Adaptive Vector Quantization for Efficient Zerotree-Based Coding of Video with Nonstationary Statistics

James E. Fowler

***Abstract*—A new system for intraframe coding of video is described. This system combines zerotrees of vectors of wavelet coefficients and the generalized-threshold-replenishment (GTR) technique for adaptive vector quantization (AVQ). A data structure, the vector zerotree (VZT), is introduced to identify trees of insignificant vectors, i.e., those vectors of wavelet coefficients in a dyadic subband decomposition that are to be coded as zero. GTR coders are then applied to each subband to efficiently code the significant vectors by way of adapting to their changing statistics. Both VZT generation and GTR coding are based upon minimization of criteria involving both rate and distortion. In addition, perceptual performance is improved by invoking simple, perceptually motivated weighting in both the VZT and the GTR coders. Our experimental findings indicate that the described VZTGTR system handles dramatic changes in image statistics, such as those due to a scene change, more efficiently than wavelet-based techniques employing nonadaptive scalar quantizers.**

***Index Terms*—Adaptive vector quantization, image-sequence coding, zerotree coding.**

I. Introduction

THE INCREASING demand for video services over modern communication links has created great need for efficient coding techniques for the storing and transmitting of digital video. While video-coding standards such as MPEG have been developed and are well suited to many cases, new coding techniques are required to deal with the particular needs of many emerging multimedia applications. For example, there is increasing interest in providing real-time video-on-demand services over the Internet, the World Wide Web (WWW), and wireless links. Coding techniques designed for these communication channels, which are inherently unreliable due to network congestion or channel fading, must robustly handle situations in which packets of data are lost during transmission.

To compensate for the unreliable nature of their transmission, emerging networks may allow the sender to set a priority for individual packets so that the sender may specify which packets are the most important to receive. A video encoder operating on such a channel must then determine what information must be received as opposed to what information may be lost without unacceptably degrading the quality of the received video. Subband coding can provide this type of partitioning in a way that is meaningfully correlated to the operation of the human visual system while being inherently suited to implementation over priority-based packet networks.

Recent published results have demonstrated that wavelets yield excellent performance for subband image coding. Zerotree-based methods, such as the original embedded-zerotree-wavelet (EZW) algorithm by Shapiro [1] and the recent space-frequency-quantization (SFQ) technique by Xiong *et al.* [2] have, by reducing redundancy among wavelet coefficients with tree-based prediction structures (zerotrees), broken previous wavelet-coding performance barriers and dramatically advanced the state of the art in still-image coding. Naturally, one would like to extend the promising performance of these still-image techniques to video. In this paper, we propose a system for the coding of video originating in the rate-distortion-based zerotrees of SFQ, whose still-image performance ranks near the top among zerotree-based techniques.

SFQ couples simple uniform scalar quantization with zerotrees constructed using rate-distortion criteria. Specifically, given a fixed scalar-quantizer stepsize, a tree structure is built from the leaves up. In constructing this tree, the cost in distortion which would result from zeroing a subtree of wavelet coefficients is weighed against the cost in rate which would be required if that subtree were not labeled as a zerotree. After the tree is determined, the coefficients that are marked as "significant" (i.e., not zero) are coded using the uniform scalar quantizer. To determine the best quantizer stepsize to use for a particular image, the above process is repeated for each stepsize in a finite set of allowed stepsizes, and the one that yields the lowest rate-distortion cost is chosen. A Lagrangian-multiplier parameter controls the balance between distortion and rate throughout the algorithm.

The most straightforward application of SFQ to video would be perhaps to simply use the algorithm to code an image sequence frame-by-frame. However, the exhaustive search for the optimal SFQ stepsize, which essentially entails producing a multitude of codings of a single image and choosing the best one, is too computationally expensive to perform for each frame of a video sequence. Alternatively, one might envision applying SFQ frame-by-frame using a quantizer stepsize that does not change between frames. This static-quantizer approach, too, is problematic since, even if it were possible to select a stepsize yielding suitable performance over one portion of an image sequence, changes in image statistics due to the nonstationarity inherent to real video data would inevitably require some form of adaption. Scene changes in which the video content changes significantly produce dramatic shifts in image statistics and are particularly troublesome for static quantizers to handle. Yet such scene changes can be expected to occur frequently in many real-life video applications.

Manuscript received July 6, 1998; revised August 16, 2000. This work was supported by the National Science Foundation under Grant INT-9600260. This paper was recommended by Associate Editor J. Villasenor.

The author was with Laboratoire I3S, Université de Nice-Sophia Antipolis, Sophia Antipolis, France. He is now with the Department of Electrical and Computer Engineering and the NSF Engineering Research Center, Mississippi State University, Starkville, MS 39762 USA.

Publisher Item Identifier S 1051-8215(00)10624-X.

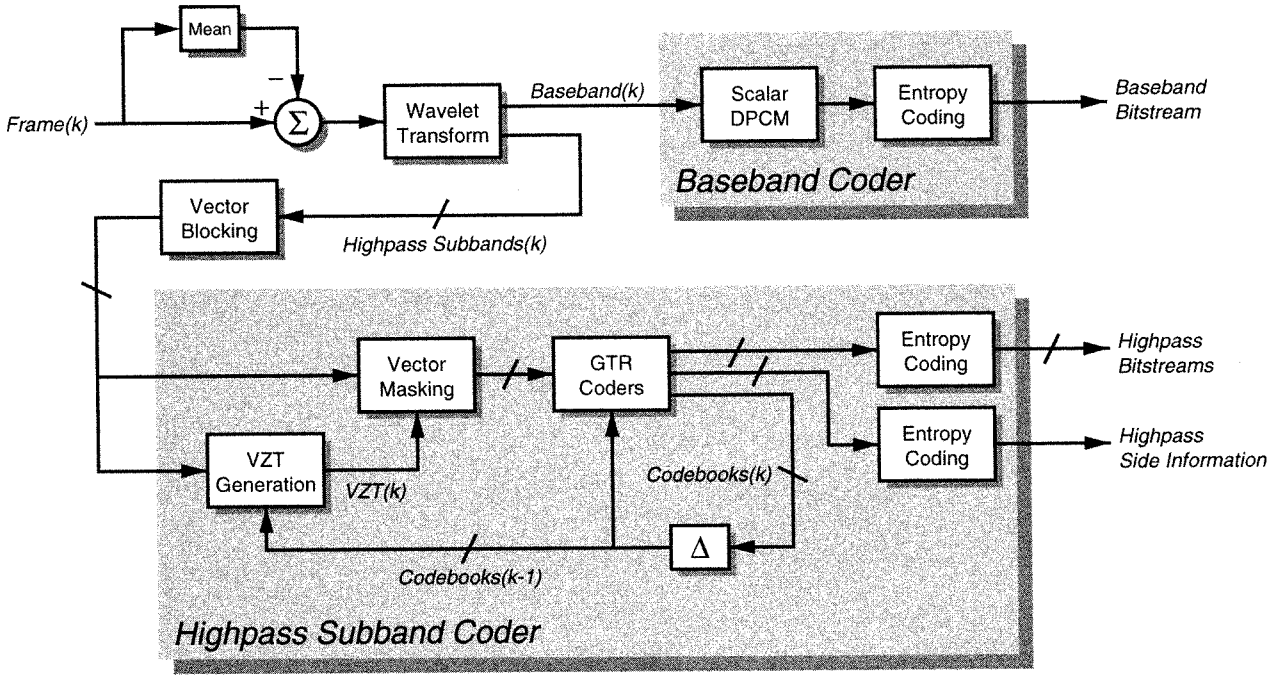


Fig. 1. Block diagram of the VZTGTR video-coding system. The block labeled with Δ indicated a frame delay. Lines labeled with a slash indicate multiple data paths (one for each subband).

