Highly Scalable Wavelet-Based Video Codec for Very Low Bit-Rate Environment

Jo Yew Tham, Surendra Ranganath, and Ashraf A. Kassim

***Abstract*—In this paper, we introduce a highly scalable video compression system for very low bit-rate videoconferencing and telephony applications around 10–30 Kbits/s. The video codec first performs a motion-compensated three-dimensional (3-D) wavelet (packet) decomposition of a group of video frames, and then encodes the important wavelet coefficients using a new data structure called *tri-zerotrees* (TRI-ZTR). Together, the proposed video coding framework forms an extension of the original zero tree idea of Shapiro for still image compression. In addition, we also incorporate a high degree of video scalability into the codec by combining the layered/progressive coding strategy with the concept of embedded resolution block coding. With scalable algorithms, only one original compressed video bit stream is generated. Different subsets of the bit stream can then be selected at the decoder to support a multitude of display specifications such as bit rate, quality level, spatial resolution, frame rate, decoding hardware complexity, and end-to-end coding delay. The proposed video codec also allows precise bit rate control at both the encoder and decoder, and this can be achieved independently of the other video scaling parameters. Such a scheme is very useful for both constant and variable bit rate transmission over mobile communication channels, as well as video distribution over heterogeneous multicast networks. Finally, our simulations demonstrated comparable objective and subjective performance when compared to the ITU-T H.263 video coding standard, while providing both multirate and multiresolution video scalability.**

***Index Terms*—Motion compensation, multirate video scalability, multiresolution, tri-zerotrees, video coding, wavelet transform.**

I. Introduction

DIGITAL satellite broadcasting, desktop videoconferencing, video-CD playback, video-on-demand, Internet retailing, telebanking, and many other services will become ubiquitous by the turn of the century. However, the biggest drawback of digital technology is the voluminous amount of data it generates. For example, a typical NTSC color video frame, with 720 pixels $\times$ 480 lines, 8 bits/pixel per color, and 30 frames/s, requires a large transmission capacity of 237 Mbits/s. Without any compression, a compact disk (CD) with a storage capacity of about 650 Mbytes can store only approximately 20 s of NTSC video. Furthermore, full motion playback is unlikely due to slow data transfer rates (between 300 Kbits/s and 1.8 Mbits/s) of typical CD-ROM devices on the market. Other applications such as desktop videoconferencing and telephony are also limited by the bandwidth constraint of most telephone modems (below 33.5 Kbits/s). All of these applications motivate the need for a good and efficient video compression algorithm which can produce acceptable video quality at compression ratios of about 150:1 or higher. Hence, an efficient very low bit-rate compression algorithm will form the enabling technology [8] for the implementation of many advanced digital applications.

Having a powerful compression scheme alone, however, may not be the complete solution to some applications such as image/video database browsing and multipoint video distribution over heterogeneous networks. There is also a growing need for other useful features such as video scalability. The term *scalable* refers to methods which allow partial decodability of different portions of the same compressed bit stream by the decoder in order to meet certain requirements. Consider the scenario of a multiparty videoconferencing session in which the parties may have systems with very different hardware configurations, and are connected via an inhomogeneous network such as the Internet. High-end parties will expect a high-quality session, while lower end parties are constrained by their slower CPU's, lower memory, and narrower connection bandwidths. This creates the need to produce a common data stream which can simultaneously fulfill the differing requirements and limitations of the various parties. If a high bit-rate data stream is transmitted to all parties, congestion and unexpected corruption of the data delivered to lower end receivers may occur. On the other hand, a low-bandwidth data stream will unnecessarily penalize higher end parties who can afford to subscribe to a better quality session. Therefore, it would be useful to have a highly *scalable*[1] video compression scheme [23], [24], [28] which allows selective transmission of different substreams (in terms of data packets) of compressed video to different parties, depending on their respective needs. In this manner, each party can have the best possible quality session, independently of other party's constraints. A similar scalability issue is also useful in a video-on-demand scenario.

From the above examples, it is evident that both high compression and scalability are desirable to meet the demands of emerging digital video applications. Much research work is geared toward achieving these two goals. Hence, before we proceed further, we review some of the related works. First,

Manuscript received August 20, 1996; revised February 28, 1997. This work was supported in part by the Wavelets Strategic Research Programme (Department of Mathematics, National University of Singapore) under MOE/NSTB Grant RP960 601/A. This paper was presented in part at the IEEE International Workshop on Intelligent Signal Processing and Communication Systems, Singapore, November 1996.

The authors are with the Department of Electrical Engineering, National University of Singapore, 119260 Singapore.

Publisher Item Identifier S 0733-8716(98)00307-2.

[1]The conventional video compression schemes such as MPEG-1 and 2 [10], [12], H.261 [25], and H.263 [7] are inherently nonscalable, although limited scalability features have been proposed [3].

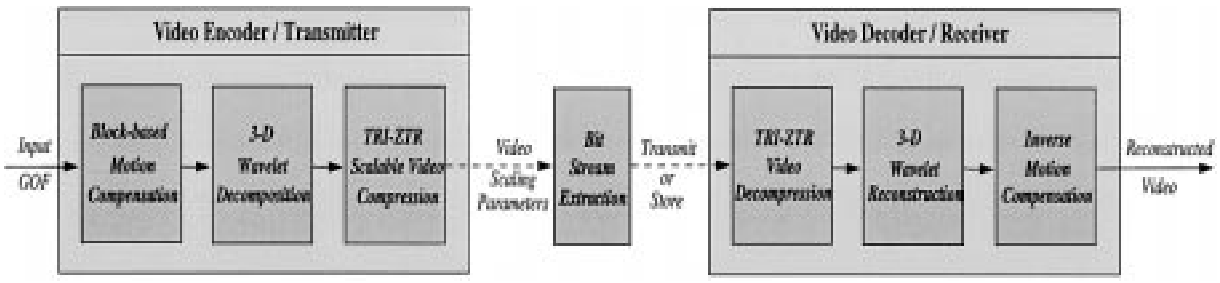

Video Encoder / Transmitter
Input GOF
Block-based Motion Compensation
3-D Wavelet Decomposition
TRI-ZTR Scalable Video Compression
Video Scaling Parameters
Bit Stream Extraction
Transmit or Store
Video Decoder / Receiver
TRI-ZTR Video Decompression
3-D Wavelet Reconstruction
Inverse Motion Compensation
Reconstructed Video

Fig. 1. Overview of the proposed scalable video encoder and decoder.

the idea of using a three-dimensional (3-D) subband/wavelet coding strategy has been implemented in a number of video compression systems. For example, Podilchuk *et al.* [17] show that a 3-D subband coder, when combined with geometric vector quantization, can produce good compression performance at low bit rates. A similar 3-D structure is also employed by Chen and Pearlman [2] in which they extend the zerotree method by Said and Pearlman [18] (an improved version of Shapiro's original work [21], [22]) to coding video sequences. A number of techniques for incorporating motion estimation and compensation for video coding have also been investigated. Ohm [14] proposes a method to perform motion compensation prior to temporal subband decomposition. On the other hand, Ngan *et al.* [13] propose to first perform 3-D wavelet decomposition, and then estimate (and classify) the motion information in the wavelet domain. Taubman and Zakhor [23], [24] employ a global motion compensation scheme which accounts for camera panning motion in their 3-D subband structure for video compression. In fact, they are also some of the pioneers in developing a highly scalable video coding system for medium and high bit-rate applications.

In this paper, we propose a motion-compensated 3-D wavelet video coding system[2] which is scalable and suitable for very low bit-rate applications, such as videoconferencing and telephony, and operates at around 10–30 Kbits/s. It is capable of supporting both multiresolution and multirate video scalability with very fine granularity, by employing the concept of layered/progressive coding together with the idea of embedded resolution block coding using a new data structure called *tri-zerotrees* (TRI-ZTR). The proposed video coding scheme is an extension of the two-dimensional (2-D) zero tree proposed by Shapiro [21], [22], and is related to the 3-D zero tree coding by Chen and Pearlman [2]. The main contributions of this paper are in the following three areas: we employ a different wavelet-packet structure which further decomposes the higher frequency subbands with the aim of better preserving details at a given bit rate [26]; we propose an embedded resolution block coding method using resolution flags to provide multiresolution video scalability in addition to multirate scalability; and we introduce a rearrangement scheme to minimize the bit overhead needed to achieve video scalability for operation in very low bit-rate environments.

Fig. 1 gives a general overview of the proposed scalable video codec. The stream of incoming video frames is first partitioned into distinct groups of frames (GOF's) of $\mathcal{F}$ frames each, where $\mathcal{F} = 2^k$, $k = 0, 1, 2, \cdots$, for easy processing. The issue of choosing an appropriate size $\mathcal{F}$ will be discussed in Section V. Stage 1 of the encoder aims to exploit the temporal correlation within a GOF with respect to a reference frame by means of a fast block-based motion compensation technique [29]. In Stage 2, this motion-compensated GOF is then transformed by using a 3-D separable hierarchical wavelet (packet) decomposition framework. Finally in Stage 3, a new modified TRI-ZTR data structure is proposed to effectively encode the GOF into an embedded and scalable compressed video bit stream. To achieve multiresolution scalability, we introduce the idea of *resolution block coding* by means of encoding certain partitioning information into the bit stream. Partial bit stream extraction is carried out based on a given set of video scaling parameters such as bit rate, spatial resolution, and frame rate. The new downscaled bit stream is then transmitted to the decoder for reconstruction. It is clear that the decoder essentially performs the inverse processes of the three main stages of the encoding part, but in the reverse order of processing. Finally, it is also worth noting that the received video bit stream can be stored at the decoder, and then further downscaled, if necessary.

The rest of the paper is organized as follows. Sections II and III discuss Stages 1 and 2 of the video encoder, respectively. We also explain the formation of a TRI-ZTR, and show how it relates to the proposed 3-D wavelet-packet structure. Section IV considers Stage 3, which forms a major part of this paper. Here, we explain in detail how a fully embedded and scalable compressed video bit stream can be generated via primary and secondary passes, propose a new rearrangement scheme, and a method to include resolution block coding. We also probe into the organization of the bit stream to investigate how multiscalability features can be incorporated into a single compressed bit stream. In Section V, we show precisely how unique subsets can be extracted by the decoder to support various video scaling parameters. A performance analysis and comparison of results with the ITU-T H.263 video coding standard are presented in Section VI. Finally, Section VII summarizes the work and concludes the paper.

II. Fast Block-Based Motion Compensation/Registration

This section describes Stage 1 of the video encoder which aims to exploit the temporal correlation of the frames within a GOF before they are decomposed by means of a dyadic

[2]We first extended Shapiro's zerotrees [21], [22] to 3-D scalable video coding in [27], and later to a 3-D structure with motion registration in [28].

wavelet transform. This is done here by performing block-based (local) motion compensation/registration prior to 3-D wavelet (packet) decomposition.

In a conventional hybrid motion-compensated and transform coding scheme (such as MPEG [10] and H.263 [7]), block motion vectors between frames are estimated, and the motion-compensated frame difference is transformed and coded (see, e.g., [7] and [10] for more details). In the 3-D motion-compensated scheme, however, there are no difference frames. Here, the first frame f_1 in a GOF is used as a reference frame, and succeeding frames $f_n, n = 2, 3, \cdots, \mathcal{F}$ in the GOF are then "mapped/registered" with respect to the reference frame f_1 by estimating a set of block motion vectors for each frame. The expectation is that the resulting motion-compensated GOF is better correlated temporally, and this leads to high-energy compaction when the GOF is decomposed temporally in the next stage (i.e., most of the signal energy is concentrated in the lowest frequency temporal subband). However, if the motion registration process is not perfect, residual error energy is obtained in the higher frequency temporal subbands which needs to be coded.

