered output blocks and summing the corresponding elements of the blocks in the overlapped regions. For the rest of this paper, we will consider this more general summation vector decoding structure with overlapping blocks. An algorithm for designing a set of locally optimal codebooks from a set of training images will be developed in the following section.

IV. Codebook Design

Given a fixed transform encoder, we want to design a set of codebooks $\{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$ that optimize the performance of the vector decoder of Fig. 3, where $\mathcal{C}_k = \{\mathbf{C}_k(i) : i = 1, \cdots, M_k\}$. For mathematical tractability, we use the mean squared error of the block of pixels, defined as

$$d = E[\|\mathbf{x} - \hat{\mathbf{x}}\|^2] \tag{6}$$

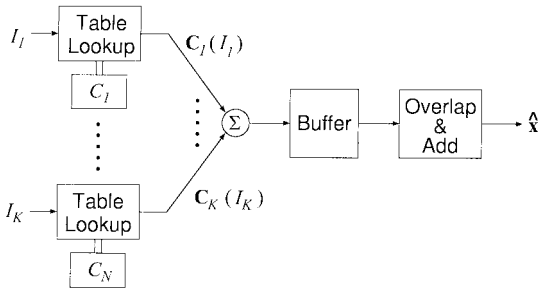

Fig. 3. Additive vector decoder with block overlapping for transform coding.

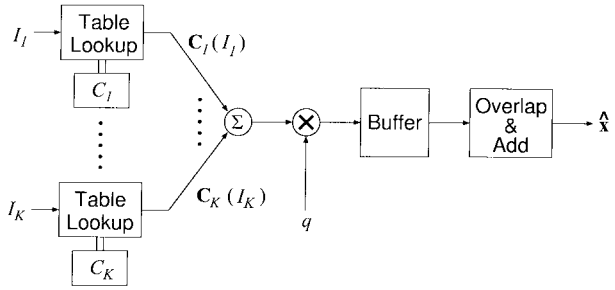

Fig. 4. Additive vector decoder with block overlapping and quantizer scaling for DCT coding.

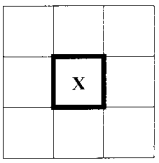

Fig. 5. Example of a neighborhood of input vector $\mathbf{x}$.

to measure the distortion of the decoded images. Although this distortion measure may not exactly match a model of human perception, it is closely related to the general perceptual quality of the coded image, and therefore can be used to guide the codebook design algorithm. This is verified by the experimental results presented in a later section.

Let us define the *neighborhood* $\mathcal{N}_{\mathbf{x}}$ of an input block $\mathbf{x}$ as a particular subset of blocks associated with $\mathbf{x}$ in the input image. An example is shown in Fig. 5 where the neighborhood of an input block $\mathbf{x}$ is the set consisting of the block $\mathbf{x}$ and all the eight blocks adjacent to $\mathbf{x}$ in the input image. Notice that neighborhoods are symmetric in general, i.e., a block $\mathbf{x}_1$ is in $\mathcal{N}_{\mathbf{x}_2}$ if and only if $\mathbf{x}_2$ is in $\mathcal{N}_{\mathbf{x}_1}$. This symmetry property is assumed throughout this paper. We also assume that the neighborhood of a block $\mathbf{x}$ does not vary with the location of $\mathbf{x}$ in the image, therefore a spatially invariant set of ordered pairs $\Phi = \{(m,n) : \mathbf{x}_{(m,n)} \in \mathcal{N}_{\mathbf{x}}\}$ can be defined, where $\mathbf{x}_{(m,n)}$ denote the input vector m blocks to the right and n blocks below the current input vector $\mathbf{x}$. (Note that $\mathbf{x}_{(0,0)} \equiv \mathbf{x}$.)

For each code vector $\mathbf{C}_k(i)$, a set of *partial code vectors* $\{\mathbf{C}_{k,(m,n)}(i) : (m,n) \in \Phi\}$ can be defined such that each partial code vector has the same dimension as the input vector, and the elements of $\mathbf{C}_{k,(m,n)}(i)$ are the contributions of $\mathbf{C}_k(i)$ to the reconstruction of the block of pixels at coordinate $(-m,-n)$ relative to the current input vector from which