In order to achieve better performance on video which includes scene changes, we capitalize on recently developed techniques [3], [4] for efficient adaptive vector quantization (AVQ) designed specifically for the coding of nonstationary sources. In this paper, we propose a system for the intraframe coding of video sequences which extends SFQ to vectors and then employs the generalized-threshold-replenishment (GTR) AVQ algorithm [3], [4] for efficient coding of vectors of wavelet coefficients. Our vector zerotree (VZT) structure efficiently describes which "insignificant" vectors are not to be coded while our GTR coders process the remaining "significant" vectors, all the while adapting to changing statistics of these vectors. In addition, we improve perceptual performance by invoking simple, perceptually motivated weighting in both the creation of the VZT and in the GTR coding.

We are motivated to "vectorize" the zerotree concept for a number of reasons. Primarily, rate-distortion theory dictates that quantization of vectors is more efficient than scalar quantization [5]. Secondly, creating zerotrees of vectors significantly reduces the number of nodes in the trees as compared to the scalar case; consequently, we expect that the burden of side information needed to represent the VZT structures will be significantly less. Finally, although nonadaptive quantization has been successfully applied to the coding of many types of data, including speech, audio, images, and video, such sources can rarely be assumed to be stationary in practice. On the other hand, AVQ in general, and GTR in particular, has been shown to achieve efficient rate-distortion performance for nonstationary sources [4]. Below, we demonstrate that our VZTGTR system operates more efficiently than SFQ using static scalar quantization when the statistics of a video stream change dramatically due to a scene change.

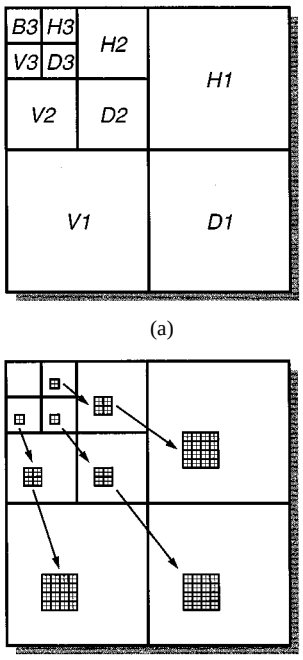
In the following, we start with an overview of our proposed VZTGTR video-coding system in Section II—we describe the structure of our VZTGTR system (Section II-A), present details on the creation of the VZT data structure via rate-distortion based criteria (Section II-B), and discuss the operation of GTR coders which adaptively code significant vectors (Section II-C). We follow with Section III which contains a body of experimental results that compare the performance of our VZTGTR system to that of SFQ on an image sequence containing a scene change. Additionally, for completeness, we present a comparison to stack-run (SR) coding [6], [7], a simple and effective wavelet-based technique using scalar quantization without zerotrees. Finally, a few concluding remarks are made in Section IV.

II. The Video-Coding System

Our VZTGTR video-coding system is depicted in Fig. 1. Our system uses a three-level dyadic subband decomposition employing the biorthogonal 9/7 wavelet filter described in [8]. This three-level decomposition results in 10 subbands as depicted in Fig. 2(a). The low-pass subband (baseband) is coded independently using scalar DPCM followed by a uniform scalar quantizer and arithmetic entropy coding. Each high-pass subband is blocked into vectors and coded as described in the following sections.

A. VZT Structure

The VZT structure used in our system is similar to the zerotree data structure used in SFQ [2], which in turn has its origins in the classic EZW technique [1]. The key structural difference between our VZT and the zerotrees of [1], [2] is that our VZT is constructed for square vectors of wavelet coefficients



(a)

(b)

Fig. 2. (a) Wavelet decomposition used in the VZTGTR system. Hl donates horizontal low-pass filtering followed by vertical high-pass filtering. Vl denotes horizontal high-pass filtering followed by vertical low-pass filtering, and Dl denotes high-pass filtering in both directions (l is the level of decomposition). (b) The VZT structure defined as a zerotree of 2×2 vectors. Baseband vectors are not included in the VZT structure as the baseband is processed independently of the high-pass bands.

rather than for scalar values. As shown in Fig. 2(b) for the case of 2×2 vectors, each 2×2 vector of wavelet coefficients at decomposition level $l > 1$ of the subband decomposition has four children vectors at level $l - 1$; these children vectors are also size 2×2. Node j at level l, n_{jl}, of our VZT corresponds to a vector i in subband s, v_{is}, of the subband decomposition. Node n_{jl} holds one of two values: S to indicate that each of the four children of vector v_{is} are significant (children of symbols S *or* Z), or Z to indicate that v_{is} is a VZT root. The occurrence of a VZT root in the VZT structure indicates that we will not code any of the descendant vectors of v_{is} (although we will code vector v_{is} itself).

B. VZT Pruning

We determine a new VZT structure for each frame of input video by starting with a full tree (i.e., a VZT structure with all nodes labeled S), and "carving" out the VZT over several iterations of a pruning algorithm. Our VZT pruning algorithm estimates the best VZT given the set of GTR codebooks produced while coding the previous frame. The VZT pruning is based on a rate-distortion criterion that determines the best VZT tree considering the cost, in terms of distortion, of coding sets of vectors as zero (as is implied by the occurrence of a zerotree root) versus the gain, in terms of rate, of not coding a zerotree of vectors. Additionally, our VZT-pruning algorithm compensates for the fact that the sensitivity of human vision to image distortion is highly dependent on the subband in which the distortion occurs. We note that our VZT-pruning algorithm differs from the tree pruning of [2] in that entropy-constrained VQ (ECVQ)