Many different methods can be used to perform the above motion registration process. We employed a three-parameter motion model (similar to [9]) in [28] to compensate for both camera zooming and local object motion. This scheme, which is more complex than using a simple translational model, was found suitable for exploiting large motion with camera zooming, such as that found in the standard Table Tennis sequence. However, the video codec being considered here targets very low bit-rate applications such as videoconferencing and telephony. As these usually have small local movements without camera zoom, we employed a simpler and faster block-based motion compensation algorithm. This algorithm is based on an "*unrestricted center-biased diamond search*" (UCBDS) method [29] to estimate the block motion vectors. It has a compact diamond-shaped search pattern, and uses a center-biased search strategy which is particularly efficient for finding small motion vectors typically found in videoconferencing sequences. Simulations [29] have shown that UCBDS can give up to 31% speed improvement over the fast four-step search proposed by Po and Ma [16], while achieving a comparable or better prediction accuracy. In this work, we use a search window with a maximum range of ± 8 pixels, 32×32 blocks, and the estimated motion vectors are coded using an adaptive Huffman coder.

An important aspect of a motion compensated 3-D coding scheme is invertibility. This is the ability to perfectly reconstruct a GOF (when no quantization is performed) after the frames have been motion compensated. This issue has been investigated by Ohm [14], and Taubman and Zakhor [23], [24] for a scheme with global compensation. The block-based motion compensation scheme used here, however, is noninvertible. As a result, this scheme may introduce artifacts in the form of "block overlaps" and "block holes" when inverse motion compensation is performed at the decoder. Our simulations showed that these artifacts are generally not objectionable for low-motion sequences typical of the videoconferencing scenes being considered here. However, these artifacts do increase with the amount of motion, and the degree of visual degradation can become more pronounced. For completeness, we note that this problem of noninvertibility can potentially be solved by using an additional "error frame." To do so, the encoder would first perform a forward motion compensation, and then follow this by an inverse motion-compensation (as in the decoder). The difference between the inverse motion compensated frame and the actual frame would be the "error frame" which can be coded separately. However, coding the error frames requires additional bits, and introduces more complexity in ensuring a fully embedded and scalable bit stream. As a result, we found that this method is not very suitable for the proposed very low bit-rate scalable video codec. On the other hand, the conventional hybrid motion-compensated coding scheme also introduces a noticeable "blocking" artifact at very low bit rates. To better illustrate these types of artifacts, a comparison is shown in Section VI.

III. 3-D Wavelet (Packet) Framework and Formation of TRI-ZTR

Transform coders perform an energy compaction which allows coding of data with fewer bits. Currently, the discrete cosine transform (DCT) has been adopted in all international image and video compression standards, such as JPEG [15], MPEG [10], [12], H.261 [25], and H.263 [7]. However, over the past decade, the discrete wavelet transform (DWT) [1] has proven to be an excellent transform for data compression. Some of the attractive features of wavelets, such as good time–frequency localization and multiscale representation of signals, have contributed to its increasing popularity. In this paper, we employ a separable 3-D wavelet (packet) decomposition (e.g., [2], [14], [17], [24], [28]) to transform a given GOF into the wavelet domain. We first perform temporal filtering, followed by spatial decomposition of each of the resulting temporal subbands. In the next three subsections, we will describe the construction of the proposed 3-D framework via spatiotemporal filtering, and then establish the intersubband relationships and the formation of TRI-ZTR within such a 3-D structure.

A. Dyadic Wavelet Temporal Decomposition

As explained in Section II, temporal decomposition is performed on a motion-compensated GOF. Fig. 2(a) shows a GOF with $\mathcal{F} = 4$ frames, which is decomposed along the temporal dimension into $\mathcal{F} = 4$ temporally filtered frames by means of a conventional octave-bandwidth (dyadic) wavelet decomposition. In the decomposition process, the lowest frequency subband at each level is recursively decomposed and critically subsampled. An $\mathcal{L}_T = 2$ level temporal decomposition is illustrated in Fig. 2(b), where level $t = 2$ represents the lowest frequency. In general, we let $\mathcal{L}_T = \log_2 \mathcal{F}$ so that the coarsest (lowest frequency) temporal subband is comprised of only one temporally filtered frame. We used the Daubechies wavelet filter of length 4 [5], [6], and by using a periodic (wrap-around) data extension at the boundaries of the GOF, perfect reconstruction is possible. For the case when $\mathcal{F} = 2$,

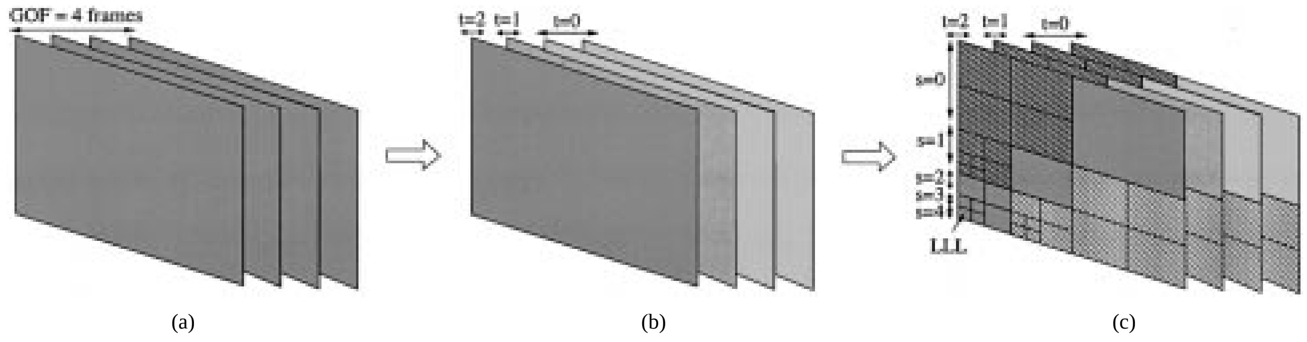


Fig. 2. Proposed 3-D wavelet (packet) framework: (a) motion-compensated GOF, (b) temporally decomposed GOF using dyadic wavelets, and (c) followed by spatial decomposition using separable wavelet packet.

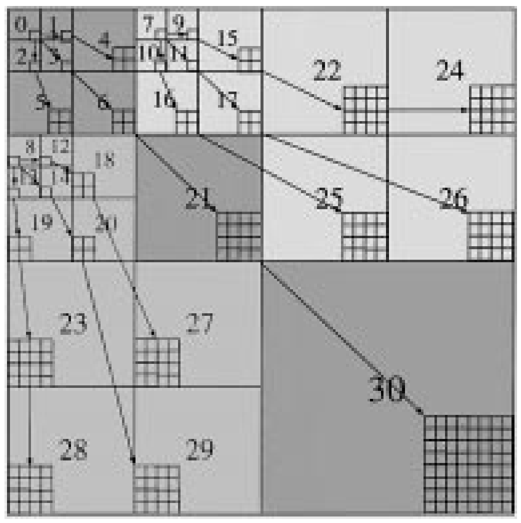


Fig. 3. Cross-sectional template of a 3-D wavelet (packet) framework. The three different shaded regions represent three independent pyramidal tree structures, while the numbered subbands denote the scanning sequence. The arrows define the parent–child relationships of the trees rooted at subbands 0, 7, and 8.

the simple two-tap Haar filter [6], which essentially evaluates the sum and difference between the frames, is used.

B. Wavelet-Packet-Based Spatial Decomposition

In order to generate a 3-D wavelet (packet) framework as depicted in Fig. 2(c), a separable one-dimensional (1-D) wavelet-packet decomposition is performed along the horizontal and vertical dimensions of a temporally filtered frame. Fig. 3 illustrates the resulting wavelet-packet structure of one temporally filtered frame. To obtain this, a separable 2-D dyadic wavelet transform is first performed on the frame. Subsequently, some of the higher frequency subbands are further decomposed to yield the wavelet-packet structure shown in Fig. 3. The subbands are numbered from zero to 30, with zero representing the coarsest subband. The numbers also indicate the scanning order that is followed during the encoding process. In our simulations, we choose the number of spatial scales $\mathcal{L}_S$ such that the coarsest subband[3] 0 is approximately 8×8. For spatial decomposition, we use the biorthogonal 9–7 spline wavelets [1], [4] as we found that these filter banks gave comparatively fewer ringing artifacts at low bit rates [1]. Since the filters are symmetric, we employ a symmetric (reflective) data extension scheme at the boundaries.

[3]For a CIF-based format video, the size of the coarsest subband 0 is (11×9).

C. Intersubband Relationships and Formation of TRI-ZTR

An interesting characteristic of recursive subband/wavelet decomposition is the formation of spatial orientation trees with multiscale support. This idea was first exploited by Lewis and Knowles [11], and Shapiro [21], [22] for still image coding. In this paper, we extend Shapiro's zerotrees from 2-D to 3-D (see also [2] and [27]) for video coding. We also use the wavelet-packet structure in Fig. 3 which is different from the conventional dyadic subband structure. This is motivated by the desire to selectively retain more of the higher frequency coefficients, which in turn better preserve edge information at a given bit rate [26]. As a result, the intersubband relationships and the definition of an orientational tree are slightly different from the previous work. In Fig. 3, each small square denotes a wavelet coefficient, and the arrows indicate the parent–child relationships that are defined among these coefficients. Note that three independent trees are formed, rooted at subbands 0 (diagonal tree), 7 (vertical tree), and 8 (horizontal tree). Each temporally decomposed frame consists of these three trees. In the conventional dyadic wavelet transform structure, each parent coefficient in a given subband (except for the finest) is defined to have four children at the next finer self-similar subband. In the wavelet-packet structure shown in Fig. 3, parent–child relationships can be constructed as above, except for subbands 22 and 23. This deviation from the conventional definition is a consequence of further decomposing the higher frequency subbands. Nevertheless, these tree structures generally conform well with the expectation that, on the average, the children nodes have less energy than their parent; this decreasing-energy property of a tree is critical for the frequent formation of a zerotree (as defined by Shapiro [21], [22]). Any of the three zerotrees that are possible with the wavelet-packet decomposition considered here is called a *tri-zerotree* (TRI-ZTR).