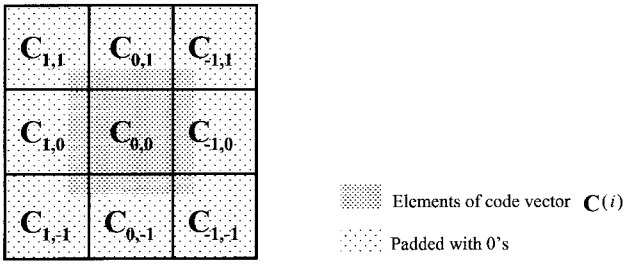

Fig. 6. Construction of partial code vectors for the nine-block neighborhood of Fig. 5. (The index i is omitted in the figure for simplicity.)

the index i is generated. An example of the construction of partial code vectors is illustrated in Fig. 6 for the nine-block neighborhood of Fig. 5. In this diagram, each block in the figure represents one partial code vector $\mathbf{C}_{k,(m,n)}(i)$ where $m,n \in \{-1,0,1\}$. The heavily shaded region represents elements of $\mathbf{C}_{k,(m,n)}(i)$ inherited from $\mathbf{C}_k(i)$, whereas the lightly shaded region represents elements of $\mathbf{C}_{k,(m,n)}$ that are defined as zero.

With the partial code vectors so defined, a block of pixels in the output image at the same spatial coordinates as the original block $\mathbf{x}$ in the input image is reconstructed as

$$\hat{\mathbf{x}} = \sum_{k=1}^{K} \sum_{(m,n)\in\Phi} \mathbf{C}_{k,(m,n)}(I_k(\mathbf{x}_{(m,n)})) \tag{7}$$

where $\mathbf{x}_{(m,n)}$ is the block at coordinate (m,n) relative to $\mathbf{x}$ in the input image, and $I_k(\mathbf{x}_{(m,n)})$ is the index encoded from the k-th transform coefficient of $\mathbf{x}_{(m,n)}$. Thus designing the codebooks $\{\mathcal{C}_1,\cdots,\mathcal{C}_K\}$ is equivalent to designing a set of partial codebooks $\{\mathcal{C}_{k,(m,n)} : k = 1,\cdots,K; (m,n) \in \Phi\}$ where $\mathcal{C}_{k,(m,n)} = \{\mathbf{C}_{k,(m,n)}(i) : i = 1,\cdots M_k\}$. Notice that a partial codebook can be considered as a mapping from a finite set of integers to the K-dimensional vector space. The codebook design problem is to find the mappings $\{\tilde{\mathbf{C}}_{(m,n)}(\cdot) : (m,n) \in \Phi\}$ to minimize the distortion

$$d = E\left[\left\|\mathbf{x} - \sum_{(m,n)\in\Phi} \tilde{\mathbf{C}}_{(m,n)}(I(\mathbf{x}_{(m,n)}))\right\|^2\right] \tag{8}$$

subject to the structural constraint on the partial code vectors.

For each $k \in \{1,\cdots,K\}$ and for each $(m,n) \in \Phi$, let us define

$$\mathbf{g}_{k,(m,n)}(\mathcal{N}_{\mathbf{x}}) = \mathbf{x} - \sum_{\acute{k}=1}^{K} \sum_{\substack{(\acute{m},\acute{n})\in\Phi \\ (\acute{k},\acute{m},\acute{n})\neq(k,m,n)}} \mathbf{C}_{\acute{k},(\acute{m},\acute{n})}(I_{\acute{k}}(\mathbf{x}_{(\acute{m},\acute{n})})), \tag{9}$$

and write the distortion as

$$d = E[\|\mathbf{g}_{k,(m,n)}(\mathcal{N}_{\mathbf{x}}) - \mathbf{C}_{(m,n)}(I_k(\mathbf{x}_{(m,n)}))\|^2]. \tag{10}$$

Then the following theorem can be readily derived from basic estimation theory [13].

Theorem 1—Necessary Condition for Optimality: For every $k \in \{1,\cdots,K\}$ and for every $(m,n) \in \Phi$, the optimal partial code vectors of the codebook $\mathcal{C}_{k,(m,n)}$ are

$$\mathbf{C}^*_{k,(m,n)}(i) = \Upsilon_{(m,n)} E[\mathbf{g}_{k,(m,n)}(\mathcal{N}_{\mathbf{x}}) \,|\, I_k(\mathbf{x}_{(m,n)}) = i] \quad \text{for } i = 1,\cdots,M_k \tag{11}$$

where $\Upsilon_{(m,n)}$ is a masking operator that sets selected elements of the vector to zero according to the structural constraint on the partial code vector $\mathcal{C}_{k,(m,n)}(i)$.

