

# Optimal 3-D Coefficient Tree Structure for 3-D Wavelet Video Coding

Chao He, Jianyu Dong, *Member, IEEE*, Yuan F. Zheng, *Fellow, IEEE*, and Zhigang Gao

**Abstract**—An optimal three-dimensional (3-D) coefficient tree structure for 3-D zerotree wavelet video coding is considered in this paper. The 3-D zerotree wavelet video coding is inspired by the success of the two-dimensional (2-D) zerotree wavelet image coding. Existing 3-D zerotree wavelet video codecs use the either symmetric or symmetric-alike 3-D tree structure, which is a straightforward extension of the symmetric 2-D tree structure in the zerotree wavelet image coding. In this paper, we show that the 3-D zerotree coding does not need to be applied symmetrically along all the directions, as the 2-D zerotree image coding does. We find that an asymmetric 3-D tree structure working with a more flexible asymmetric 3-D wavelet transform can produce a higher compression ratio than traditional symmetric approaches. The new 3-D tree structure can be used to improve the rate-distortion performance of many existing 3-D zerotree wavelet video codecs without sacrificing other features such as scalability and computational efficiency. The new tree structure is applied to the 3-D set partitioning in hierarchical trees method and receives convincing peak signal-to-noise ratio improvements in experiments.

**Index Terms**—Set partitioning in hierarchical trees (SPIHT), symmetric and symmetric-alike three-dimensional (3-D) tree structure, 3-D zerotree coding, video compression, wavelet transform.

## I. INTRODUCTION

WITH THE increasing demands of video streaming over computer networks and video database browsing, more features are desired for video compression, such as low computational complexity and good scalability in addition to a good rate-distortion performance [25], [28]. Conventional video compression using hybrid motion compensation (MCP) and discrete cosine transform (DCT) algorithms which have a number of advantages, such as reasonably high compression ratio, mature technology in DCT, and availability of industrial standards for implementation. On the other hand, the MCP-DCT scheme has difficulty in providing the feature just mentioned. The reason is twofold. First, motion compensation involves a searching procedure which is expensive in computation. Second, motion compensation is applied to images with a fixed resolution, which is not easily scalable. While much research is being performed to solve these problems for MCP, a popular alternative is the

three-dimensional (3-D) wavelet transform video coding [10]. The basic idea is to replace the motion compensation by wavelet transforms along the temporal direction to remove temporal redundancy. The temporal wavelet transform has a much lower computational complexity than the motion compensation, since no exhaustive searching computation is involved. In the early 3-D wavelet-based approaches [5], [8], [9], [21], [22], scalar or vector quantization with run length coding is used to encode the wavelet coefficients. These methods are not inherently scalable.

Scalability is an essential functionality in many image or video applications, as discussed in [14]. The term scalability means that the decoder can reconstruct the image or video to different qualities by decoding various amounts of bits from a single bit stream. The more bits the decoder uses, the higher the quality of the reconstructed image or video it produces. As a result, an encoder does not need to encode the original content at many different bit rates or resolutions to serve users with difference bandwidths. It can generate only one bitstream to meet many needs.

Since the invention of the embedded zerotree wavelet algorithm (EZW) [24], many zerotree wavelet image codecs such as set partitioning in hierarchical trees (SPIHT) [23], significance-linked connected component analysis (SLCCA) [3], and listless SPIHT [18] have been proposed. In addition to such advantages as simpler codec structure, lower computational complexity, and better rate-distortion performance, most of them are scalable very well.