The 3-D extension of a TRI-ZTR can now be described by including the temporal dimension. Recall from Fig. 2(b) that the temporally filtered frames are ordered from the coarsest $(t = 2)$ to the finest $(t = 0)$ temporal scale. Within a temporal scale, the frames are merely arranged in the order they were computed during the temporal decomposition. While scanning the GOF for TRI-ZTR's, we follow the 2-D scanning sequence shown in Fig. 3, starting from the coarsest temporal frame, and proceed towards the finest. A 3-D TRI-ZTR can be formed rooted at some spatiotemporal location in a GOF if: 1) a 2-D TRI-ZTR is formed at the given spatial location in the temporal frame being considered, and 2) 2-D TRI-ZTR's are formed in every subsequent temporal frame, with their root nodes at the given spatial location. This collection of 2-D TRI-ZTR's forms the 3-D TRI-ZTR at the given location. This extension to the temporal dimension is reasonable since the wavelet transform is performed on a motion-compensated GOF, and we can expect the energy compaction property of the transform to concentrate most of the energy in the coarser temporal frames. In contrast to the 3-D TRI-ZTR's, in [2], the definition of a 2-D tree is extended by considering the temporal octave scales, and constitutes a direct extension of the 2-D parent–child relationship to 3-D (see [2] for more details). Using the motion-compensated GOF, the 3-D TRI-ZTR's have the potential to form large trees, and hence to convey a large number of predictably insignificant coefficients to the decoder. As the likelihood of forming a TRI-ZTR is high at the early stages of the encoding process (which corresponds to the first few layers with large thresholds), the proposed video codec can operate well at very low bit rates.

IV. Scalable Video Coding via Tri-Zerotrees

Progressive transmission is very useful in many practical applications such as large image/video database browsing. In such situations, a user first sees a coarse version of an image reconstructed from very few bits, and as more of the bit stream is received, the image quality is successively refined until the end of the bit stream is reached. This allows fast retrieval of an intelligible image, and gives a user the option to terminate the transmission at any time if the image is found to be irrelevant. On the other hand, a nonprogressive transmission will require the entire bit stream to be received before the image is viewable. A central idea of progressive transmission is successive approximation of the image's pixel values (or, the magnitudes of the transform coefficients)—similar in spirit to the bit-plane coding strategy [25]. One of the most successful ideas for a progressive image coder was introduced by Shapiro with his *embedded zerotrees of wavelet coefficients* (EZW) algorithm [21], [22], and this has demonstrated excellent rate-distortion performance for 2-D still image compression. An important property of EZW is that it generates a multirate scalable compressed bit stream. A few enhancements to EZW have been incorporated by Tham [26], Sampson *et al.* [20], and Said and Pearlman [18] who later developed a fast and efficient image codec in [19].

The main challenges of a progressive coding approach which employs successive approximation can be described by the following two problems. The first problem is to efficiently select the more important wavelet coefficients, and then code them earlier than the others. This is motivated by the fact that at the early stage of the encoding process (which corresponds to very low bit-rate coding), the bits have to be utilized economically for coding as many of the more important coefficients as possible, in order to ensure fast recognition of an image upon receiving a minimum number of bits. The second problem is to successively refine the values of the coefficients. Furthermore, a scalable video coding system will also require a method to explicitly partition the data for partial bit stream extraction. This becomes difficult at very low bit rates since the bit budget available for coding both the data as well as the partitioning information will present a significant constraint.

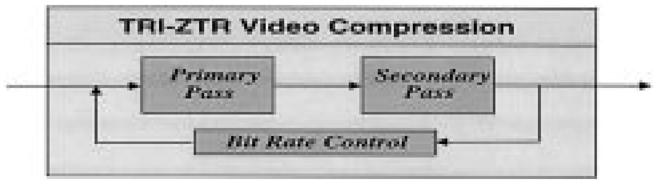


Fig. 4. Overview of TRI-ZTR scalable video encoder which consists of a primary pass, a secondary pass, and a precise bit-rate controller.

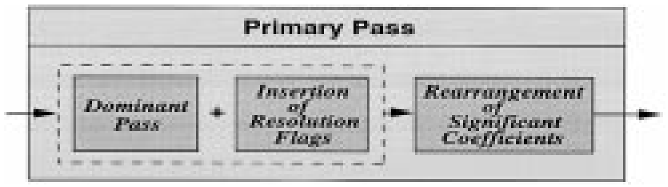

Primary Pass
Dominant Pass
+
Insertion of Resolution Flags
Rearrangement of Significant Coefficients

Fig. 5. Primary pass is made up of three steps: dominant pass, insertion of resolution flags, and rearrangement of significant coefficients.

In this section, we will detail how the proposed TRI-ZTR video codec attempts to address the above issues in an efficient manner for a very low bit-rate environment. A successive refinement/layered coding strategy is used, where the more important wavelet coefficients are selected first and coded in multiple embedded stages—each stage adding another bit of precision to their magnitude. At each stage, two main passes are performed, namely, a *primary pass* and a *secondary pass*, which address the first and second problem, respectively. Explicit partitioning information is also encoded to achieve multiresolution scalability with minimal bit overhead. Fig. 4 presents an overview of the proposed TRI-ZTR video encoder with precise bit rate control. The next two subsections will explain each of the two main passes in detail. As some of the ideas employed here are similar to the original EZW, only a brief outline will be given, when there is no risk of confusion.

A. Primary Pass

The main purpose of a primary pass is to perform effective selection of the more important wavelet coefficients, and then encode the information in an economical manner. A primary pass (as depicted in Fig. 5) consists of three key steps: a dominant pass, insertion of resolution flags, and rearrangement of significant coefficients. To begin, we take the "more important coefficients" to be those with larger absolute values, i.e., those containing more signal information. Hence, by sending the larger coefficients earlier in the bit stream, a lower distortion can be achieved at a particular bit rate. This, however, requires the coefficients to be prioritized in terms of magnitude before the coding is carried out, and this process can incur a large overhead to code the positions of the coefficients. The zerotrees [21], [22] implicitly represent this positional information by exploiting the intersubband relationships and the tree structure to reduce this overhead. This idea is incorporated in our work as the first step (called dominant pass) of a primary pass. When it is combined with the second step which inserts resolution flags, we can also provide multiresolution scalability. The third step, which involves rearrangement of significant coefficients, ensures that multiresolution scalability incurs only a small overhead.

1) Dominant Pass: The main objective of a dominant pass (similar to [21] and [22]) is to identify important wavelet coefficients in descending order of magnitude. This is done on each *GOF block* consisting of $\mathcal{F}$ frames. Two different lists, namely, a dominant list and a subordinate list, are maintained throughout the encoding process. Initially, the dominant list contains all of the wavelet coefficients in a GOF, which are ordered according to a predetermined scanning sequence which was described in Section III-C, and the subordinate list is empty. As in [21] and [22], the magnitude of each coefficient c in the dominant list is compared with a series of decreasing positive thresholds T_i where $i = 1, 2, \cdots$ denotes the ith pass. In all of our simulations, we chose the initial threshold $T_1 = \max\{|c|\} - 1$. A coefficient c is considered *significant* (i.e., important) if $|c| > T_1$, and *insignificant* otherwise. If c is significant, its sign (either positive or negative) is encoded; it is then removed from the dominant list and appended to the subordinate list. At the end of the dominant pass, the threshold is halved (i.e., $T_i = \frac{1}{2}T_{i-1}$). As with zerotrees, a TRI-ZTR also aims to efficiently encode the positions of significant coefficients in a GOF by forming spatiotemporal trees to indicate predictably insignificant coefficients at finer scales. A TRI-ZTR is identified if the root node itself and all of the descendant nodes are insignificant with respect to the current threshold. On the other hand, if the root node is insignificant but one or more of the descendant nodes are significant, then an “isolated zero” is encoded. In essence, a dominant pass will produce four possible symbols (i.e., positive, negative, TRI-ZTR root, isolated zero) to indicate the signs and positions of significant coefficients.

2) Insertion of Resolution Flags During a Dominant Pass: In addition to multirate scalability, the proposed video codec also provides for multiresolution scalability. By this, we mean the ability to trade both spatial resolution and frame rate for bit rate.[4] However, only spatial resolutions and frame rates which are 2^{-n} of the original resolution are supported, where n is some positive integer. As an example, suppose that we want to allow the decoder to select from, say, a maximum of $\mathcal{R}_s = 3$ different possible spatial resolutions (i.e., either full, half, or quarter size video). To achieve this capability efficiently, the compressed bit stream has to be partitioned into *resolution blocks* in such a manner that different display resolutions can be chosen by decoding only the pertinent partitions of the bit stream. These resolution blocks are constructed by inserting $\mathcal{R}_s = 3$ *resolution flags* (RFG’s) in each temporally filtered

[4] As will be explained later, this feature also allows decompression hardware scalability when the spatial resolution and/or the frame rate is reduced.

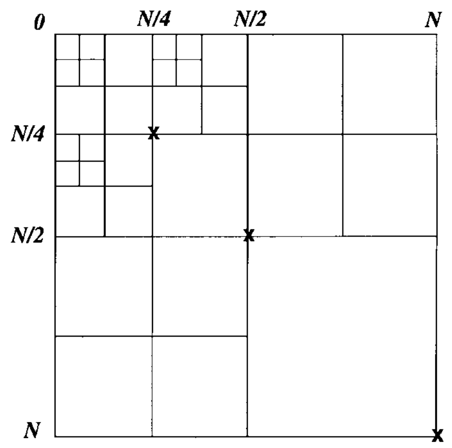


Fig. 6. Template indicating the positions (as marked by the crosses) where $\mathcal{R}_s = 3$ resolution flags, are inserted for multiresolution video scalability.

frame during a dominant pass. Fig. 6 illustrates the positions where the RFG’s are inserted. Note also that the RFG’s are inserted at the end of the predetermined conventional octave scales, thus giving rise to octave spatial resolution scalability. These RFG symbols are then encoded, together with the other four possible symbols in a dominant pass, using an adaptive model arithmetic coder [30]. Furthermore, as we encode an RFG symbol into the bit stream, we also insert a special symbol into the subordinate list to demarcate the significant coefficients into their respective resolution blocks. This will be seen to be useful for rearranging the significant coefficients in the subordinate list (Section IV-A3) as well as for inserting RFG’s during the secondary pass (Section IV-B2).

From another viewpoint, let us now consider Fig. 7 to understand how the compressed bit stream is being partitioned by the RFG symbols. The shaded region represents the arithmetic encoded symbols generated for a given GOF block in *one* dominant pass, and the vertical lines denote the encoded RFG symbols. It is evident that the portion of bit stream between two vertical lines corresponds to a resolution block at a particular spatiotemporal scale. A group of $\mathcal{R}_s$ consecutive resolution blocks will constitute a *frame block.* Since there are $\mathcal{R}_s$ RFG’s for each temporally filtered frame in a GOF, we have $\mathcal{R}_s \times \mathcal{F}$ vertical lines, or equivalently, that many resolution blocks within the shaded region. We note that the resolution blocks not only segment the bit stream into distinct spatial resolution scales, but implicitly into unique temporal resolution scales as well. By knowing this correspondence between the partitions (unique resolution blocks) in the bit stream and the spatiotemporal scales, we can select different video resolution scaling parameters by extracting only the relevant partitions of the bit stream.

3) Rearrangement of Significant Coefficients in Subordinate List: As mentioned earlier, the coefficients found significant with respect to the current threshold are moved to the subordinate list. The manner in which they are incorporated into the existing subordinate list influences the number of RFG symbols that need to be coded during the second step of the secondary pass. Naturally, for coding efficiency, this number must be kept to a minimum. Assume that the current sub-

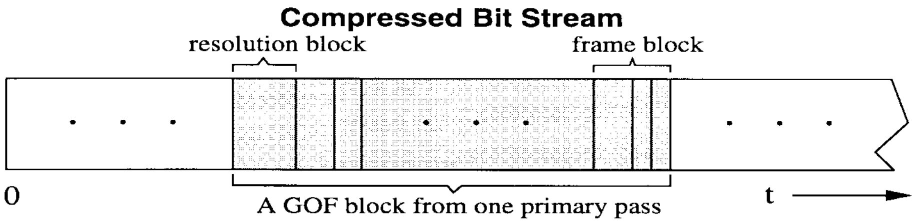


Fig. 7. Shaded region represents the portion of a compressed bit stream generated in one primary pass. The encoded resolution flags (as denoted by the vertical lines) indicate how distinct resolution blocks and frame blocks are defined.