Given: current frame = k
initial VZT $V^{(0)}$, initialized to full tree
VQ codebooks $\mathcal{C}_s$ from frame $k-1$
initial probabilities $p_{cs}^{(0)}$ of codewords $c \in \mathcal{C}_s$
set of rate-distortion parameters, λ_s
set of perceptual weights, α_s
initial costs $J(v_{is}) = 0$

set iteration count $m = 0$
do
  set $m = m + 1$
  set $V^{(m)} = V^{(m-1)}$
  for level $l = 1$ to number of levels of decomposition do
    for each subband s in level l do
      for each vector v_{is} with node $n_{jl} \in V^{(m)}$, do
        calculate squared vector norm, $P(v_{is}) = \|v_{is}\|^2$
        calculate distortion, $D(v_{is}) = \|v_{is} - \hat{v}_{is}\|^2$, where

$$\hat{v}_{is} = \arg\min_{c} \left\{ \alpha_s \|v_{is} - c\|^2 + \lambda_s \left[-\log_2 \left(p_{cs}^{(m-1)} \right) \right] \right\}, \; c \in \mathcal{C}_s \quad (1)$$

        calculate updated probabilities $p_{cs}^{(m)} = N_{cs}/N_s$,
          where N_{cs} = num vectors quantized to c,
          and N_s = num vectors in subband s
        for each vector v_{is} with node $n_{jl} \in V^{(m)}$, do
          estimate rate $R(v_{is}) = -\log_2 \left(p_{cs}^{(m)} \right)$,
            where c is the winning codeword from (1)
          calculate cost $G(v_{is}) = \alpha_s D(v_{is}) + \lambda_s R(v_{is})$
          if level $l > 1$,
            calculate cost $J_1 = \sum_x \alpha_s P(x)$, where vector x
              is a descendant of v_{is}
            calculate cost $J_2 = \sum_x G(x) + J(x)$,
              where vector x is a child (level $l-1$) of v_{is}
            if $J_1 \leq J_2$,
              set $n_{jl} = Z$, $J(v_{is}) = J_1$, and remove from
                $V^{(m)}$ all descendant nodes of v_{is}
            else,
              set $n_{jl} = S$ and $J(v_{is}) = J_2$
          else,
            set $n_{jl} = S$ and $J(v_{is}) = 0$
while $V^{(m)} \neq V^{(m-1)}$

Fig. 3. VZR-pruning algorithm.

[9] replaces the uniform scalar quantization used in [2], and we modify the rate-distortion criterion to normalize distortion measures with perceptual weighting. Fig. 3 shows our VZT-pruning algorithm in detail.

Each iteration of VZT pruning starts at the bottom of the tree and works its way to the top. For each vector in each subband of the current level, the rate-distortion nearest neighbor is determined using (1) in Fig. 3; this equation is the ECVQ nearest-neighbor rule with a perceptually weighted distortion calculation. Once each vector has been assigned a nearest neighbor, new partition probabilities are calculated. The algorithm estimates the best cost, $J(v_{is})$, associated with the descendants of current vector v_{is} by examining two possible cases. That is, the algorithm decides whether the current node in the VZT should be a zerotree root or not. The algorithm estimates the cost (J_1 in Fig. 3) of assigning a zerotree root to the current node as the sum of the squared vector norms of all descendants of v_{is}, as this will be the total distortion produced by zeroing out the descendants. On the other hand, the cost (J_2 in Fig. 3) of not zeroing out all the descendants of the current vector is estimated as the rate-distortion cost of coding the children of v_{is}, $G(x) = \alpha_s D(x) + \lambda_s R(x)$, plus the best cost, $J(x)$, associated with descendants of the children of v_{is}, where x is a child of v_{is}. The best cost associated with the descendants of v_{is} is

then $J(v_{is}) = \min(J_1, J_2)$. If the algorithm decides to set the current node to be a zerotree root, all the nodes corresponding to descendants of v_{is} are removed from the VZT. The algorithm iterates until the VZT remains unchanged between iterations.

The perceptual weights α_s serve to "normalize" each distortion calculation in view that the perceptual effect of distortion on image quality varies significantly from subband to subband. Our use of perceptual weights follows the approach taken in [10] and [11], in which values for just-noticeable distortion (JND) were determined following perceptual experiments for base sensitivity: random noise was added to a mid-gray background in each subband and the variance of the noise was increased until the noise was just noticeable against the background. The perceptual weights, α_s, used in our video-coding system are the reciprocals of the JND values reported in [10]. Although in general it would be possible for the values of α_s to vary from frame to frame in coding a sequence of images, we currently fix the α_s values to be the same for all frames.

C. AVQ with GTR

The vectors that are not "pruned" as indicated by the newly determined VZT are passed to a set of GTR coders. GTR, an online AVQ algorithm based on rate-distortion criteria, has been discussed extensively elsewhere [3], [4], so only a brief review will be presented here.

The GTR algorithm operates as follows. First, the GTR coder determines the codeword "closest" to the current source vector in a rate-distortion sense. That is, for current vector i in subband s, v_{is}, codebook $\mathcal{C}_s$ is searched to find the rate-distortion-based nearest neighbor that minimizes $J = D(c) + \gamma_s R(c)$, where codeword c is in codebook $\mathcal{C}_s$, $D(c)$ is the distortion between v_{is} and c, and $R(c) = -\log_2 p_{cs}$ is a probability-based rate estimate. This nearest nearest-neighbor search is exactly that of the well known ECVQ algorithm [9]; to use the terminology introduced in [4], [12], we call this nearest-neighbor-search process the *vector coder* of the GTR algorithm.

Once the rate-distortion-based nearest neighbor, c^*, is determined, the algorithm decides whether an update to the codebook is warranted. If the GTR coder decides to update the codebook, the current vector may be coded with less distortion; however, the codebook update will necessitate the transmission of additional bits (so-called "side information") to inform the decoder of the update. To decide if an update is warranted, an update cost function, $\Delta J = \Delta D + \gamma_s \Delta R$ is calculated, where ΔD is the potential gain in distortion due to an update, ΔR is the cost in side information of the update, and γ_s is a Laplacian constant dictating the tradeoff between rate and distortion in the GTR coder for subband s. Suppose we do not update the codebook; in this case, the current source vector will be coded with a distortion of $D(c^*)$, the distortion between v_{is} and c^*. On the other hand, if the codebook is updated, we will add some vector $\hat{v}_{is}$ to the codebook resulting in v_{is} being coded with distortion $D(\hat{v}_{is})$, the distortion between v_{is} and $\hat{v}_{is}$ Thus, the potential improvement in distortion due to a codebook update is $\Delta D = D(\hat{v}_{is}) - D(c^*)$. If $\Delta J < 0$, the gain in distortion outweighs the cost in rate, and the codebook is updated. Otherwise, merely the index of c^* is sent to the decoder. This process is then repeated for the next source vector.