This theorem basically means that the best partial code vector to use at the decoder is the conditional expectation of $\mathbf{g}_{k,(m,n)}$ given that the kth index received for the block is i. Notice that $\mathbf{g}_{k,(m,n)}$ is a function of the input image block $\mathbf{x}$'s neighborhood. Therefore the conditional expectation in (11) can be easily estimated from a large training set of representative images by computing the ensemble average [14], [15] as follows:

$$\tilde{\mathbf{C}}^*_{k,(m,n)}(i) = \frac{1}{|\mathcal{R}_{k,(m,n)}(i)|} \sum_{\mathbf{v}\in\mathcal{R}_{k,(m,n)}(i)} \Upsilon_{(m,n)} \mathbf{g}_{k,(m,n)}(\mathcal{N}_{\mathbf{v}}) \tag{12}$$

where $\mathcal{V} = \{\mathbf{v}\}$ is the set of input blocks extracted from the training images; $\mathcal{R}_{k,(m,n)}(i) = \{\mathbf{v} : \mathbf{v} \in \mathcal{V}, I_k(\mathbf{v}_{(m,n)}) = i\}$; and $|\cdot|$ denotes the number of vectors in the set.

Hence, an iterative codebook design algorithm can be developed from the above theorem: Starting from any initial configuration, we can iteratively apply (12) to improve one partial codebook at a time, keeping all other codebooks fixed. The distortion d can also be estimated from the training images by computing the ensemble average

$$d = \sum_{v\in\mathcal{V}} \|\mathbf{v} - \hat{\mathbf{v}}\|^2 \tag{13}$$

where $\hat{\mathbf{v}}$ denotes the reconstructed version of the training image block $\mathbf{v}$. This distortion d decreases monotonically with successive iterations. Since the distortion is bounded from below and decreases monotonically, it converges to a local minimum. A distortion threshold (ϵ) is used as a convergence criterion similar to the generalized Lloyd algorithm (GLA) for vector quantizer codebook design [15]. We stop the iterations when the percentage decrease in distortion is less than a preset threshold. A pseudocode description of the algorithm for the neighborhood definition of Fig. 5 is given in the Appendix.

The training images used to design the codebooks crucially affects the performance of the decoder. In general, a large training set with a large variety of images assures robust performance for images outside the training set. However, if it is know a prior that the decoder is used only to decode a particular class of images, e.g., head-and-shoulder images, the performance of the decoder can be boosted by using only training images that belongs to the class.

V. Application to DCT Coding with a Quantizer Scale

In most existing transform coding schemes for compression of images and video, the DCT is employed. The two-dimensional (2-D) DCT of a $P \times P$ block of pixels $\{x(i,j) :$

$i,j = 0,\cdots,P-1\}$ can be defined as

$$y(u,v) = \frac{1}{4w_i w_j}\sum_{i=0}^{P-1}\sum_{j=0}^{P-1} x(i,j)\cos\frac{(2i+1)u\pi}{2P} \times \cos\frac{(2j+1)v\pi}{2P} \tag{14}$$

for $u,v = 0,\cdots,P-1$, where

$$w_k = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k=0 \\ 0, & \text{otherwise} \end{cases}$$

and the inverse DCT is defined as

$$\hat{x}(i,j) = \frac{1}{4}\sum_{u=0}^{P-1}\sum_{v=0}^{P-1} \hat{y}(u,v)\cos\frac{(2i+1)u\pi}{2P} \times \cos\frac{(2j+1)v\pi}{2P} \tag{15}$$

for $i,j = 0,\cdots,P-1$.

Typically, a zig-zag scanning scheme is applied to order the elements of the 2-D array of DCT coefficients into a one-dimensional (1-D) sequence $\{y_0,\cdots,y_{K-1}\}$ where $K = P^2$. The DC coefficient y_0 is coded independently with sufficiently high resolution while the AC coefficients $\{y_1,\cdots,y_K\}$ are quantized with uniform step sizes defined by a fixed weighting table $w_k : k = 1,\cdots,K$ and a quantizer scale q such that the step size for the kth transform coefficient is

$$g_k = qw_k. \tag{16}$$

The weighting table determines the relative importances of the DCT coefficients, while the quantizer scale is used to adjust the compression ratio. For this class of DCT coding schemes, an input image block can be modeled as the vector sum of a DC component and an AC component, which are coded separately. Thus we may assume that the transform coding is applied only on the AC component, so that the input to the transform coder, $\mathbf{x}$, is the mean-removed image block. The separately quantized DC component (mean) is added to the decoded AC component at the output of the transform decoder.