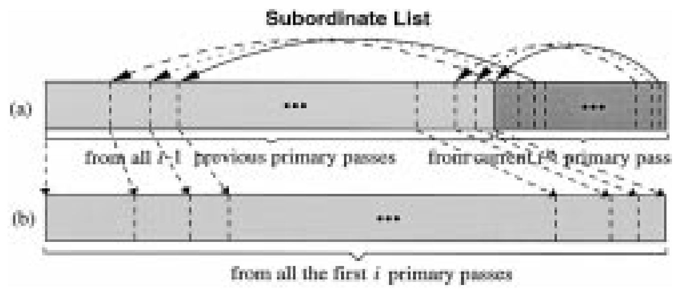

Subordinate List
(a)
(b)
from all i-1 previous primary passes
from current i-th primary pass
from all the first i primary passes

Fig. 8. Snapshot of the subordinate list (a) just before and (b) just after the rearrangement step in a primary pass.

ordinate list already contains significant coefficients that are segregated according to their resolution blocks. When a new set of significant coefficients are found during a dominant pass, the coefficients are appended to the end of the subordinate list, as depicted in Fig. 8(a). This arrangement is inefficient because the significant coefficients belonging to a particular spatiotemporal scale are now fragmented into noncontiguous blocks in the list, and the number of resolution blocks (RFG's) increases with the number of passes. Such a shortcoming can be overcome by appending the new coefficients in each resolution block to the existing coefficients in the corresponding block of the subordinate list. The rearranged (defragmented) subordinate list is illustrated in Fig. 8(b). It is evident that the number of resolution blocks in the subordinate list now remains unchanged at $\mathcal{R}_s \times \mathcal{F}$ per GOF, independent of the number of passes. An interesting point to note here is that, as the encoding process proceeds with successively smaller thresholds, potentially smaller significant coefficients from coarser scales may be placed ahead of larger coefficients found earlier from the finer scales.

B. Secondary Pass

When the data at the end of a primary pass are transmitted, the decoder will have three pieces of information about all significant coefficients in the subordinate list. First, the decoder knows their signs (either positive or negative) as conveyed by the dominant pass. Second, it can also identify their exact positions by replicating the same scanning sequence used in the encoder. Third, the decoder knows the most significant bit of each new significant coefficient (since their magnitudes are larger than T_i but smaller than $T_{i-1} = 2T_i$), and assigns a zero to all insignificant coefficients. Fig. 9 presents an overview of a secondary pass which consists of three important steps: a subordinate pass, insertion of RFG symbols, and a reordering protocol.

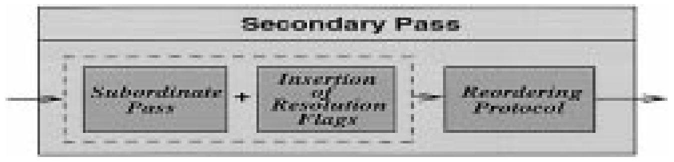


Fig. 9. Secondary pass is made up of three key steps: a subordinate pass, insertion of resolution flags, and a reordering protocol.

1) Subordinate Pass: The main function of a subordinate pass is the same as that of the original EZW [21], [22]. It aims to further refine the precision of all significant coefficients found thus far. At the end of a primary pass, each significant coefficient will have a reconstruction value as can be interpreted by the decoder, and is associated with an uncertainty interval whose length is equal to the current threshold value. The subordinate pass halves this uncertainty interval for each entry in the subordinate list. This is done by transmitting either a "0" or "1" to indicate whether the actual magnitude of each entry lies in the lower or upper half of the (previous) uncertainty interval, respectively. In this process, the quantization interval of the significant coefficients is being refined, and is easily associated with successive approximation of the significant coefficient values.

2) Insertion of Resolution Flags During a Subordinate Pass: Similar in motivation to the second step of a primary pass, the primary goal of this step is to integrate multiresolution scalability into the proposed video codec. To achieve this, we also need to insert appropriate RFG symbols during a subordinate pass to explicitly partition the bit stream[5] into distinct resolution blocks. As the subordinate list is already demarcated into resolution blocks, we can encode the RFG symbols appropriately while refining the coefficients in the list. Altogether, the three different symbols (i.e., "RFG," "0," "1") are encoded using an adaptive model arithmetic coder [30]. In this manner, the uniqueness of each resolution block (and hence, the frame block) is maintained for multiresolution video scalability.

Recall how the third step in a primary pass rearranged the entries in the subordinate list so that all significant coefficients from the same resolution block are grouped in a contiguous block. As mentioned, it is apparent that the cost of encoding the RFG symbols in this step has now been reduced to a fixed number $\mathcal{R}_s \times \mathcal{F}$ [instead of $i \times (\mathcal{R}_s \times \mathcal{F})$], which is independent of the iteration index i. This is a significant improvement for a video codec which operates in a very low bit-rate environment

[5]Note the entire bit stream, which is made up of both the primary and secondary passes, needs to be fully partitioned into distinct resolution blocks.

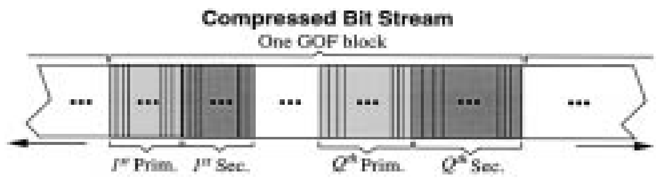


Fig. 10. Typical compressed bit stream which is made up of unique resolution blocks.

since the available bit budget must be shared between the compressed video and the partitioning information.

3) Reordering Protocol: The principal objective of this step is to reorder/prioritize all of the significant coefficients in the subordinate list to attempt to place the more important piece of information[6] earlier in the list. In this way, these entries will be refined first in the next subordinate pass, and this allows obtaining the best possible quality of the reconstructed video at a given bit rate. In [21] and [22], the significant coefficients are reordered according to the following four priority criteria: 1) precision, 2) reconstruction magnitude, 3) scale, and 4) spatial location. A similar approach is also adopted in our proposed video codec, but it has been slightly modified to account for the preservation of unique resolution blocks. Specifically, we confine the reordering by precision and reconstruction magnitude to each of the $\mathcal{R}_s \times \mathcal{F}$ resolution blocks, while adhering to the predetermined scanning sequence. In summary, the reordering protocol used in this work can be described as prioritization with respect to: 1) precision, 2) reconstruction magnitude, 3) temporal scale, 4) spatial scale, and 5) spatial location. Finally, this reordering protocol does not incur any overhead in terms of additional bits.

At the end of this secondary pass, the next primary pass will resume, and these two passes will alternate until a certain target bit rate or distortion level is achieved. This generates a compressed video stream consisting of distinct but embedded resolution blocks which can support both multirate and multiresolution scalability. An example of such a compressed bit stream is depicted in Fig. 10.

V. Video Scalability and Rescalability

In the preceding sections, we discussed the generation of a fully embedded and scalable compressed bit stream for storage/transmission. This section focuses on how *one* compressed bit stream can be manipulated to meet a multitude of display specifications and system requirements, such as bit rate, distortion level, display resolution, frame rate, decompression hardware complexity, and end-to-end coding delay. We call these specifications the *video scaling parameters*. In particular, we concentrate on the issue of partial bit stream extraction, as illustrated in Fig. 1. The next two subsections first detail the supported video scaling parameters, and then give examples to illustrate the degree of video scalability achievable.

[6] Information here provides an indication of how much reduction in distortion is achieved after receiving that part of the coded message.

A. Supported Video Scaling Parameters

The principal idea behind a highly scalable video compression system is to provide an easy means for the decoder to select different substreams from one compressed bit stream *after* it has been generated. A combination of different portions of the bit stream will produce a different version of the video as if it were being generated separately by the encoder. As an illustration, let us consider a video coding system with the following specifications: an original $M \times N$ video sequence which is encoded with a target frame rate of F_R frames per second (fps); a GOF block size of $\mathcal{F} = 2^n$ frames which are decomposed into $\mathcal{L}_T = n$ temporal levels; a maximum of $\mathcal{R}_s$ spatial scales; and a target bit rate of Ψ bits per second. Assume further that there are $\mathcal{Q}$ rounds of primary and secondary passes (i.e., $\mathcal{Q}$ quantization layers). The compressed bit stream is portrayed in Fig. 10.

1) Bit Rate and Distortion Level Scaling Parameters: As the compressed bit stream is fully embedded, we can now scale for different bit rates with an arbitrary granularity. It is evident from Fig. 10 that, if $\mathcal{Q}$ primary and secondary passes are completed, the current GOF block will consist of $2\mathcal{Q} \times (\mathcal{R}_s \times \mathcal{F})$ resolution blocks.[7] Let $\psi^q_{e,f}$ and $\varphi^q_{e,f}$ denote the total number of bits generated for the resolution block in the eth spatial scale of the fth temporally filtered frame of the current GOF[8] during the qth primary and secondary pass, respectively. The target bit rate Ψ can easily be converted to the total bit budget allocated to each GOF as

$$\Psi_{\text{GOF}} = \frac{\mathcal{F}}{F_R}\Psi \quad \text{bits}$$

where Ψ_{GOF} can also be expressed as

$$\Psi_{\text{GOF}} = \sum_{q=1}^{\mathcal{Q}} \sum_{f=1}^{\mathcal{F}} \sum_{e=1}^{\mathcal{R}_s} \left(\psi^q_{e,f} + \varphi^q_{e,f}\right) \quad \text{bits.}$$

In view of the above layered substream hierarchy, different subsets of the original bit stream can be extracted based on a given bit-rate scaling parameter to produce a new compressed bit stream. Suppose that, due to some transmission bandwidth constraints, a receiver can receive no more than 14.4 Kbits/s of data. This translates to $\Psi'_{\text{GOF}} = (\mathcal{F}/F_R) \times 14.4$ Kbits per GOF.[9] In this case, only the first Ψ'_{GOF} Kbits of each GOF block in the original bit stream are extracted for transmission. Specifically

$$\Psi'_{\text{GOF}} = \sum_{q=1}^{\Upsilon} \sum_{f=1}^{\mathcal{F}} \sum_{e=1}^{\mathcal{R}_s} \left(\psi^q_{e,f} + \varphi^q_{e,f}\right) + \Delta_\Psi \quad \text{bits}$$

where $\Upsilon < \mathcal{Q}$ is the index of the maximum possible complete quantization layer, as constrained by Ψ'_{GOF}, that is common

[7] Note, however, that the number of resolution blocks generated with an incomplete $\mathcal{Q}$th layer is between $(2(\mathcal{Q}-1) \times (\mathcal{R}_s \times \mathcal{F}), 2\mathcal{Q} \times (\mathcal{R}_s \times \mathcal{F}))$. To be precise, the last transmitted resolution block in the $\mathcal{Q}$th layer may also be incomplete. Nevertheless, for the sake of simplicity in this example, we assume a complete $\mathcal{Q}$th layer encoding using the available bit budget.