Motivated by the success of the zerotree methods in image compression, researchers have extended nearly all of them from two-dimensional (2-D) to 3-D for video coding [2], [4], [11], [12], [17], [19], [20], [25], [27]–[29]. The basic structure of these codecs is relatively straightforward. First, a 3-D wavelet transform is applied on a number of consecutive frames called a group of frames (GOF) of the video. Second, a 3-D tree structure is defined for the wavelet coefficients. Finally, the wavelet coefficients are coded using the zerotree algorithms. The extension of the EZW algorithm has been done in [2] and [4]. The work of [28] is the extension of the SLCCA. Tham *et al.* [25] use the TriZTR idea, which defines three zerotrees. In [11], a general low-delay method is proposed. The first extension of the SPIHT is the 3-D SPIHT [13]. In [17], an improved version of the SPIHT with seven trees is developed. Reference [19] is the Listless SPIHT's extension, whose contribution is the low computational complexity. Reference [12] is a modification of [13], where an unbalanced zerotree is permitted for low-delay coding. Another contribution of [12] is its color-embedded coding, in which  $Y$ ,  $U$ , and  $V$  components are coded by the SPIHT algorithm all together to generate a fully color-embedded bit stream.

Manuscript received January 6, 2001; revised May 18, 2003. This paper was recommended by Associate Editor Z. Xiong.

C. He, Y. F. Zheng, and Z. Gao are with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: hec@ee.eng.ohio-state.edu; zheng@ee.eng.ohio-state.edu; gaoz@ee.eng.ohio-state.edu).

J. Dong was with The Ohio State University, Columbus, OH 43210 USA. She is now with California State University, <AUTHOR: PLEASE PROVIDE CURRENT LOCATION AND EMAIL>

Digital Object Identifier 10.1109/TCSVT.2003.816514

In [30], the resolution and frame rate scalability is obtained by partitioning and/or reordering the bit stream from the 3-D SPIHT. A comprehensive version of the 3-D SPIHT is presented in [14]. The 3-D SPIHT has also been used in content-based coding in [20]. In general, the 3-D zerotree based methods are comparable with H.263 in rate-distortion performance, but perform better in computational simplicity and scalability [14]. The scalability here includes not only image resolution, but also frame rate [14], [25], [30].

There are three issues involved in the 3-D zerotree wavelet video coding. The first issue is the sequence of the 3-D wavelet transform. For the 2-D image, the symmetric 2-D wavelet transform, which performs the 1-D wavelet transforms along  $X$  and  $Y$  dimensions alternately, is an obvious choice. For the 3-D video, the decoupled 3-D wavelet transform (or called wavelet packet transform) is better than the symmetric sequence. The second issue is the extension of the dimensions of the zerotree coding algorithms. This problem is trivial since there is no limitation on the dimensions in almost all the 2-D zerotree coding algorithms. The last issue is how to define an efficient 3-D tree structure for the wavelet coefficients. Unfortunately, most of the existing 3-D zerotree codecs do not consider the structure of the tree systematically. A commonly used tree is the symmetric 3-D tree [4], [11], [13], [17], [19], [20], [28]. The performance of the symmetric 3-D tree is not optimal because the properties of the video signal and the wavelet coefficients are not symmetric along all three dimensions. In [25], an asymmetric tree structure is defined, but it does not follow the idea of the zerotree coding exactly. In [14], a symmetric-alike tree structure is proposed which performs better than the strictly symmetric tree structure. Unfortunately, its performance is still limited by some symmetric requirements in its tree structure. For example, the transform stages along all the dimensions are equal, the basic coefficient unit for arithmetic coding is a symmetric  $2 \times 2 \times 2$  block, and the tree structure is modified from the symmetric tree in order to fit the subband structure after the decoupled 3-D wavelet transform.

The 3-D zerotree coding is also used in multispectral image compression [6]. In [6], the Karhunen–Loeve transform is used along the temporal direction to remove very strong correlation among images. A different 3-D tree is used there because of the Karhunen–Loeve transform. First, a 2-D tree is defined in each frame. Then, the roots of these 2-D trees are attached one by one to form a 3-D tree.

In this paper, we present a more efficient 3-D tree structure for zerotree wavelet video coding. We propose three rules which are drawn from the principles of the zerotree algorithms. The most important rule is that a longer tree is better in clustering zeros, and therefore better for compression. Following these rules, we develop a new asymmetric 3-D tree structure which can improve the rate-distortion performance while maintaining all other features of the 3-D zerotree wavelet video codecs. The new 3-D tree structure is applied to the 3-D SPIHT and achieves a significant peak signal-to-noise ratio (PSNR) improvement.