To update the codebook, a GTR coder must determine an update vector $\hat{v}_{is}$ for the current source vector. In the terminology of [4], [12], the coding of this update vector is called the *codebook coder* process of the GTR algorithm and is responsible for side information transmitted to the decoder. In general, the overall rate-distortion performance of the GTR coder will depend on the coding efficiency of not only the vector coder (the mapping of v_{is} to c^*) but also the codebook coder (the mapping of v_{is} to $\hat{v}_{is}$). In the general model of AVQ systems presented in [4], [12], the codebook coder and the vector coder are two fundamental components common to all AVQ systems. However, the interoperation of these two components is currently not well understood and is, in general, complex under general conditions (for some insights under certain simplifying assumptions, refer to Zeger *et al.* [13]). Because of this complexity, most AVQ literature has not pursued codebook-coder design in depth and has instead relied upon simple codebook coders, usually in the form of a uniform scalar quantizer applied to each component of the update vector. With simple codebook coders, it is often assumed that side information accounts for only a small portion of the total rate, allowing one to incorporate high-resolution quantization and ignore the distortion introduced by the codebook coder. In our VZTGTR system described here, we too use simple uniform scalar quantization for codebook coding in our GTR coders. However, recognizing that the rate due to side information is often, in practice, nonnegligible, we choose empirically the stepsizes q_s of our codebook coders so as to yield optimal rate-distortion performance (more details on this process follow in Section III).

We conclude this section by noting some implementation details pertaining to the use of GTR in our VZTGTR system. Each GTR coder starts coding an input video sequence with a null codebook, i.e., a codebook with no codewords. The codebook is then populated through codebook updates until a maximum of 256 codewords is reached, after which each codebook update necessitates the removal of an existing codeword. This codeword removal is accomplished via the move-to-front variant of the GTR algorithm as described in [3], [4]. Additionally, we incorporate perceptual weighting in each GTR coder; i.e., in the GTR coder for subband s, we use $\gamma_s = \lambda_s/\alpha_s$, where λ_s and α_s are, respectively, the rate-distortion and perceptual-weight parameters used previously in VZT pruning. Each GTR coder produces a sequence of VQ indices (from the vector coder), as well as side information (from the codebook coder). Each VQ-index sequence is coded independently using arithmetic coding, producing a separate bitstream for each subband. The side information from each GTR coder consists of a map of binary flags indicating, for each vector coded in the subband, whether an update occurs, and, in the case of an update, the vector components of the updated vectors. The side information for all subbands is multiplexed together and combined with the VZT information; i.e., the update flags are combined with the VZT symbols, resulting in an stream of symbols from a four-symbol alphabet. This symbol stream is entropy coded using arithmetic coding.

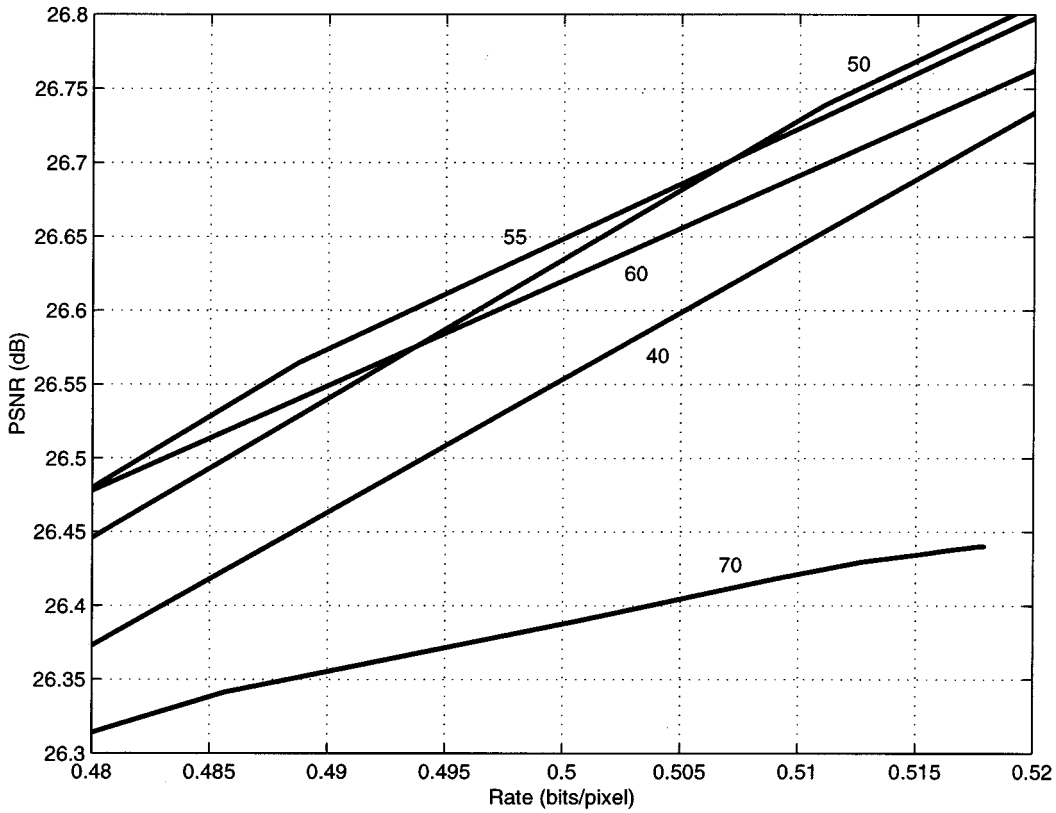


Fig. 4. Rate-distortion curves for the VZTGTR system for a variety of codebook-coder quantizer stepsizes q over the "Football" portion of the "Football-Susie" test sequence. Each curve is calculated by increasing λ from zero while holding q fixed.

The update-vector components are coded as described above using a uniform scalar quantizer followed by arithmetic entropy coding.

III. Experimental Results

We now describe experiments conducted to compare the performance of our VZTGTR system with that of other wavelet-based coding algorithms, namely SFQ [2] and SR [6], [7]. Additionally, we present results for the MPEG-1 video-coding standard [14] which does not employ wavelets but uses a block-based DCT transform. We focus our attention on the situation in which the statistics of an image sequence change dramatically due to a scene change. To simulate the scene change, we use a 200-frame image sequence composed of 125 frames from the "Football" sequence followed by 75 frames from the "Susie" sequence. This test sequence is grayscale with a spatial resolution of 352×240 pixels (SIF resolution) and a temporal sampling of 30 frames/s (noninterlaced). We arrange the experiment so that the algorithms code the initial 125 frames (the "Football" portion) of the sequence at the same target bit rate and then examine performance after the scene change. We note that all rate values are calculated from "real" bitstreams; for the VZTGTR, SFQ, and SR algorithms, these bitstreams are produced by adaptive arithmetic coders [15]. All rate figures are presented in bits/pixel (bpp). Each of the wavelet-based techniques considered uses a 3-level dyadic wavelet decomposition implemented with the biorthogonal 9/7 filter coefficients given in [8].