[8] Here, we denote the first resolution block as the block with the coarsest spatial and temporal information.

[9] For simplicity, we neglect other bit overheads such as packet headers and channel protection codes.

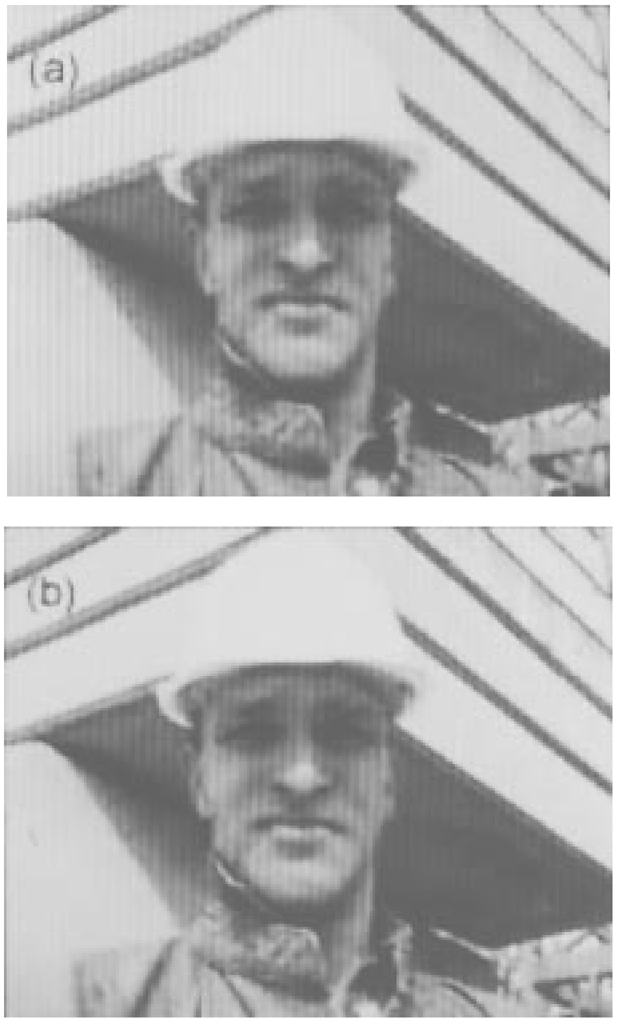


Fig. 11. Foreman frame 12 using (a) TRI-ZTR with conventional octave-scale structure, and (b) with proposed wavelet-packet structure at the same bit rate.

to both the primary and secondary passes. The remaining bits Δ_Ψ can be expressed as shown in the equation at the bottom of the page. The indexes (e_p, f_p) and (e_s, f_s) represent the (resolution block, frame) pair in the $(\Upsilon + 1)$th quantization layer of a primary and secondary pass, respectively, where the substream extraction process has to be terminated. In practice, this possibility of achieving a precise target bit rate is very useful for both constant bit rate (CBR) and variable bit rate (VBR) applications.

In order to support distortion level scalability, a method is proposed in [23] and [24] to include appropriate *distortion tags* in the headers. In our case, a similar approach can be employed by inserting distortion tags at the beginning of each key[10] temporally filtered frame. However, as this process can incur additional bits (which is especially true for a very low bit-rate video codec), the distortion tags should only be inserted after a certain number of quantization layers. This is so since the video reconstructed using only the first few layers is generally of very poor quality for any practical application. However, we have not investigated this issue in detail.

2) Display Resolution and Frame Rate Scaling Parameters: Another important feature of scalable video is *multiresolution scalability*, which refers to both spatial resolution (display frame size) and temporal resolution (frame rate) scalability. Although we can scale for different bit rates with arbitrarily fine granularity, multiresolution scalability is restricted to only "octave granularity," as explained earlier. More precisely, the possible display spatial resolutions are given by

$$d_s = 2^{-2\nu}(M \times N), \qquad \nu \in \{0, 1, \cdots, \mathcal{R}_s - 1\}$$

where $M \times N$ is the original display resolution of the video, while the possible frame rates are

$$f_r = 2^{-t} F_R \quad \text{fps}, \qquad t \in \{0, 1, \cdots, \mathcal{L}_T\}.$$

Recall that a precise bit rate can be obtained independently of the chosen display frame size and/or frame rate. Suppose now that we select a certain combination of display resolutions such that $d_s = 2^{-2d'_s}(M \times N)$, and $f_r = 2^{-f'_r} F_R$, where $0 \le d'_s \le \mathcal{R}_s - 1, 0 \le f'_r \le \mathcal{L}_T$. Then, we can still extract and generate a new compressed bit stream with any arbitrary bit rate, subject to a maximum of

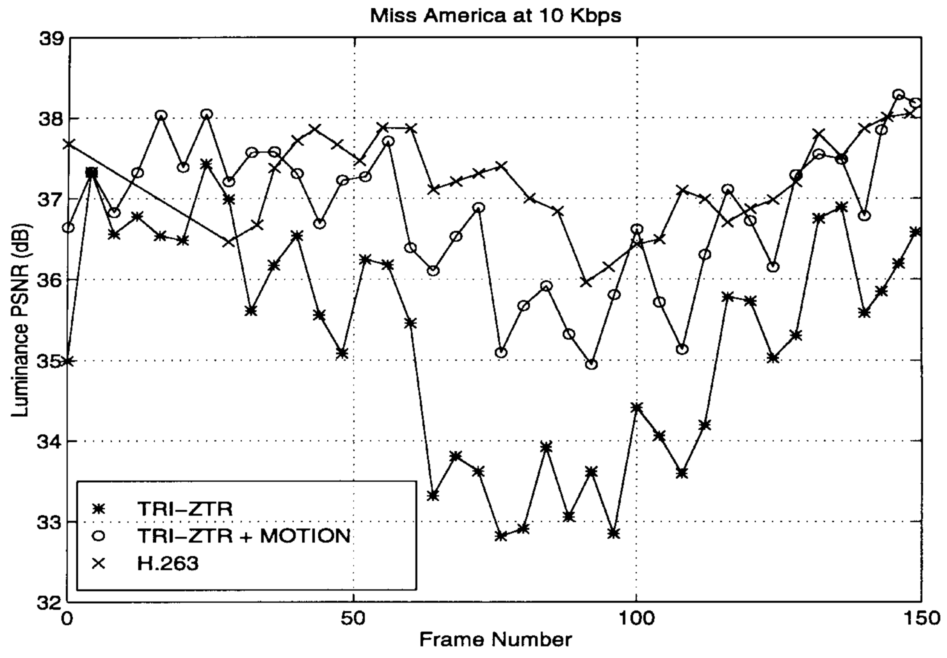
$$\sum_{q=1}^{\mathcal{Q}} \sum_{f=1}^{\mathcal{F}/2^{f'_r}} \sum_{e=1}^{\mathcal{R}_s - d'_s} \left(\psi_{e,f}^{q} + \varphi_{e,f}^{q}\right)$$

bits per GOF block.

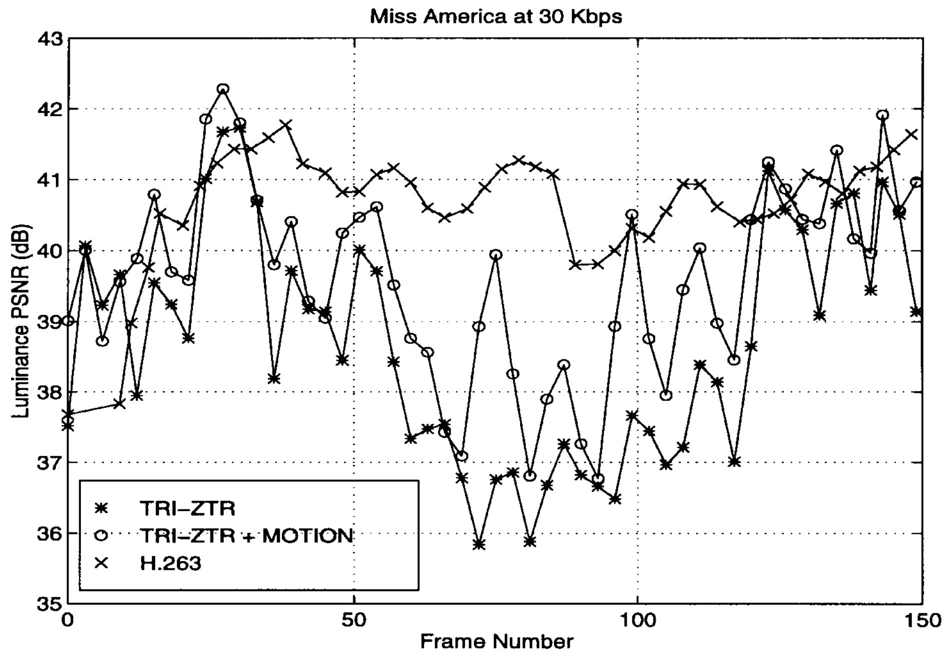
As there is distinct partitioning information in terms of unique resolution blocks in the compressed bit stream, the above video scalability feature represents *explicit* multiresolution scalability. On the other hand, both frame size and frame rate scalability can also be achieved by means of *implicit* multiresolution scalability, which does not involve any explicit partitions. Such an implicit scalability feature is, in fact, inherent in a pyramidal/wavelet framework. However, as there is no partitioning information in an implicit scheme, the decoder will have no indication as to which subsets of the bit stream are needed for a chosen reduced target resolution. In other words, the decoder has to first receive

[10] In this case, the distortion tags can be inserted at the end of each required quantization layer.

$$\Delta_\Psi = \begin{cases} \displaystyle\sum_{f=1}^{f_p} \sum_{e=1}^{e_p} \psi_{e,f}^{\Upsilon+1}, & \text{if the extraction process is truncated in a primary pass} \\ \displaystyle\sum_{f=1}^{\mathcal{F}} \sum_{e=1}^{\mathcal{R}_s} \psi_{e,f}^{\Upsilon+1} + \sum_{f=1}^{f_s} \sum_{e=1}^{e_s} \varphi_{e,f}^{\Upsilon+1}, & \text{if the extraction process is truncated in a secondary pass.} \end{cases}$$



(a)

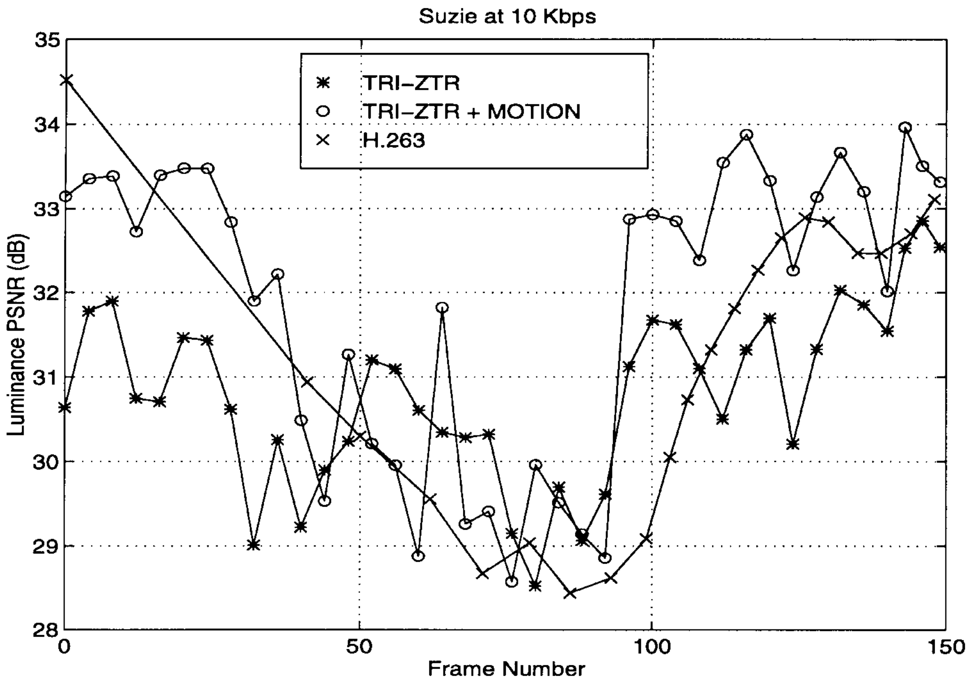


(b)