The paper is organized as follows. Section II presents an analysis of the structures of the video signal and the 3-D wavelet transform, respectively. In Section III, we study the efficient tree structure and describe the three tree construction rules. A more

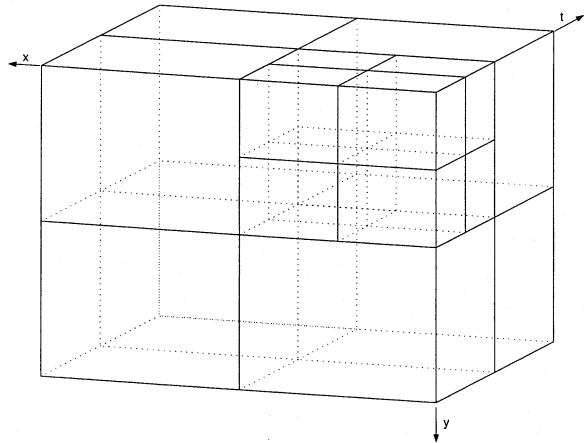


Fig. 1. Symmetric 3-D wavelet transform.

TABLE I  
AVERAGE STANDARD DEVIATIONS (STD) OF VIDEO  
SEQUENCES ALONG HORIZONTAL, VERTICAL,  
AND TEMPORAL DIRECTIONS

STD	X	Y	T
Carphone	50.63	45.76	13.80
Mother and Daughter	33.64	35.03	12.28
Hall Monitor	34.74	42.72	5.99

efficient 3-D tree structure is developed by following the three rules. In Section IV, the implementation of the 3-D SPIHT using the new 3-D tree is described. Finally, conclusions are given in Section V.

## II. 3-D WAVELET TRANSFORM

The first part of a 3-D zerotree wavelet video codec is a 3-D wavelet transform applied to a GOF. In order to clearly describe the 3-D wavelet transform, we define the following notations: 1)  $W_X$ ,  $W_Y$ , and  $W_T$  are the wavelet transforms along the  $X$ ,  $Y$ , and  $T$  directions, respectively; 2)  $L_X$ ,  $L_Y$ , and  $L_T$  are the operators which produce low-pass subbands after the  $W_X$ ,  $W_Y$ , and  $W_T$  transforms, respectively; 3)  $H_X$ ,  $H_Y$ , and  $H_T$  are the operators which produce high-pass subbands after the  $W_X$ ,  $W_Y$ , and  $W_T$  transforms, respectively; and 4)  $V$  is the GOF which the wavelet transform is applied to.

If the 3-D wavelet transform is applied in a sequence of  $W_T W_X W_Y \dots W_T W_X W_Y$ , it is a symmetric 3-D wavelet transform (Fig. 1) because the wavelet transforms are applied alternately along the  $X$ ,  $Y$ , and  $T$  directions. The symmetric 3-D wavelet transform has been used in [4], [11], [17], and [28] as a straightforward extension from the popular symmetric 2-D wavelet transform in the image compression. Furthermore, since the transform is symmetric, a simple symmetric 3-D tree structure can be used directly in the following zerotree coding algorithms.

The symmetric way is not the best for the video signal because the 3-D video does not have symmetric statistical properties along all the directions. Table I shows the average standard deviation (STD) of the first 128 illumination frames of the three video sequences used in [14] along the horizontal  $X$ , vertical

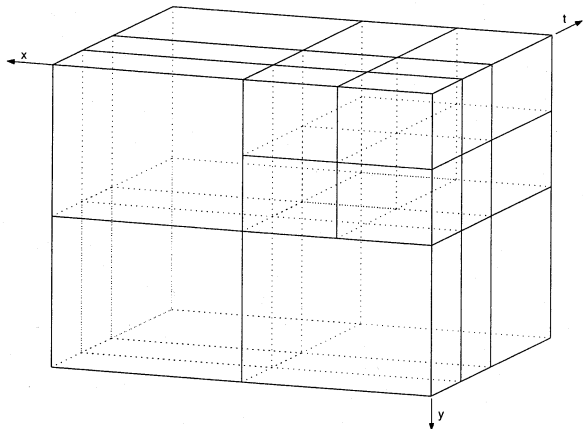


Fig. 2. Decoupled 3-D wavelet transform.