In Sections III-A–D, we "optimize" the operation of the coding techniques to the "Football" portion of our "Football-Susie" test sequence. That is, the parameters of each technique are selected to provide best average distortion performance for a target bit rate over the "Football" portion of the test sequence. Then, in Section III-E, we examine the performance of the algorithms, with parameters unchanged, over the latter "Susie" portion of the "Football-Susie" test sequence.

Throughout the results, we select a target bit rate of 0.5 bpp which, at SIF resolution and full-frame rate, corresponds to an output channel rate of approximately 1.27 Mbps, about the bandwidth of a CDROM ($\approx$1.5 Mbps). For applications whose bandwidth is more constrained, as may be the case for network-based packet video, we may be able to meet the desired channel rate by reducing the frame rate or spatial resolution.

A. The VZTGTR System

The rate-distortion performance of our VZTGTR system depends mainly on the values of q_s, the stepsizes of the uniform scalar quantizers used by the codebook coders to code update vectors, and λ_s, the parameters that determine the balance between rate and distortion throughout the system. In general, our system allows q_s and λ_s to vary by subband; however, for simplicity in the results here, we constrain the system to use the same values for all subbands. That is, $q_s = q$ and $\lambda_s = \lambda$ for each subband s. We "optimize" these q and λ over the "Football" portion of the "Football-Susie" test sequence by the following process.

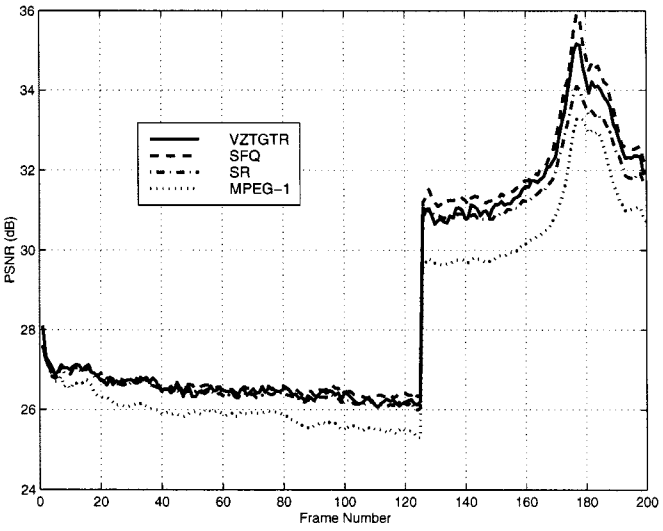
First, we select q so as to provide the best rate-distortion performance over the initial frames of the test sequence. That is, considering average rate and average distortion over only the "Football" portion of the test sequence, we calculate rate-distortion performance curves for a variety of q values and choose the one that yields the highest PSNR at the target rate of 0.5 bpp. Fig. 4 shows these rate-distortion curves; $q = 55$ is the chosen stepsize. Next, to meet the target bit rate over the initial frames, we fix $q = 55$ and adjust, by trial and error, the rate-distortion parameter λ so that the rate, averaged over the first 125 frames of the "Football-Susie" test sequence, is approximately 0.5 bpp. This $\lambda(\lambda = 150)$ is then used in the coding of each subband in each image of the test sequence.

We use a VZTGTR structure built upon 2×2 vectors of wavelet coefficients. Although larger vector dimensions allow, theoretically, for greater rate-distortion efficiency in the vector coder, clearly the additional side information required to transmit update vectors mitigates to some extent this advantage. Additionally, large vector sizes result in less flexibility in determining zerotrees of insignificant vectors since large regions of coefficients must be set to zero for a zerotree to occur. These large regions of zero coefficients tend to over-smooth the image resulting in reduced visual quality. Our experience has indicated that a vector dimension of 2×2 gives an appropriate tradeoff between vector-coder efficiency, side-information burden, and visual quality.

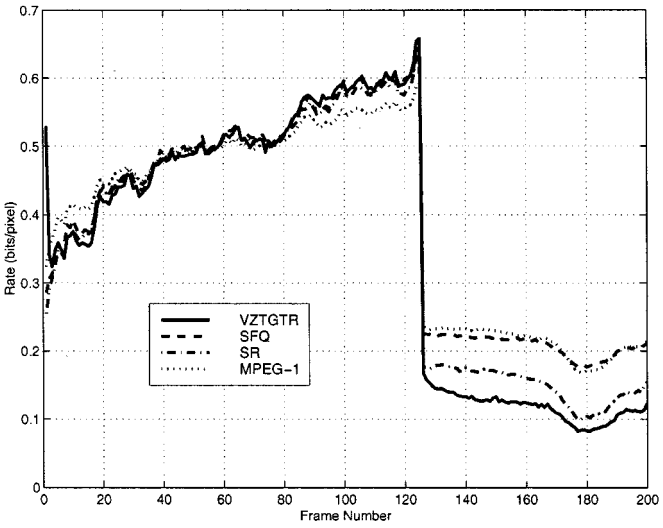
Our VZTGTR system uses DPCM followed by uniform scalar quantization for the coding of the baseband; this intraband DPCM uses a simple 2-D scalar predictor averaging neighboring intraband pixels above and to the left of the current pixel. We use the same quantizer stepsize, $q_{\rm bb} = 31.5$, that we use in our implementation of SFQ (see below) to quantize the difference values resulting from the DPCM predictor.

We use an implementation of the arithmetic coder described by Witten *et al.* [15] to code the various quantities output by the VZTGTR system. Specifically, our implementation allows the definition of multiple contexts in which the arithmetic-coding probability tables can be independently adapted. In the VZTGTR system, we use separate arithmetic-coding contexts for the update-vector components, the VQ indices, and the joint VZT-symbol/GTR-update-flag symbol stream.

Finally, we note that our current (nonoptimized) implementation of the VZTGTR system, with all parameters fixed, takes about 1.8 s on a Pentium II 266-MHz 128-MB computer to code a single frame of the test sequence.



(a)



(b)

Fig. 5. Performance of the VZTGTR video-coding system versus that of the SFQ [2] algorithm, the SR [6], [7] algorithm, and MPEG-1 [16] on the "Football-Susie" test sequence consisting of 125 frames from "Football" followed by 75 frames from "Susie." (a) Distortion. (b) Rate.

B. The SFQ Algorithm

The SFQ algorithm, as presented in [2], was designed to code single images. Consequently, we apply SFQ in a frame-by-frame fashion to code our "Football-Susie" test image sequence. As we did for VZTGTR in the previous section, we "optimize" the performance of SFQ to the initial "Football" portion of the test sequence. To do so, we must select a scalar-quantizer stepsize q and rate-distortion parameter λ for the SFQ algorithm.