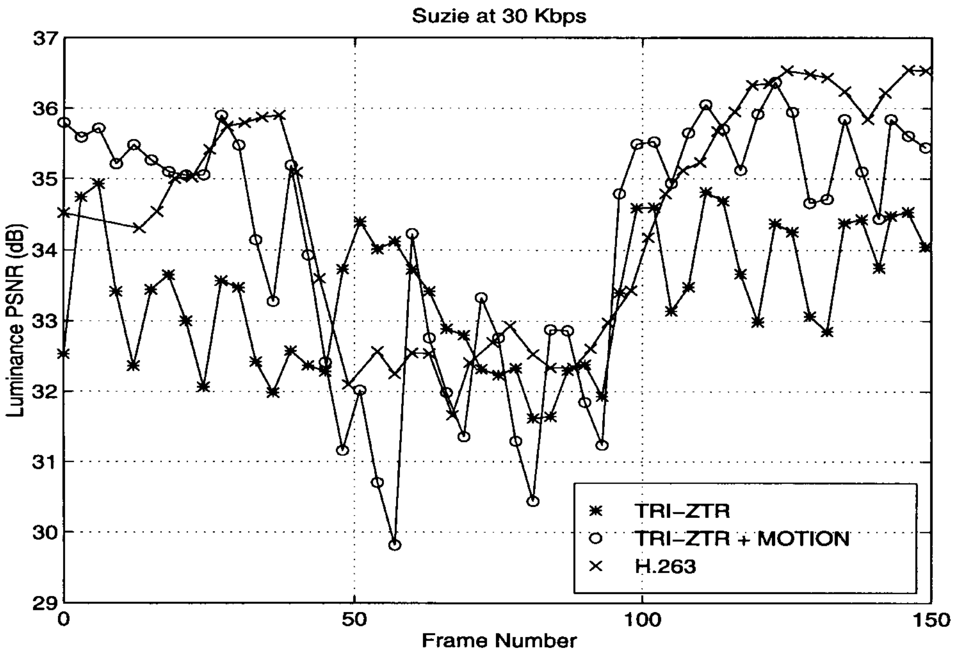
Fig. 12. PSNR comparison for Miss America at 10 and 30 Kbits/s.

the *entire* bit stream which corresponds to the original full-resolution video (although the bit rate can be arbitrarily chosen to support different bandwidths) for correct TRI-ZTR decompression. An inverse DWT of smaller size (both spatially and temporally) can then be performed to reconstruct the reduced resolution video. This implicit approach, however, suffers from two significant drawbacks. First, it results in wasted transmission bandwidth, as explained above. From a rate-distortion perspective, this may result in a poorer quality video at a given bit rate, as a fraction of the received bit stream does not contribute to improving the quality of the video. In contrast, the proposed explicit multiresolution scheme uses the entire new bit stream for reducing the distortion of the video at the chosen resolution. Second, the implicit scheme does not provide the possibility to scale for the decoder's hardware complexity (especially memory requirement), as will be explained next. On the other hand, the explicit scheme requires a network switching node to know which substreams (data packets) to forward on to the different receivers.

3) Decompression Hardware Complexity and End-to-End Coding Delay Parameters: In terms of decompression hardware complexity, the three most important components



(a)



(b)

Fig. 13. PSNR comparison for Suzie at 10 and 30 Kbits/s.

are: 1) the monitor's maximum display resolution, 2) CPU speed/power, and 3) available working memory. The first factor is directly related to the choice of the display frame size. A receiver with a lower resolution monitor can choose to receive spatially scaled down frames. The second factor determines whether the received compressed bit stream can be decompressed in real time for display. It is obvious from Fig. 1 that, by scaling down both the display frame size and frame rate, we can reduce the decoding complexity of each of the three stages, and hence, achieve significant speedup. The third factor that will affect the feasibility for real-time processing is the amount of available working memory. Choosing a lower spatial resolution and/or a smaller GOF block will definitely reduce the amount of required working memory. Finally, we note that the implicit multiresolution scalability approach, as discussed above, will still require the large working memory space needed for decompressing full resolution video, before it can be downscaled in resolution.

Another important property in any real-time (synchronous) application is *interactivity*, which is usually characterized by

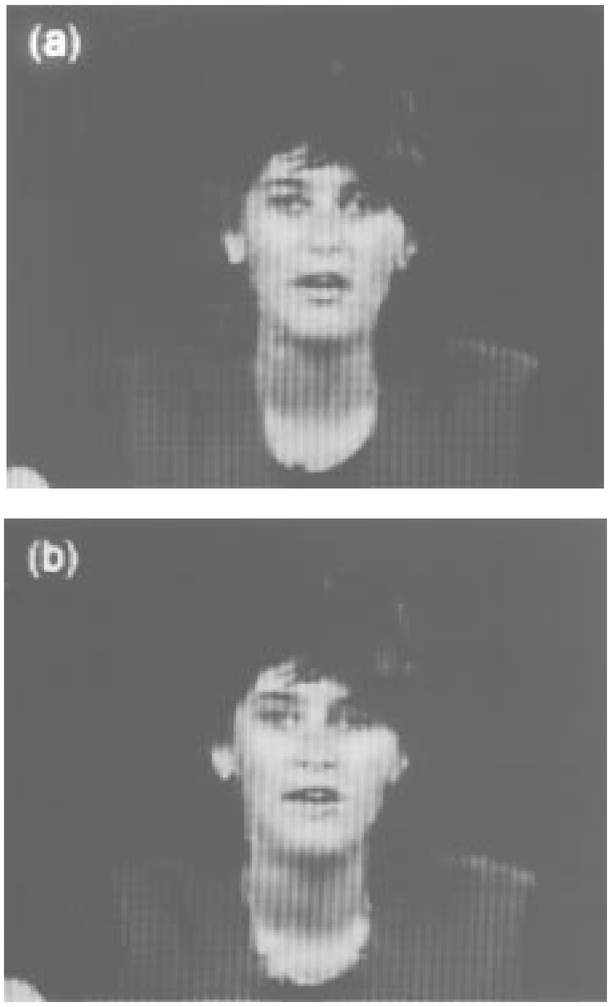


Fig. 14. Miss America frame 64 encoded at 10 Kbits/s by (a) H.263 and (b) TRI-ZTR + MOTION.

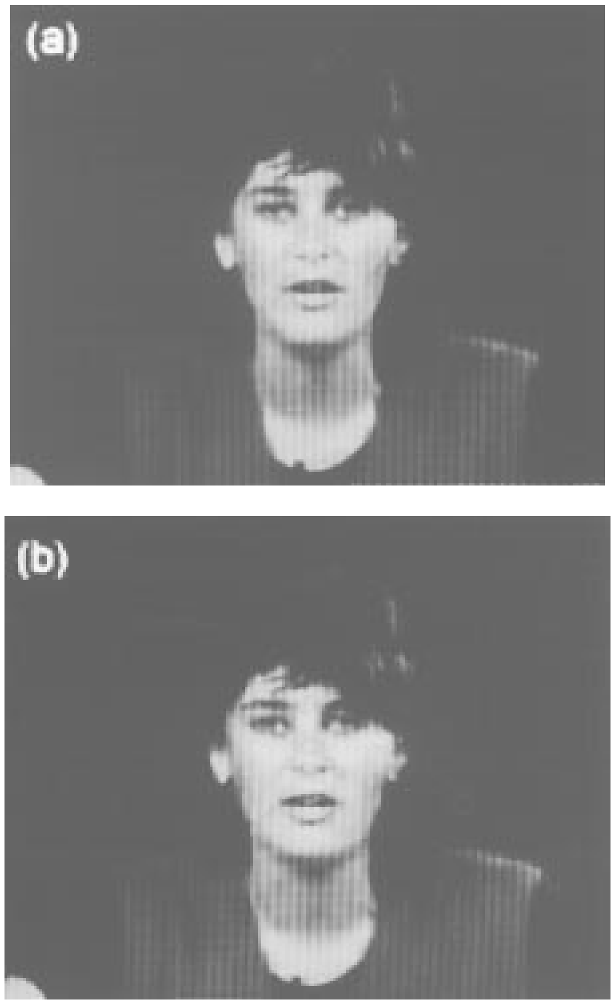


Fig. 15. Miss America frame 60 encoded at 30 kbits/s by (a) H.263 and (b) TRI-ZTR + MOTION.

the interactive response time. As a direct consequence of processing the frames in GOF blocks, it is found [23], [24] that the associated coding delay can be upper bounded by $2\tau_d$, where

$$\tau_d \approx \frac{\mathcal{F}}{F_R} \quad \text{seconds.}$$

Furthermore, it is reported [25] that a coding delay of more than about 300 ms can become quite objectionable for interactive applications. This means that choosing a GOF with $\mathcal{F} = 4$ can be still within the acceptable range of end-to-end delay, if the full frame rate is 24 fps or higher. In general, a larger GOF will have a better compression performance at the expense of coding delay.