$Y$ , and temporal  $T$  directions, respectively. As an example, the average STD along the temporal direction  $T$  is calculated by

$$\text{mean}_{y,x}(\text{std}_t(V(t,y,x)))$$

where  $V(t,y,x)$  is the GOF,  $\text{std}_t$  is the function of STD with respect to  $t$ , and  $\text{mean}_{y,x}$  is the mean function with respect to  $y$  and  $x$ . The result shows that the STD along  $T$  is much smaller than the STDs along the other two directions  $X$  and  $Y$ , while the STDs along  $X$  and  $Y$  are very close. Therefore, it is reasonable to apply the transforms along the  $T$  direction in a different way from the transforms along the  $X$  and  $Y$  directions in the 3-D wavelet transform.

This problem has been investigated in [1] and [14], and a decoupled 3-D wavelet transform (or so-called 3-D wavelet packet transform) is used there. The decoupled wavelet transform has a pattern of  $W_T \dots W_T W_X W_Y \dots W_X W_Y$  as shown in Fig. 2. In the decoupled 3-D wavelet transform, several cascaded wavelet transforms along the  $T$  direction remove the correlation along the temporal direction first. Then, a set of alternate wavelet transforms along the  $X$  and  $Y$  directions remove the correlation in the spatial domain. Most recent 3-D wavelet video codecs use the decoupled 3-D wavelet transform [1], [5], [8], [12], [13], [15], [16], [19]–[21], [25], [29], [30] for its better signal decomposition ability [1], [14].

### III. 3-D WAVELET COEFFICIENT TREE STRUCTURE FOR ZEROTREE CODING

After the 3-D wavelet transform, a 3-D coefficient tree structure needs to be defined for the zerotree coding algorithms to code the wavelet coefficients. Although the symmetric or symmetric-alike tree is most used in 3-D zerotree wavelet video codecs, the zerotree coding algorithms are quite flexible with the structure of the tree. Any tree structure can be used as long as it covers all the wavelet coefficients. Certainly, different trees have different performances in compression. Therefore, the structure of the tree is of importance for the compression. The symmetric tree may perform very well in 2-D image coding, but not the best choice for coding videos for the reasons stated earlier. In this section, we will discuss how to construct an efficient tree for the 3-D zerotree wavelet video coding. It is not feasible to

quantitatively relate the tree structure with the performance of compression because we do not have a constant model for the video signal, as its content changes constantly. For the sake of discussion, we will make some assumptions and simplifications, and will use the result of experiments to support our analysis.

#### A. Rules for Constructing Optimal Tree

The wavelet transform is an energy compacting process. After the transform and quantization, only a few coefficients are nonzero (significant) while many other coefficients are zero (insignificant). Compression can be achieved by efficiently coding the positions of the nonzero and zero coefficients and the values of the former. Thus, a zerotree coded bit stream contains two kinds of bits, for value and position, respectively. It is not simple to reduce the bits for value because the wavelet transform is a signal decomposition process, and there is not close correlation among the values of the wavelet coefficients. Fortunately, the bits for position can be saved significantly due to the fact that the positions of zero and nonzero coefficients often follow certain patterns. For example, if we scan the wavelet coefficients in one subband in a predefined order such as a zig-zag, we may see many sequences of zeros. Run length and stack run [26] coding algorithms take advantage of this pattern by using a single symbol to represent a sequence of zeros. Another popularly utilized pattern is the zerotree. The zero (after quantization or thresholding) wavelet coefficients corresponding to the same local area in the original signal often form a tree structure in the wavelet domain. A high compression ratio is achieved by using a single symbol or bit to represent a tree of zero coefficients (zerotree) [23], [24]. In general, the more zero coefficients an algorithm can cluster together, the higher compression ratio it can achieve.