The rate-distortion performance of the SFQ algorithm is closely tied to its scalar-quantizer stepsize which is used in each of the high-pass subbands (the low-pass subband is coded independently—see below). The original description of SFQ called for restricting the possible q to a finite set, and then performing an exhaustive search of these allowed stepsizes to find the best one for a particular image. Specifically, the allowed stepsizes are $q = 7.5 + 0.1k$ where $k = 1, 2, \ldots, 245$. Our implementation of SFQ, using an exhaustive search over these 245 possible stepsizes, requires about 106 s of computation (on a Pentium II 266-MHz 128-MB computer) to code a single frame of the test sequence. However, the coding of this same frame with the stepsize already specified takes only 0.92 s.

Since an exhaustive search is too computationally expensive to perform for each image of a video sequence, we fix the q to a preselected value which is then used to code the entire sequence. To determine this stepsize, we perform the exhaustive search described above on only the first frame of the sequence. Afterwards, we adjust by trial and error the rate-distortion parameter, λ, so that the rate, averaged over the initial 125 frames

(a) (b) (c) (d)

Fig. 6. Reconstructed frames from the "Football-Susie" sequence. (a) Frame 60, VZTGTR (26.5 dB, 0.510 bpp). (b) Frame 160, VZTGTR (31.5 dB, 0.119 bpp). (c) Frame 60, SFQ (26.5 dB, 0.523 bpp). (d) Frame 160, SFQ (31.7 dB, 0.214 bpp).

with q fixed, is approximately 0.5 bpp. This $\lambda(\lambda = 378)$ is then used to code the entire "Football-Susie" test sequence.

In our implementation of the SFQ algorithm, the baseband is coded with uniform scalar quantization. The baseband quantizer stepsize ($q_{\rm bb} = 31.5$) is chosen to provide the best rate-distortion performance (over the baseband alone) for the λ chosen above; the stepsizes $q_{\rm bb} = 7.5 + 0.1k$, $k = 1, 2, \ldots, 245$, are searched to find the optimal stepsize.

The scalar-quantizer indices produced by the SFQ algorithm are entropy coded using the same multiple-context arithmetic coder used for the VZTGTR system. We use a separate arithmetic-coding context for the quantizer indices of each subband.

C. The SR Algorithm

The SR algorithm [6], [7], like SFQ, is a wavelet-based, still-image coding method. However, unlike SFQ, SR does not use zerotrees for efficient coding of insignificant coefficients; instead, SR applies scalar quantization to the wavelet coefficients and codes, in raster-scan fashion within the subbands, the lengths of runs of zeros between significant (i.e., nonzero) coefficients. Afterwards, an arithmetic coder operating with a novel four-symbol alphabet codes the run lengths and the significant-coefficient values.

As described in [6] and [7], SR uses a single static-uniform scalar quantizer over all the subbands of an image (the authors do, however, mention the possibility of applying dead-zone quantizers to increase the lengths of zero runs). The rate-distortion performance of the SR algorithm is determined entirely by the stepsize of this uniform scalar quantizer. Although SR is a fast coding algorithm (our implementation requires 0.51 s to code one image of our "Football-Susie" test sequence), it faces the same problem as SFQ when applied to video in that exhaustively searching for an optimal stepsize for each frame is infeasible, while scene changes in video content require that some form of adaption be present in the algorithm. To illustrate this point, we apply the SR algorithm frame-by-frame to the "Football-Susie" test sequence which contains a scene change. To do so, we select the SR scalar-quantizer stepsize so that the rate averaged over the initial "Football" portion of the test sequence is 0.5 bpp and then use this stepsize ($q = 70$) to code the entire test sequence.

To entropy code scalar-quantizer indices, our implementation of the SR algorithm employs a two-context adaptive arithmetic coder as described in [6] and [7]; the implementation of this coder is the same as that used for the VZTGTR system and the SFQ algorithm.

D. The MPEG-1 Standard

We use the University of California Berkeley MPEG-1 coder [16] to provide a comparison between the wavelet-based al-

(a) (b) (c) (d)

Fig. 7. Reconstructed frames from the "Football-Susie" sequence. (a) Frame 60, SR (26.3 dB, 0.514 bpp). (b) Frame 160, SR (31.2 dB, 0.159 bpp). (c) Frame 60, MPEG (25.8 dB, 0.508 bpp). (d) Frame 160, MPEG (30.2 dB, 0.221 bpp).

TABLE I
AVERAGE DISTORTION AND RATE PERFORMANCE OF THE ALGORITHMS OVER THE "FOOTBALL-SUSIE" TEST SEQUENCE

	"Football" portion only		"Susie" portion only		Entire test sequence	
	PSNR (dB)	Rate (bpp)	PSNR (dB)	Rate (bpp)	PSNR (dB)	Rate (bpp)
VZTGTR	26.7	0.500	32.2	0.117	28.0	0.356
SFQ	26.6	0.503	32.4	0.208	28.0	0.392
SR	26.5	0.497	31.8	0.149	27.8	0.367
MPEG-1	25.9	0.495	30.7	0.217	27.7	0.389

gorithms described above and more traditional DCT-based approaches as exemplified by the MPEG-1 video-coding standard. As we are considering only intraframe coding in the results presented here, we disable motion compensation in the MPEG coder (i.e., only I-frames are used to code the "Football-Susie" test sequence). As before, we "optimize" the MPEG coder to the target bit rate of 0.5 bpp over the initial "Football" portion of the test sequence. To do so, we adjust the quantization-scale value ("quality factor") of the MPEG coder until the bit rate averaged over the initial "Football" frames is 0.5 bpp. We then maintain this quantization scale over the remainder of the test sequence. All bit rates and distortion values reported for the MPEG coder are calculated for only the Y (luminance) component of the image sequence. The execution time on the "Football-Susie" sequence for this implementation of MPEG-1 is approximately 0.1 s/frame.

E. Performance After Scene Change

The above procedures can be considered to have "optimized" the operation of the coding techniques to the initial frames of the "Football-Susie" test sequence. That is, the algorithm parameters of each technique were selected to provide the best distortion performance for an average rate of 0.5 bpp over the "Football" portion of the test sequence. We now investigate performance after the occurrence of a scene change in which the image statistics differ significantly from those on which the "optimized" parameters were originally determined. To do so, we allow the algorithms under consideration to continue coding the "Football-Susie" test sequence on the latter 75 frames (the "Susie" portion) with parameters unchanged from those used to code the initial frames of the sequence. The frame-by-frame evolution of the rate and distortion performance of the algorithms over the entire "Football-Susie" sequence is given in Fig. 5, while rate and distortion figures averaged over portions of the sequence are shown in Table I. Frames from both the "Football" and "Susie" portions of the reconstructed output sequences are presented in Figs. 6 and 7.