B. Degree of Video Scalability

As mentioned above, both multirate and multiresolution scalability can be achieved simultaneously during the bit stream extraction process. Such a feature allows a wide and fine gradation of bit stream scaling parameters. As an example, suppose we encode a 30 fps CIF-format video sequence of size 352 pixels × 288 lines, and use the following parameters: $\mathcal{R}_s = 3$, $\mathcal{F} = 8$, $\mathcal{L}_T = \log_2 \mathcal{F}$. This means that we can have three possible spatial resolutions, four different temporal resolutions, and arbitrary bit rates at the decoder. Table I illustrates some examples of possible combinations of display specifications. Clearly, the examples shown above are by no means exhaustive. With the given input settings, we can have any combinations of these three sets of decoders' display parameters: spatial resolution = $\{352 \times 288, 176 \times 144, 88 \times 72\}$; frame rate = $\{30, 15, 7.5, 3.75\}$; bit rate = {any precise bit rate subject to the maximum allowable bit rate in the bit

TABLE I
EXAMPLES OF COMBINATIONS OF DISPLAY SPECIFICATIONS

Resolution Scaling		*Bit Rate Scaling*
Spatial Resolution (pixels × lines)	Frame Rate (frames/s)	Bit Rate (Kbits/s)
88 × 72	30	8
88 × 72	30	20
88 × 72	15	20
88 × 72	7.5	15
88 × 72	3.75	12
176 × 144	15	30
176 × 144	7.5	35
352 × 288	15	30
352 × 288	30	40

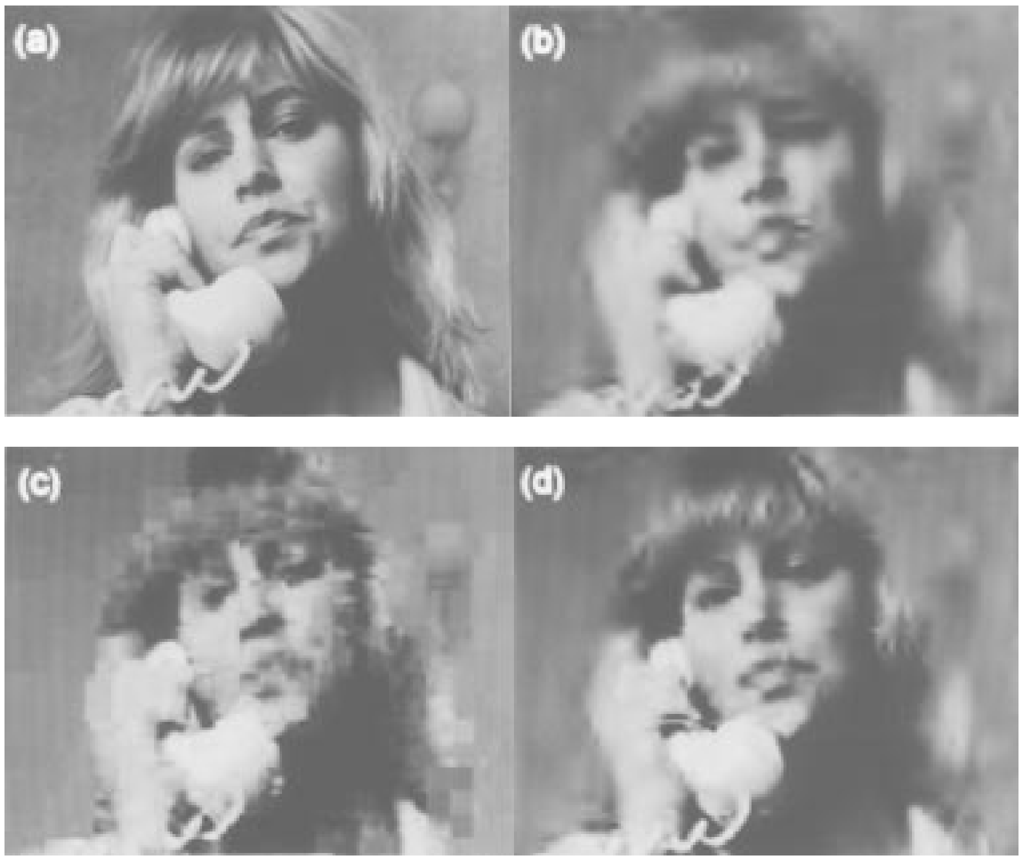


Fig. 16. Suzie frame 84. (a) Original frame, encoded at 10 Kbits/s by (b) TRI-ZTR without motion compensation, (c) H.263, and (d) TRI-ZTR + MOTION.

stream}. Having chosen a certain display spatial resolution and frame rate, the choice of a bit rate will then fully determine the distortion level of the compressed video. Finally, it is noted that a new compressed bit stream, which is extracted from the original bit stream based on some selected scaling parameters, is a fully embedded and scalable bit stream itself. Hence, this provides the possibility to further rescale the bit stream by imposing other scaling parameters to generate an even more downscaled version of a compressed video.

VI. Performance Analysis and Comparison

In Section III, we proposed a wavelet-packet structure to better preserve high-frequency details at a given bit rate, as compared to a conventional octave subband decomposition. The result of doing this is shown in Fig. 11. Better edge information is seen when using the wavelet-packet structure. However, it should be pointed out that this improvement is quite dependent on the frequency content of a frame. The Foreman sequence exhibits sharp edges at the building in the background, and coding gain is obtained when the higher frequency subbands are further decomposed, as in the wavelet-packet structure.

We now present results to compare the TRI-ZTR scalable video coding schemes with the ITU-T H.263 standard which also targets very low bit-rate applications. The H.263 results were produced with the publicly available TMN encoder software [7]. In all of the simulations, the QCIF (176×144) video sequences used have an original frame rate of 30 fps. For a target bit rate of 10 Kbits/s (or 30 Kbits/s), the encoded frame rate is 7.5 fps (or 10 fps) for both methods. For TRI-ZTR, this target frame rate is achieved by discarding every three out of four (or two out of three) frames of the original sequence during the encoding process. In order to obtain a common framework for comparison, we first encoded the sequences using H.263 at a specified frame rate and bit rate. The actual compression achieved by H.263 was then used to precisely specify the inputs to the TRI-ZTR encoder (with GOF = 4 frames). We first compare the objective PSNR results using the Miss America and Suzie sequences. This is followed by subjective comparison, and finally, the results of multiresolution video scalability are presented.

Figs. 12 and 13 show the plots of luminance PSNR versus frame number at 10 and 30 Kbits/s for the Miss America and Suzie sequences, respectively. In each plot, we compare the objective (PSNR) performance of three methods, H.263, TRI-ZTR, and TRI-ZTR + MOTION (TRI-ZTR with motion compensation). It can be seen from Fig. 12 that H.263 generally gives higher PSNR values than both TRI-ZTR methods. However, using TRI-ZTR + MOTION improves the performance of the TRI-ZTR method. For the Suzie sequence, it can be seen from Fig. 13 that the PSNR of TRI-ZTR + MOTION is almost always better than H.263 at 10 Kbits/s, but is generally comparable at 30 Kbits/s. Also, the plots for this sequence show that PSNR of TRI-ZTR is better than TRI-ZTR + MOTION in the middle of the Suzie sequence where there is fairly large and nontranslatory motion. For the

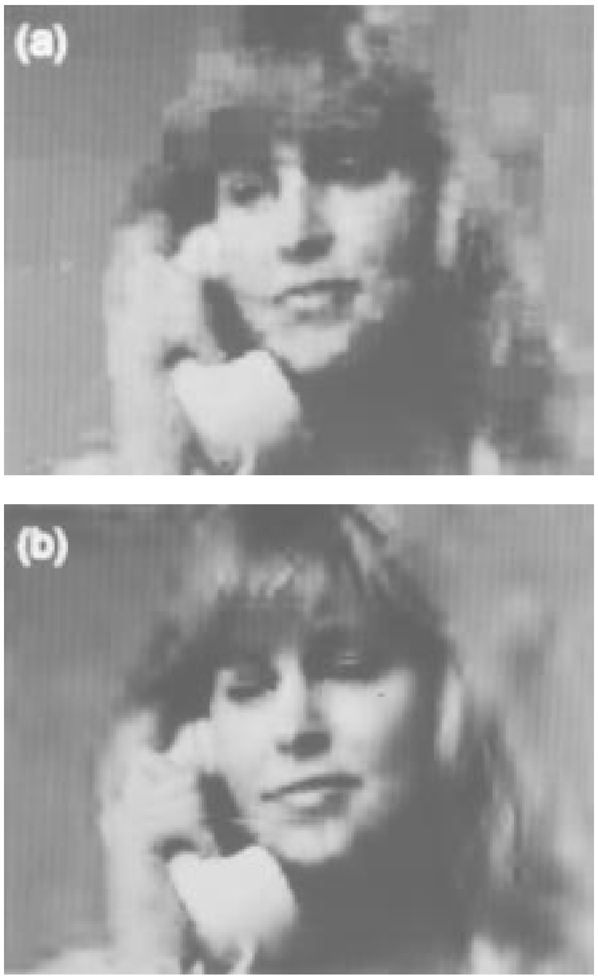


Fig. 17. Suzie frame 105 encoded at 10 kbits/s by (a) H.263, and (b) TRI-ZTR + MOTION.

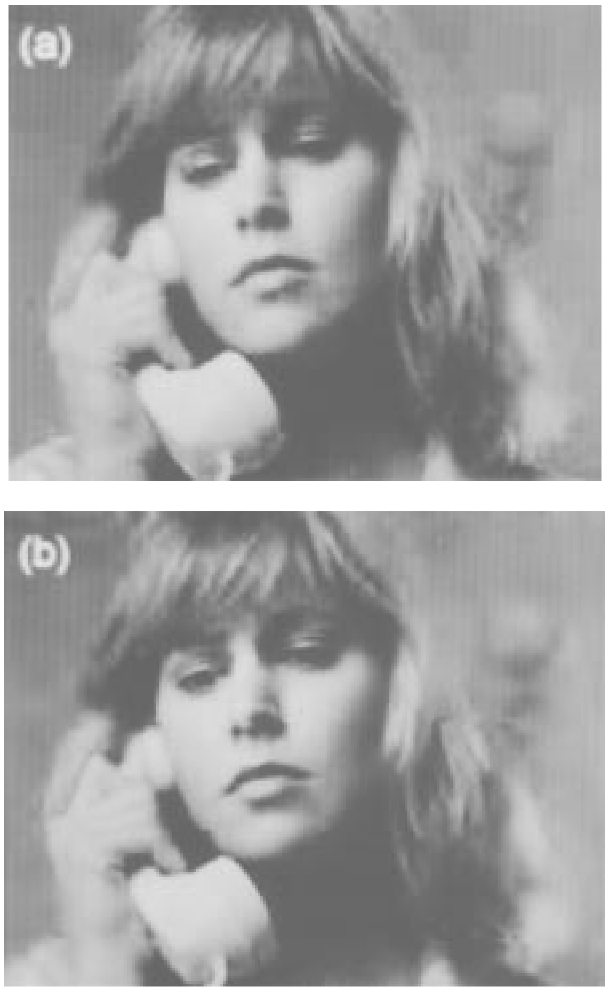


Fig. 18. Suzie frame 30 encoded at 30 kbits/s by (a) H.263, and (b) TRI-ZTR + MOTION.

Miss America sequence, on the other hand, which contains smaller motion, the use of motion compensation is always advantageous in TRI-ZTR's. These results indicate that TRI-ZTR + MOTION is to be preferred for small motion, and TRI-ZTR for large motion. Hence, it is possible that an adaptive scheme which switches between TRI-ZTR and TRI-ZTR + MOTION can keep the quality level more consistent, and we plan to investigate this in the future.

For subjective comparisons between H.263 and the TRI-ZTR methods, images are shown in Figs. 14–18. Fig. 14 illustrates frame 64 of Miss America at 10 Kbits/s. The PSNR value of the H.263 image is about 1 dB higher than the image produced by TRI-ZTR + MOTION. The visual quality of both images is comparable, although the TRI-ZTR + MOTION image shows some ringing artifacts, and appears to have slightly less resolution. At 30 Kbits/s, both methods perform comparably well in terms of visual quality, although the objective PSNR values of TRI-ZTR + MOTION are generally lower than H.263. Comparison images are shown in Fig. 15. For the Suzie sequence, Fig. 16 shows results encoded at 10 Kbits/s. Here, the frame is chosen from a part of the sequence where the motion is large and nontranslational. It is seen that the TRI-ZTR image shows no blocking artifacts, but the resolution is poor. The H.263 result, on the other hand, shows blocking over the entire face, while the TRI-ZTR + MOTION result shows blocking artifacts due to "block overlaps" and "block holes" arising from the motion compensation scheme. The severity of these artifacts depends on the amount of motion. It is seen that TRI-ZTR + MOTION produces the sharpest image of the three methods. Next, in Figs. 17 and 18, we show results on the Suzie sequence when the motion is not as large as in the previous example. At 10 Kbits/s, TRI-ZTR + MOTION produces a sharper and less blocky image than H.263. At 30 Kbits/s, the images from the two methods are comparable.