The zero wavelet coefficients can form a tree structure because a low-frequency wavelet coefficient of an image is usually larger than its corresponding high-frequency wavelet coefficients [24]. If a low-frequency wavelet coefficient is zero, it is highly likely that all its corresponding high-frequency wavelet coefficients are also zero. This zero distribution pattern gives us a chance to cluster zero coefficients efficiently in a tree structure. Fig. 3 shows 16 coefficients  $\{A, B, C, \dots, P\}$  in a symmetric 2-D tree structure for the zerotree coding. These coefficients correspond to the same local region of the image because they occupy the same relative position in each subband. We call them a coefficient set. In this way, all the wavelet coefficients can be grouped into coefficient sets and the amount of the coefficient set is the amount of the coefficients in the lowest subband. In the figure, the number on the right side of each coefficient is the magnitude (before quantization). In each subband, coefficients belonging to the same coefficient set form a coefficient block such as  $\{E, F, G, H\}$ . Arrows represent the parent-child relationships. A tree can be partitioned into isolated coefficients and small trees. For example, the tree in Fig. 3 can be partitioned into an isolated coefficient  $A$  and three smaller trees with roots in  $B$ ,  $C$ , and  $D$ , respectively. If a tree's root is located at the lowest subband, it is a full length tree. If a tree's coefficients are all zero, it is a zerotree. The idea of the zerotree coding is to represent all the wavelet coefficients as a combination of the isolated coefficients and the zerotrees, and then output the value

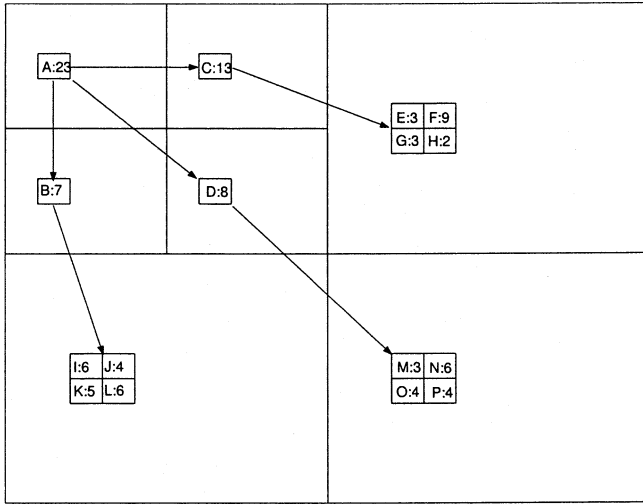


Fig. 3. 2-D coefficient tree structure.

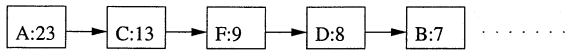


Fig. 4. Longest sorted coefficient tree structure.

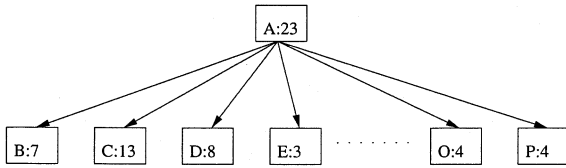


Fig. 5. Widest coefficient tree structure.

of the isolated coefficients. In general, the zerotree coding algorithm generates two kinds of bits: the position bits (positions of the isolated coefficients and the zerotrees) and the value bits (values of the isolated coefficients). Without considering the optional entropy coding, we can clearly separate the output bits for every coefficient set. That means we can study the zerotree coding of each coefficient set independently.

We use the coefficient set in Fig. 3 as an example. The tree structure of the 16 coefficients can be arbitrary instead of symmetric, as shown in Fig. 3. Figs. 4 and 5 show two examples of such. We argue that for a single coefficient set, the longest sorted tree produces the best compression ratio.