A prominent example of this class of transform coding schemes is the JPEG baseline coding algorithm, in which a quantization table completely specifies the step sizes of the quantizers. The number of bits generated by the JPEG algorithm for a particular image is determined by the image, the quantization table, and the Huffman tables. However, the currently favored strategy for adjusting the compression ratio of a JPEG coder is to scale all the entries in the quantization table by the same factor so that the relative weightings of the transform coefficients are constant while the step sizes of the quantizers vary [16]. Another example of DCT coding is the MPEG intraframe coding scheme where the quantizer scale may vary from one block to another to control the number of bits generated from each block.

With the quantizer step sizes defined by (16), the quantized transform coefficients are determined by

$$\hat{y}_k = qw_k\, Round\left[\frac{y_k}{qw_k}\right]. \tag{17}$$

TABLE I
WEIGHTING MATRIX USED IN THE EXPERIMENTS. THE AC ENTRIES ARE TWO TIMES THE QUANTIZATION STEP SIZES GIVEN IN [17]

16	22	20	32	48	80	102	122
24	24	28	38	52	116	120	110
28	26	32	48	80	114	138	112
28	34	44	58	102	174	160	124
36	44	74	112	136	218	206	154
48	70	110	128	162	208	226	184
98	128	156	174	206	242	240	202
144	184	190	196	224	200	206	198

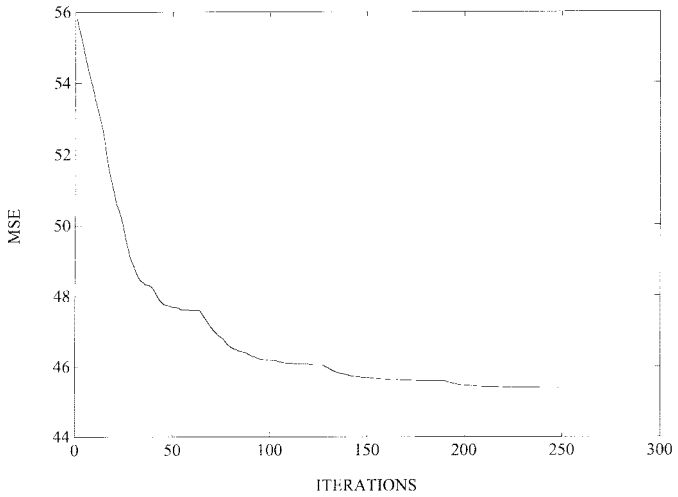

Fig. 7. Changes in average distortion during the codebook design procedure.

Since the DCT is a linear process, the AC component of the output image block can therefore be written as

$$\hat{\mathbf{x}} = q\mathbf{Q}\left(\frac{1}{q}\mathbf{x}\right) \tag{18}$$

where $\mathbf{Q}$ denotes the operation of DCT coding with quantizer scale equal to one, and $\mathbf{x}$ denotes the AC component of the input image block. This equation shows that varying the quantizer scale is equivalent to preprocessing the input image block to change the contrast (AC amplitude) of the pixel variations by a factor of q before applying the DCT coding with unit quantizer scale. The contrast of the image block is restored at the decoder by multiplying the scaling factor q to the decoded AC component. Hence, the lapped additive vector decoding structure discussed above can be applied to this class of DCT coding scheme by incorporating a scaling factor as shown in Fig. 4. Here, the output block is reconstructed as

$$\tilde{\mathbf{x}} = q\sum_{k=1}^{K}\mathbf{C}_k(I_k) \tag{19}$$

where $\{I_1,\cdots,I_K\}$ are the AC indices of the block.

Ideally, for each quantizer scale value, a distinct codebook that is tailored to the statistics of the images coded at that quantizer scale value should be used. In practice, however, if the codebook is designed from training images that contain a large enough variety of scenes, a single codebook can be used for a wide range of quantizer scales values above one. This is

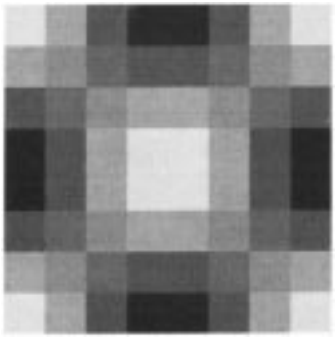

Fig. 8. DCT basis vector (2, 2). Elements of the vector are normalized and offset by 128.