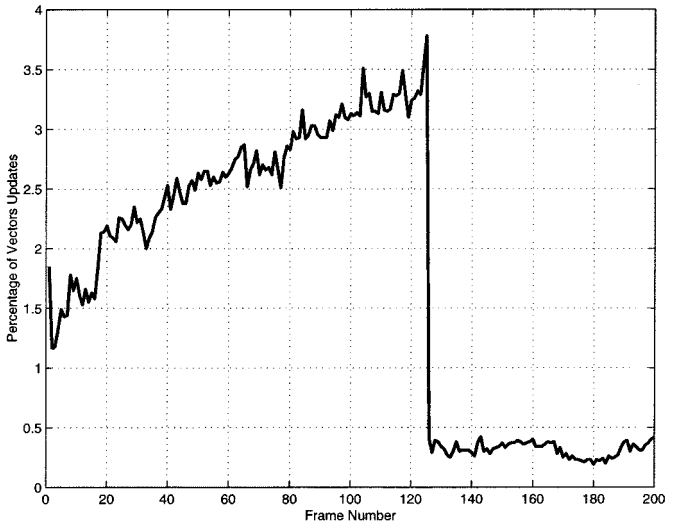
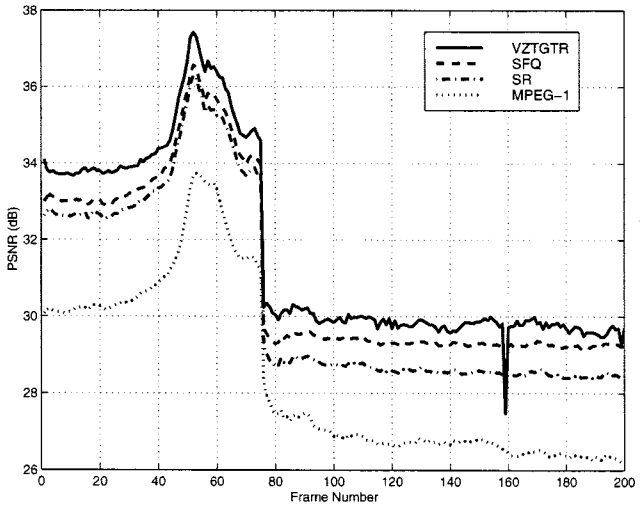


Fig. 8. Percentage of vectors per frame that are updated by the VZTGTR system while coding the "Football-Susie" test sequence. The average percentage of updates over the entire sequence is 1.7%.

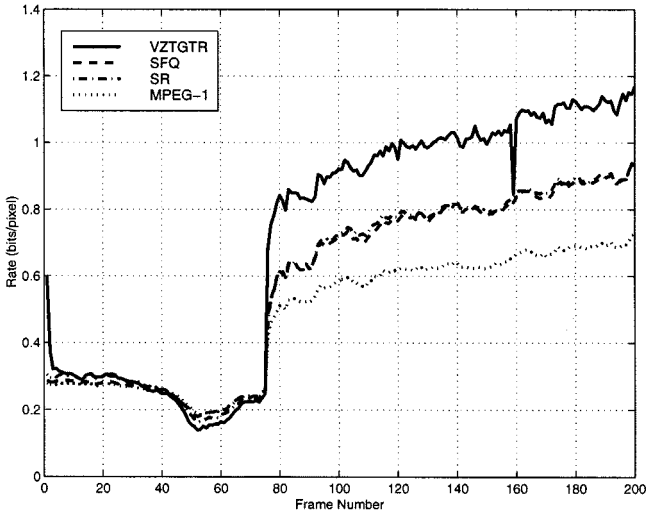
TABLE II
PERCENTAGE OF TOTAL BIT RATE BY SUBBAND FOR THE VZTGTR SYSTEM AVERAGED OVER THE ENTIRE "FOOTBALL-SUSIE" TEST SEQUENCE

Subband	VQ Indices	Side Information	Total
B3	18.2%	n/a	18.2%
H3	4.8%	4.2%	9.0%
V3	4.7%	4.1%	8.8%
D3	3.9%	2.6%	6.5%
H2	8.1%	5.0%	13.1%
V2	7.5%	5.2%	12.7%
D2	4.7%	1.6%	6.3%
H1	12.2%	1.6%	13.8%
V1	10.0%	1.6%	11.6%
D1	0.0%	0.0%	0.0%
Total	74.1%	25.9%†	100.0%

The total side information consists of 14.7% for the coding of update-vector components and 11.2% for the joint coding of GTR update flags and VZT symbols.



(a)



(b)

Fig. 9. Performance of the VZTGTR video-coding system versus that of the SFQ [2] algorithm, the SR [6], [7] algorithm, and MPEG-1 [16] on the "Susie-Football" test sequence. (a) Distortion. (b) Rate.

In Fig. 5 and in Table I, we observe that the wavelet-based algorithms have nearly identical rate-distortion performance over the "Football" portion of the test sequence. However, the VZTGTR system achieves significantly superior performance after the scene change. Over all of the "Susie" portion of the sequence, VZTGTR operates at a substantially lower bit rate than SFQ (from Table I, nearly 43% lower on average) while achieving approximately the same or slightly lower PSNR for the frames after the scene change (32.1 dB average PSNR versus 32.4 dB for SFQ). Additionally, VZTGTR clearly outperforms SR, obtaining greater PSNR at a lower rate. Finally, we note that all the wavelet-based algorithms outperform MPEG- 1 over the entire sequence in terms of rate-distortion efficiency.

From Figs. 6 and 7, similar conclusions can be drawn regarding visual-quality performance of the wavelet-based techniques. Over the "Football" portion of the sequence, the visual quality achieved by the wavelet-based algorithms is nearly identical. However, the VZTGTR system maintains a better looking image after the scene change. In particular, the VZTGTR coder gives better reproduction of edges and areas of detail. Our observations indicate that this superior perceptual performance is due to both the perceptual weightings present in the VZTGTR system as well as to the GTR coders which tend to preserve edges and other areas of high detail [4]. In comparing the performance of the wavelet-based techniques to that of MPEG-1, we note that, whereas the wavelet transform tends to distort the image by "blurring," the block-based DCT of MPEG-1 tends to generate significant "blocking" artifacts. Although it is generally difficult to perform a meaningful comparison between severely blurry and severely blocky images, the visual performance of MPEG-1 over the "Football-Susie" sequence appears to be roughly equivalent to or marginally better than that of the wavelet-based techniques even though its PSNR performance is lower.

The GTR coders in the VZTGTR system constantly transmit update vectors during the coding process, as can be seen in Fig. 8 in which the percentage of update vectors per frame is plotted for each frame of the "Football-Susie" sequence. On average, 1.7% of the vectors of the test sequence are updated; these updates account for 14.7% of the total bit rate. A breakdown of the total bit rate by subband is given in Table II.