Finally, Fig. 19 shows the result of multiresolution scalability using the Miss America sequence at 10 Kbits/s, where both the frame size and/or the frame rate has been cut down by a factor of two. In Fig. 19, the top left image is from the sequence coded at full spatial and temporal rate. The top right image is obtained by maintaining the bit rate at 10 Kbits/s, but halving the frame rate. This image appears to be slightly sharper than the previous image. The two smaller images at the bottom of Fig. 19 are also obtained at 10 Kbits/s, but at half the spatial size. Moreover, the image at the bottom right is obtained at half the temporal rate. Both images are sharper than the image at full resolution. Essentially, these examples

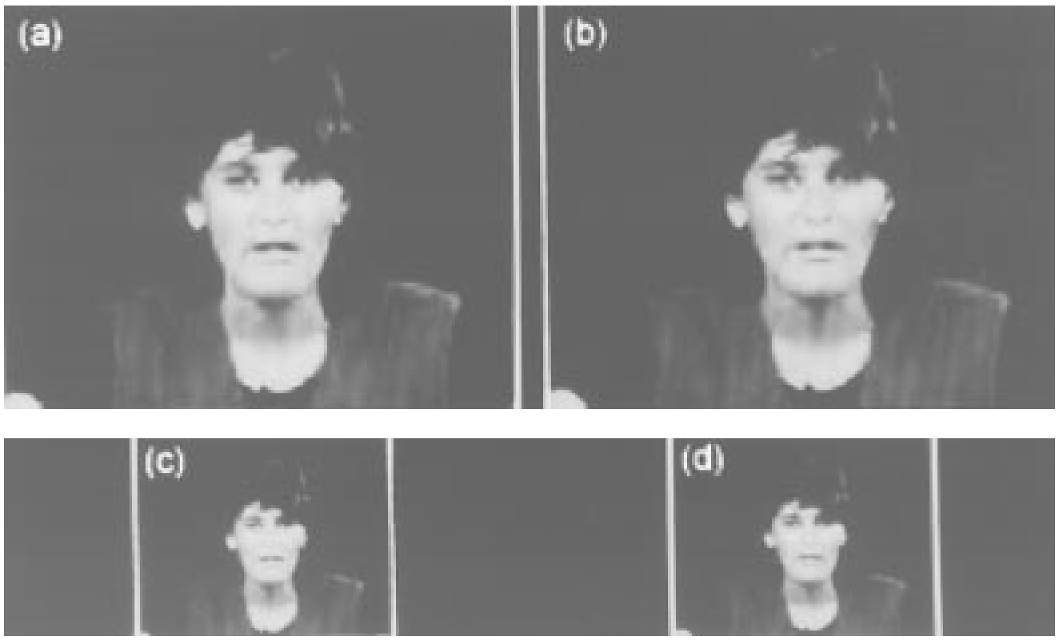


Fig. 19. Multiresolution video scalability: Miss America frame 32 using TRI-ZTR at 10 Kbits/s. (a) Full spatial and full temporal resolution, (b) full spatial but half temporal resolution, (c) half spatial but full temporal resolution, and (d) half spatial and half temporal resolution.

demonstrate that the bit savings arising from reducing the display requirements have been utilized toward improving the quality of the video.

VII. Conclusions

Videoconferencing and telephony applications in very low bit-rate environments (10–30 Kbits/s) present a real challenge for an efficient video compression scheme. The current international standard ITU-T H.263 was proposed to address this need. However, H.263 is inherently nonscalable, both in terms of bit rate and resolution. In this paper, we proposed a TRI-ZTR video compression scheme to simultaneously target very low bit-rate applications as well as to support both multirate and multiresolution video scalability.

In this proposed video coding scheme, we employ a block-based motion compensated 3-D wavelet (packet) decomposition framework to first motion-match the frames within a GOF prior to 3-D wavelet transform. A new data structure called TRI-ZTR [27], [28], which forms an extension of the original Shapiro's zerotrees [21], [22], is then used to efficiently encode the important wavelet coefficients. By combining the ideas of layered/progressive coding and embedded resolution block coding, and through the use of resolution flags (RFG's) during the primary and secondary passes, we can provide video scalability with fine granularity. It was shown how a fully embedded and resolution partitioned video bit stream can be generated to support different video scaling parameters, such as bit rate, distortion level, spatial resolution, frame rate, decoding hardware complexity, and end-to-end coding delay. Finally, simulation results demonstrate the effectiveness of TRI-ZTR to give at least comparable visual video quality to H.263, in addition to providing a high degree of video scalability. Future research will include an adaptive scheme to choose between TRI-ZTR with or without motion compensation on a GOF-by-GOF basis, an improved inverse motion compensation scheme, and exploiting the inter-GOF redundancies.

Acknowledgment

The authors would like to thank the anonymous reviewers who contributed very useful feedback and suggestions to improve the quality of this paper.

References

[1] 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.
[2] Y. W. Chen and W. A. Pearlman, "Three-dimensional subband coding of video using the zerotree method," in *Symp. Visual Commun. Image Processing*, SPIE, vol. 2727, Mar. 1996.
[3] T. Chiang and D. Anastassiou, "Hierarchical coding of digital television," *IEEE Commun. Mag.*, vol. 32, pp. 38–45, May 1994.
[4] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 45, pp. 485–560, 1992.
[5] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. 41, pp. 906–966, 1988.
[6] ——, "Ten lectures on wavelets," in *CBMS-NSF Reg. Conf. Ser. Appl. Math.*, vol. 61, SIAM, Philadelphia, PA, 1992.
[7] ITU Telecommunication Standardization Sector LBC-95, "Video codec test model TMN5," available from Telenor Research at http://www.nta.no/brukere/DVC/.
[8] L. Haibo, L. Astrid, and F. Robert, "Image sequence coding at very low bitrates: A review," *IEEE Trans. Image Processing*, vol. 3, pp. 589–609, Sept. 1994.
[9] M. Höetter, "Differential estimation of the global motion parameters zoom and pan," *Signal Processing*, vol. 16, pp. 249–265, 1989.
[10] D. J. LeGall, "The MPEG video compression algorithm," *Signal Processing: Image Commun.*, vol. 4, pp. 129–140, 1992.
[11] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 244–250, Apr. 1992.
[12] J. L. Mitchell *et al.*, *MPEG Video Compression Standard*. London, U.K.: Chapman & Hall, 1997.
[13] K. N. Ngan and W. L. Chooi, "Very low bit rate video coding using 3-D subband approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 309–316, June 1994.

[14] J.-R. Ohm, “Three-dimensional subband coding with motion compensation,” *IEEE Trans. Image Processing*, vol. 3, pp. 559–571, Sept. 1994.
[15] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1992.
[16] L. M. Po and W. C. Ma, “A novel four-step search algorithm for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 313–317, June 1996.
[17] C. I. Podilchuk, N. S. Jayant, and N. Farvardin, “Three-dimensional subband coding of video,” *IEEE Trans. Image Processing*, vol. 4, pp. 125–139, Feb. 1995.
[18] A. Said and W. A. Pearlman, “Image compression using the spatial-orientation tree,” in *IEEE Int. Symp. Circuits Syst.*, Chicago, IL, May 1993, pp. 279–282.
[19] ——, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.
[20] D. G. Sampson, E. A. B. da Silva, and M. Ghanbari, “Low bit-rate video coding using wavelet vector quantization,” in *IEEE Proc. Visual Image Signal Processing*, vol. 142, pp. 141–148, June 1995.
[21] J. M. Shapiro, “An embedded wavelet hierarchical image coder,” in *Proc. IEEE Int. Conf. ASSP*, vol. 4, Mar. 1992, pp. 657–660.
[22] ——, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
[23] D. Taubman and A. Zakhor, “Multirate 3-D subband coding of video,” *IEEE Trans. Image Processing*, vol. 3, Sept. 1994.
[24] D. Taubman, “Directionality and scalability in image and video compression,” Ph.D. dissertation, Univ. California, Berkeley, 1994.
[25] A. M. Tekalp, *Digital Video Processing*. Englewood Cliffs: Prentice-Hall Signal Processing Series, 1995.
[26] J. Y. Tham, “Detail preserving image compression using wavelet transform,” Best Student Paper, IEEE Region 10 Student Paper Contest (UG Category) 1995, *IEEE Student Papers Book*.
[27] ——, “The application of wavelet-based techniques to scalable low bit rate video compression,” Interim Tech. Rep. Presentation, Dep. Elec. Eng., National Univ. Singapore, Dec. 1995.
[28] J. Y. Tham, S. Ranganath, and A. A. Kassim, “Scalable low bit rate video compression using motion compensated 3-D wavelet decomposition,” in *IEEE ICCS/ISPACS 1996*, vol. 3, Nov. 1996, pp. 39.7.1–39.7.5.
[29] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, “A novel unrestricted center-biased diamond search algorithm for block motion estimation,” submitted for publication.
[30] I. H. Witten, R. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Commun. ACM*, vol. 30, pp. 859–861, July 1990.

Jo Yew Tham was born in Kuala Lumpur, Malaysia, in 1971. He was awarded the ASEAN scholarship in 1989 to study in Singapore. In 1995, he received the B.Eng. (Hons.) degree in electrical engineering from the National University of Singapore (NUS).

Currently, he is working as a Research Associate with the Wavelets Strategic Research Programme, NUS, and is pursuing the Ph.D. degree in electrical engineering. His current research interests involve scalable wavelet-based compression techniques for very low bit-rate coding, design and application of multiwavelets, fast motion estimation algorithms, and development of Internet-based multimedia applications.

Mr. Tham received the NUS Engineering Innovation Award and the Best Student Paper Award (UG Category) of the IEEE Regional Student Paper Contest for his work on “Detail Preserving Image Compression using Wavelet Transform.” He is a member of the Malaysian MENSA Society.

Surendra Ranganath received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1975, the M.E. degree in electrical communications engineering from the Indian Institute of Science, Bangalore in 1977, and the Ph.D. degree in electrical engineering from the University of California at Davis in 1983.

From 1982 to 1985, he was with the Applied Research Group of Tektronix, Inc., where he was working in the area of digital video processing for enhanced and high definition TV. From 1986 to 1991, he was with Philips Laboratories, NY, where he was developing algorithms for medical imaging applications. In 1991, he joined the Department of Electrical Engineering at the National University of Singapore, where he is a Senior Lecturer. His current interests are in applications of image processing and computer vision for multimedia, image and video coding, medical imaging, and neural networks.

Ashraf A. Kassim received the B.Eng. (First Class Hons.) and M.Eng. degrees in electical engineering from the National University of Singapore in 1985 and 1987, respectively. He received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, in 1993.

From 1986 to 1988, he worked on the development of machine vision systems at Texas Instruments. Since 1993, he has been a Lecturer with the Electrical Engineering Department at the National University of Singapore. His research interests include neural networks, robot guidance, machine vision, and image processing.