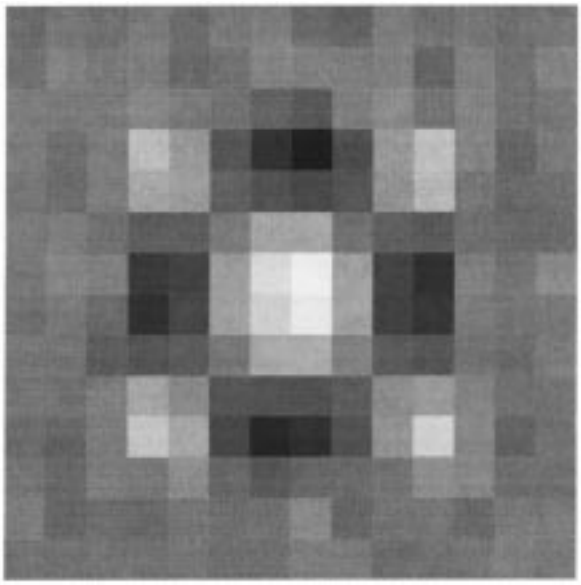

Fig. 9. Code vector in codebook (2, 2). Elements of the vector are normalized and offset by 128.

because training vectors derived from natural images have a wide range of AC magnitude, therefore the same training set can represent the statistics of a wide range of quantizer scale values. We will demonstrate, with the following computer simulation experiments, that the codebooks designed for unit quantizer scale can also operate with a range of quantizer scales, for most practical purpose.

A. Results

Computer simulations were conducted to study the vector decoding method for images compressed with 8 × 8 DCT at low to medium bit rate (0.25 to 0.5 b/pixel) where objectionable distortion can be easily found in the pictures conventionally decoded with inverse DCT. In the experiments, we use the JPEG encoder with the weighting table shown in Table I, which is based on the relative weighting of the DCT coefficients in the quantization table suggested by JPEG for luminance images. The actual quantization table for each experiment is obtained by scaling this weighting matrix by the quantizer scale q.

Fig. 10. Original boat image.

A set of codebooks with code vector dimension equal to 14 × 14 (three-pixel extension on each side of an 8 × 8 block) were designed using the iterative algorithm given in the Appendix. The training set was composed of 20 luminance images containing a large variety of scenes. Fig. 7 shows the change in distortion during the codebook design procedure in which the codebooks were refined in cycles of 63 iterations along the zig-zag scan. In each iteration, the codebook corresponding to one DCT coefficient was updated. The mean squared error (MSE) converges to 81% of the initial value in four cycles $(\epsilon = 0.005)$, corresponding to a 0.9 dB gain in SNR. Figs. 8 and 9 display, respectively, the DCT basis vector (2, 2) and a code vector in the corresponding codebook designed by the iterative algorithm. The elements in these displayed vectors are normalized and offset by a constant gray level (128). Notice the noise like high frequency "correction" components in the code vector of Fig. 9.

The vector decoder is tested with images Lenna and boat, which are outside the training set. Table II lists the peak signal-to-noise ratio (PSNR) values of the images decoded respectively by the conventional decoder (IDCT) and the vector decoder for quantizer scales ranging from one to three. The bit rates in b/pixel are also listed in the table for reference.

Figs. 10–12 show a selected portion of the original boat image, boat decoded by the conventional decoder, and boat decoded by the vector decoder, respectively (both zoomed two times by pixel repetition). The images are coded with unit quantizer scale. The corresponding results for Lenna with quantizer scale equal to 1.5 are shown in Figs. 13–15.

The images reconstructed by the vector decoder not only have higher SNR values, but also have better perceptual quality compared with those reconstructed by the conventional decoder. We can see that there is less blockiness, or staircase effect, and less mosquito noise along the edges. These improvements resulted from exploiting both the intrablock

Fig. 11. Boat image decoded by conventional decoder (0.35 b/pixel).

Fig. 12. Boat image decoded by vector decoder (0.35 b/pixel).