Finally, the results above illustrate that the VZTGTR system successfully adapts from a fast action sequence ("Football") to less active content ("Susie"). Fig. 9 shows the results for a "Susie-Football" sequence in which the roles of the "Football" and "Susie" sequences have been reversed. For these results, the procedures of Sections III-A through III-D have been repeated, this time optimizing the algorithms for the initial "Susie" portion of the test sequence. The target bit rate for the initial portion of the sequence was 0.25 bpp. For this "reversed" test sequence, we see that the VZTGTR system achieves greater PSNR for nearly the entire sequence. The latter "Football" portion of the test sequence will inherently require an increased rate over that of the initial frames in order to maintain visual quality; the VZTGTR system, through its adaption mechanism, is better able to provide this increased rate. The other algorithms, which tend to code the latter "Football" frames at a rate closer to that used for the initial portion of the sequence, achieve, consequently, lower PSNR performance. The visual-quality performance of the algorithms after the scene change in the "Susie-Football" sequence is similar to that observed above for the "Football-Susie" sequence.

IV. Conclusion

The primary task of a video-coding system is to maintain consistent visual quality at the decoder for the entire sequence [17]. The key difficulty in applying many image-coding algorithms to this task is the selection of algorithm parameters. Even if it is possible to select, *a priori*, parameters yielding suitable performance over one portion of an image sequence, dramatic shifts in statistics due to scene changes inevitably require some form of adaption.

For instance, the performance of the SFQ algorithm is closely tied to its scalar-quantizer stepsize. However, determining an optimal stepsize for each image of a video sequence is clearly infeasible from a computational standpoint. It is also unwarranted—our observations indicate that the optimal stepsize often changes little over a single scene. Yet, failure to update the stepsize after a scene change may yield dire consequences for rate-distortion performance, as well as for visual quality, as is evidenced in Figs. 5–7.

By adding AVQ coders to a rate-distortion-based zerotree framework, the VZTGTR system incorporates into SFQ an adaption mechanism necessary for efficiently handling scene changes. The VZTGTR system adds vectors to its codebooks as needed to satisfy rate-distortion criteria, and is thus better able to adjust its rate to maintain visual quality. In the experiments outlined above, this codebook updating occurs for an average of 1.7% of the vectors in each frame of the "Football-Susie" sequence, a test sequence containing a scene change. On the other hand, the bits required to transmit the update vectors to the decoder for this sequence account for only 14.7% of the total bit rate. When the static scalar quantizer used to code the update vectors is mismatched to the source as is the case during the latter frames of the "Football-Susie" sequence, the rate-distortion performance of the VZTGTR system may suffer. However, the resulting inefficiency is much less than that incurred by SFQ, whose static scalar quantizer, used in the coding of 100% of the significant wavelet coefficients, is much more crucial to the rate-distortion performance of the algorithm. In addition to superior rate-distortion performance, the VZTGTR also produces better perceptual quality for most of the frames following the scene change due to both perceptual weighting present in the VZTGTR system as well as to the GTR coders, which tend to preserve edges and other areas of high detail.

In concluding, we make several remarks concerning issues open to future work. First, we note that our VZTGTR system provides natural priority partitioning of the coded bitstream not present in the other video coding methods such as MPEG. For transmission over priority-capable packet-based networks, we anticipate increasing resilience to packet-loss by sending baseband and side-information data streams at highest priority while sending high-pass subbands with decreasing priorities based on their respective location in the subband tree. Our future plans include the investigation of the performance of VZTGTR over such priority-based transmission under packet-loss conditions.

In addition, we note that we have restricted our experiments here to intraframe coding as an efficient intraframe technique is the basis for successful motion-compensated approaches. The incorporation of motion compensation to our VZTGTR system in such a way as to remain resilient to packet loss is nontrivial and remains a topic for future investigation.

Finally, the λ_s parameters used in the VZTGTR system determine a balance between rate and distortion achieved by the system. For the results presented here, we have determined suitable λ_s values by trial and error; although this simple approach is sufficient for our needs here, it is impractical in a general setting, and one might choose to employ more sophisticated methods for searching the space of rate-distortion parameters (e.g., those proposed in [18]). Whereas some video-coding techniques may attempt to provide constant-distortion or constant-rate performance, in the experiments presented here for our VZTGTR system, it is the balance between these two quantities that is held constant. However, as stated above, the true aim should be to maintain consistent visual quality. Although our experimental results indicate that the VZTGTR system approaches this goal more closely than the other techniques examined, we anticipate that allowing time-varying and subband-varying λ_s values will help better achieve the goal of consistent quality. Our future plans include incorporating into the VZTGTR system mechanisms that dynamically adjust λ_s values as coding progresses to better maintain visual quality throughout the entire sequence.

References

[1] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[2] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, pp. 677–693, May 1997.
[3] J. E. Fowler and S. C. Ahalt, "Adaptive vector quantization using generalized threshold replenishment," in *Proc. IEEE Data Compression Conf.*, J. A. Storer and M. Cohn, Eds. Snowbird, UT, Mar. 1997, pp. 317–326.
[4] J. E. Fowler, "Generalized threshold replenishment: An adaptive vector quantization algorithm for the coding of nonstationary sources," *IEEE Trans. Image Processing*, vol. 7, pp. 1410–1424, Oct. 1998.
[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, ser. Kluwer International Series in Engineering and Computer Science. Norwell, MA: Kluwer, 1992.
[6] M.-J. Tsai, J. D. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 519–521, Oct. 1996.
[7] ——, "Stack-run coding for low bit rate image communication," in *Proc. Int. Conf. Image Processing*, Lausanne, Switzerland, Sept. 1996, pp. 681–684.
[8] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.
[9] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 31–42, Jan. 1989.
[10] I. Höntsch, L. J. Karam, and R. J. Safranek, "A perceptually tuned embedded zerotree image coder," in *Proc. Int. Conf. Image Processing*, Santa Barbara, CA, Oct. 1997, pp. 41–44.
[11] R. J. Safranek and J. D. Johnston, "A perceptually tuned sub-band image coder with image dependent quantization and post-quantization data compression," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, 1989, pp. 1945–1948.
[12] J. E. Fowler, "Adaptive vector quantization for the coding of nonstationary sources," Ph.D. dissertation, The Ohio State Univ., Columbus, 1996.
[13] K. Zeger, A. Bist, and T. Linder, "Universal source coding with codebook transmission," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 336–346, Feb./Mar./Apr. 1994.
[14] *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbits/s*, MPEG-1 Video Coding Standard, 1993.
[15] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, June 1987.
[16] K. L. Gong and L. A. Rowe, "Parallel MPEG-1 video encoding," in *Proc. Int. Picture Coding Symp.*, Sacramento, CA, Sept. 1994.
[17] T. V. Lakshman, A. Ortega, and A. R. Reibman, "VBR video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, pp. 952–973, May 1998.
[18] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.