TABLE II
COMPUTATIONAL COMPLEXITIES FOR DECODING BOAT AND LENNA. THE MULTIPLICATIONS ARE NOT NEEDED FOR VECTOR DECODING IF THE CODEBOOKS ARE PRESCALED

Image	Quantizer scale	Bit rate (bpp)	PSNR (dB)		
			Inverse DCT	Vector Decoding	Gain
Lenna	1.0	0.42	33.86	34.41	0.55
	1.5	0.34	32.64	33.20	0.56
	2.0	0.29	31.73	32.28	0.55
	3.0	0.24	30.43	30.94	0.51
Boat	1.0	0.52	32.28	32.91	0.63
	1.5	0.41	30.87	31.44	0.57
	2.0	0.35	29.92	30.44	0.52
	3.0	0.28	28.56	29.02	0.46

and interblock correlations among the quantized transform coefficients.

Fig. 13. Original Lenna image.

Fig. 14. Lenna image decoded by conventional decoder (0.34 b/pixel).

B. Computational Complexity

There are two phases in the conventional decoding method: i) regeneration of the quantized DCT coefficients and ii) computation of the inverse DCT. While one multiplication is required to regenerate each nonzero quantized transform coefficient from its code index, the inverse DCT of an 8×8 block can be computed in 464 additions and 80 multiplications with existing fast DCT algorithms [16]. Suppose an 8×8 image block is coded as r nonzero quantized DCT coefficients. Ignoring the complexity of comparison and decision operations, the conventional decoding scheme will require a total of 464 additions and $(80+r)$ multiplications for decoding the block from the code indices.

TABLE III
COMPUTATIONAL COMPLEXITIES FOR DECODING BOAT AND LENNA. THE MULTIPLICATIONS ARE NOT NEEDED FOR VECTOR DECODING IF THE CODEBOOKS ARE PRESCALED

Quantizer scale	Average complexity per 8×8 block of conventional decoder		Average complexity per 8×8 block of vector decoder	
	Lenna	*Boat*	*Lenna*	*Boat*
1.0	464 add, 84.6 mul	464 add, 86.7 mul	902 add	1324 add
1.5	464 add, 83.8 mul	464 add, 85.4 mul	746 add, (64 mul)	1050 add, (64 mul)
2.0	464 add, 83.3 mul	464 add, 84.5 mul	654 add, (64 mul)	880 add, (64 mul)
3.0	464 add, 82.8 mul	464 add, 83.4 mul	550 add, (64 mul)	676 add, (64 mul)

Fig. 15. Lenna image decoded by vector decoder (0.34 b/pixel).

On the other hand, a straightforward implementation of the additive vector decoder would require $(K-1)$ vector additions. For a fixed quantizer scale, the code vectors can be premultiplied by the quantizer scale and multiplication is not required. If the quantizer scale is variable, then each pixel is multiplied by the quantizer scale after summing the code vectors. Thus, KB additions and B multiplications are required to decode each image block by the additive vector decoder, where B is the number of pixels in a code vector $(B \geq 64)$. For the experiments described above, there would be 12 348 additions and 196 multiplications per 8×8 block. However, only one multiplication is required for each pixel in the reconstructed image if the quantizer scale is the same for every block in the image. This results in only 64 multiplications instead of 196 multiplications per 8×8 block.

A simple heuristic can be incorporated in the design of the codebooks to allow the images to be reconstructed by the additive vector decoder with fewer additions. For each DCT coefficient there is a quantizer index that represents the amplitude level zero. We call the corresponding code vectors *zero-code vectors*. Instead of using the conditional expectation in (11) to update the zero-code vectors, we assign the null vector to them. In other words, the elements of the zero-code vectors are defined as zero. With this constraint on the zero-code vectors, the average distortion still decreases monotonically in the codebook design algorithm since the remaining code vectors are still being optimally selected at each iteration. Moreover, this constraint does not significantly deteriorate the performance of the vector decoder since the zero-code vectors usually bear very little information.

With this modification, the zero-code vectors can be excluded from the summation in (19), and the additive vector decoder sums up only the nonzero code vectors. As a result, the number of additions are reduced to only rB additions for each 8×8 image block. Since a large fraction of the DCT coefficients are quantized to zero, r is usually a small number. Table III shows the average computational complexities per 8×8 block for decoding the test images, Lenna and boat, by the additive vector decoder in the experiments of Section V-A. In the calculations of the complexities, we did not count the blocks that do not contain any nonzero quantized AC coefficients, since these DC blocks can be trivially decoded. Comparing with the inverse DCT decoder, the vector decoder could decode an image with fewer multiplications but slightly more additions.

The numbers of additions in Table III are based on code vector dimension 14×14 $(B = 196)$. If the rate-distortion performance advantage of using overlapping output blocks is sacrificed by decreasing the dimension of the code vectors, fewer additions will be required. In the case of nonoverlapping output blocks, $B = 8 \times 8 = 64$, the number of additions can be reduced to 33% of those shown in Table III. Then the overall computational complexity will be less than that of the inverse DCT decoder, while the average rate-distortion performance of the vector decoder will still be at least as good as the inverse DCT decoder. Moreover, notice that as the bit rate decreases, there are fewer nonzero quantized DCT coefficients, hence fewer additions are required.

C. Codebook Storage

In the vector decoder of Fig. 3, there are a total of $M = \sum_{k=1}^{K} M_k$ possible code vectors, where M_k is the range of the index I_k. Assuming that the input image has a resolution of 8 b/pixel, it can be easily derived from the definition of the DCT in (14), that the number of different quantization levels

for the DCT coefficient y_{uv} of an 8×8 block is

$$M_{uv} = 2 \times \mathit{Round}\left\{ \frac{255 w_u w_v}{4 g_{uv}} \sum_{m=0}^{7} \sum_{n=0}^{7} \mathit{step}\left[\cos \frac{(2m+1)u\pi}{16} \times \cos \frac{(2n+1)v\pi}{16} \right] \right\} + 1 \tag{20}$$

where $\mathit{step}(x) = x$ for $x \geq 0$, $\mathit{step}(x) = 0$ for $x < 0$, and q_{uv} is the quantizer step size for y_{uv}. The constants w_u, w_v are defined in (14). For the weighting matrix given in Table I, there are a total of 1733 possible code vectors.

However, it is not necessary to store every possible code vector. Since the probability distribution of the DCT coefficients are biased, a large percentage of the code vectors have practically zero probability of occurrence. We noticed that more than 60% of the possible code vectors are never addressed by the training images during the codebook design procedure. These "zero-probability" code vectors can be omitted from the codebook storage without deteriorating the performance of the decoder. Furthermore, it is trivial to see that the decoder does not need to store the zero-code vectors, leading to a further reduction in codebook storage. Based on these observations, we found that only 627 code vectors needed to be stored. Each code vector has $14 \times 14 = 196$ elements, and a 12-b scaled integer was used to represent each element. Hence, the total amount of memory required for codebook storage is 184 Kbytes, which can be implemented easily in either software or hardware.

In the experiments of Section V-A with the test images Lenna and boat, the zero-probability code vectors were never addressed. For some rare images, however, the encoder may encounter input image blocks that do not conform to the statistics of the training images, and generates indices that address the zero-probability code vectors. In these cases, we can generate on-line the scaled DCT basis vectors defined in (4) to use as the code vectors.

VI. Conclusion

We have investigated a vector decoding technique with a summation structure for reconstructing improved quality pictures from the bitstreams generated by a standard transform encoder. A set of well-designed codebooks is crucial for good performance of the vector decoder. An iterative algorithm is developed for designing a set of locally optimal codebooks from a training set of representative images. With the codebooks designed by this algorithm, computer simulations demonstrated that the vector decoder reconstructs low bit rate coded images with not only higher SNR, but also better perceptual quality. In particular, there is less blockiness and less mosquito noise along the edges in the decoded pictures. We have also shown that the vector decoder can be implemented with reasonably low complexity of computation and of codebook storage; hence, it can be beneficially included in a practical system that uses standard transform coding for image or video compression.

Appendix
Summation Vector Decoder Codebook Design Algorithm

1) Encode the training images to generate the indices.
2) Generate a set of initial codebooks:

For $k = 1, \cdots, K$

For $i = 1, \cdots, M_k$

$$\tilde{\mathbf{C}}_{k,(m,n)}(i) = \begin{cases} q_k(i)\mathbf{T}_k, & \text{for } m,n = (0,0) \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Calculate

$$d_o = d = \frac{1}{|\mathcal{V}|} \times \sum_{\mathbf{v} \in \mathcal{V}} \left\| \mathbf{v} - \sum_{k=1}^{K} \sum_{m=-1}^{m=1} \sum_{n=-1}^{n=1} \tilde{\mathbf{C}}_{k,(m,n)}(I_k(\mathbf{v}_{(m,n)})) \right\|^2$$

$$\Delta d = 0.$$

3) While $(\frac{\Delta d}{d} > \epsilon)$

For $k = 1, \cdots, K$

For $m = -1, 0, 1;\ n = -1, 0, 1$

For $i = 1, \cdots, M_k$

if $|\mathcal{R}_{k,(m,n)}(i)| \neq 0$ then do

$$\mathbf{C}_{k,(m,n)}(i) = \frac{1}{|\mathcal{R}_{k,(m,n)}(i)|} \sum_{\mathbf{v} \in \mathcal{R}_{k,(m,n)}(i)} \Upsilon_{(m,n)} \times \left\{ \mathbf{v} - \sum_{\acute{k}=1}^{K} \sum_{\substack{\acute{m}=-1 \\ (\acute{k},\acute{m},\acute{n}) \neq (k,m,n)}}^{\acute{m}=1} \sum_{\acute{n}=-1}^{\acute{n}=1} \mathbf{C}_{\acute{k},(\acute{m},\acute{n})}(I_{\acute{k}}(\mathbf{v}_{(\acute{m},\acute{n})})) \right\}$$

$$d = \frac{1}{|\mathcal{V}|} \sum_{\mathbf{v} \in \mathcal{V}} \left\| \mathbf{v} - \sum_{k=1}^{K} \sum_{m=-1}^{m=1} \sum_{n=-1}^{n=1} \mathbf{C}_{k,(m,n)}(I_k(\mathbf{v}_{(m,n)})) \right\|^2$$

$$\Delta d = d_o - d$$

$$d_o = d$$

Endwhile.

References

[1] S.-W. Wu and A. Gersho, "Enhanced video compression with constraints on the bit stream syntax," in *Proc. ICASSP*, Apr. 1993, pp. 488–499.

[2] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 157–166, Mar. 1982.

[3] C. A. Gonzales *et al.*, "DCT coding for motion video storage using adaptive arithmetic coding," *Image Commun.*, vol. 2, pp. 145–154, Aug. 1990.

[4] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 91–95, Mar. 1992.

[5] S. Minami and A. Zakhor, "An optimization approach for removing blocking effects in transform coding," in *Proc. Data Compression Conf.*, Apr. 1991, p. 442.

[6] A. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[7] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1991.
[8] W.-Y. Chan, "Product code vector quantization methods with application to high fidelity audio coding," Ph.D. dissertation, Univ. Calif., Santa Barbara, Dec. 1991.
[9] S.-W. Wu and A. Gersho, "Improved decoder for transform coding with application to the JPEG baseline system," in *IEEE Trans. Commun.*, pp. 251–254, Feb. 1992.
[10] ———, "Nonlinear interpolative decoding of standard transform coded images and video," in *SPIE Conf. Image Processing Algorithms and Techniques III*, Feb. 1992, pp. 88–99.
[11] A. Rao, D. Miller, K. Rose, and A. Gersho, "A generalized VQ method for combined compression and estimation," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Apr. 1996, vol. 4, pp. 2032–2035.
[12] S.-W. Wu and A. Gersho, "Lapped vector quantization of images," *J. Opt. Eng.*, July 1993.
[13] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd Ed. New York: McGraw Hill, 1984.
[14] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. Commun.*, vol. 38, pp. 1285–1287, Sept. 1990.
[15] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.
[16] M. Rabbani and P. Jones, *Digital Image Compression Techniques*. SPIE, 1991.
[17] ISO/CCITT, *JPEG Draft Tech. Spec. Rev. 8*, Aug. 1990.

Siu-Wai Wu (A'96) received the B.S. degree from the Chinese University of Hong Kong in 1988, and the M.S. and Ph.D. degrees from the University of California, Santa Barbara, in 1989 and 1993, respectively.

From April 1993 to January 1996, he was a Member of the Technical Staff at AT&T Bell Laboratories, Murray Hill, NJ. During this period, he optimized the MPEG2 video compression algorithm for the U.S. HDTV Grand Alliance system, which excelled in the test conducted by the FCC in the summer of 1995. Since February 1996, he has been a Senior Staff Engineer at General Instrument, San Diego, CA, where he is currently developing HDTV broadcasting products. He has published over 20 technical papers in the areas of video compression and signal processing, and has several patents pending. He co-authored the chapter "Vector Quantization Techniques in Image Compression" in the *Handbook of Visual Communications*, (New York: Academic, 1995).

Dr. Wu received the Video Technology Transactions Best Paper Award from the IEEE Circuits and Systems Society in 1992.

Allen Gersho (S'58–M'64–SM'78–F'82), for a photograph and biography, see this issue, p. 793.