To prove our argument, we first use *Algorithm 1* as a zerotree coding algorithm to code a single coefficient tree. Given a quantization step size, the first step generates the position bits, and the second step generates the value bits. Notice that this simple algorithm is close to the space-frequency quantization (SFQ) algorithm in [31]. It can also be viewed as a simplified version of EZW or SPIHT.

*Algorithm 1* A simple coefficient tree coding algorithm.

- 1) Quantize all the coefficients, and browse the coefficient tree from top to bottom and from left to right. Skip the coefficients included in a coded zerotree (which will be defined later). If the current coefficient is the root of a zerotree (which means that all its offspring coefficients and itself are zero), output bit “0” to code this zerotree;

otherwise, output bit “1” to indicate this coefficient is an isolated coefficient.

- 2) Code the value of all the isolated coefficients.

Suppose we have a coefficient set  $C = \{c_i, i = [1, N]\}$  with  $N$  coefficients which are sorted from large to small. Corresponding to a threshold  $th$ , suppose we have  $M$  zero coefficients which are defined as  $\text{ZeroC}(C, th) = \{c_i | \text{round}(c_i/th) = 0\}$ . Given a tree structure  $Tr$ , we run *Algorithm 1*. After the first step, the isolated coefficients are  $\text{IsoC}(C, Tr, th)$ . It should be noticed here that isolated coefficients may include zero coefficients, but nonzero coefficients must be isolated coefficients. Consequently, we have  $\text{IsoC}(C, Tr, th) \supseteq C - \text{ZeroC}(C, th)$ . For an arbitrary zerotree structure  $Tr$ , the total amount of bits of the zerotree coding  $\text{Bits}(C, Tr, th)$  is the sum of the amount of the position bits  $\text{PosBits}(C, Tr, th)$  from the first step and the amount of the value bits  $\text{ValBits}(\text{IC}(C, Tr, th))$  from the second step

$$\begin{aligned} \text{Bits}(C, Tr, th) \\ = \text{PosBits}(C, Tr, th) + \text{ValBits}(\text{IC}(C, Tr, th)). \end{aligned} \quad (1)$$

We want to find a  $\widehat{Tr}(C, th)$  which can minimize  $\text{Bits}(C, Tr, th)$

$$\begin{aligned} \widehat{Tr}(C, th) &= \arg \min_{Tr} (\text{Bits}(C, Tr, th)) \\ &= \arg \min_{Tr} (\text{PosBits}(C, Tr, th) \\ &\quad + \text{ValBits}(\text{IC}(C, Tr, th))). \end{aligned} \quad (2)$$

First, we consider how to minimize position bits

$$\widehat{Tr}(C, th) = \arg \min_{Tr} (\text{PosBits}(C, Tr, th)).$$

Obviously, we have

$$N - M + 1 \leq \text{PosBits}(C, Tr, th) \leq N.$$

The maximum amount is  $N$  when every coefficient needs one bit to represent whether it is the root of a zerotree or an isolated coefficient. The minimum amount is  $N - M + 1$ , since every nonzero coefficient needs one bit of “1” to indicate it is an isolated coefficient, and zero coefficients need at least one bit. The minimum amount is obtained when all  $M$  zero coefficients form one zerotree. For a given threshold  $th$ , the coefficient tree structure  $\widehat{Tr}(C, th)$  which clusters all the zero coefficients  $\text{ZeroC}(C, th)$  is thus the optimal tree and generates the least amount of the position bits.

For the tree  $\widehat{Tr}(C, th)$ , we have  $\text{IsoC}(C, \widehat{Tr}, th) = C - \text{ZeroC}(C, th)$ . Therefore,  $\text{ValBits}(\text{IsoC}(C, \widehat{Tr}, th)) = \text{ValBits}(C - \text{ZeroC}(C, th))$ . Then for an arbitrary tree  $Tr$ , one has  $\text{ValBits}(\text{IsoC}(C, Tr, th)) \geq \text{ValBits}(C - \text{ZeroC}(C, th))$  because  $\text{IsoC}(C, Tr, th) \supseteq C - \text{ZeroC}(C, th)$ . That means that  $\widehat{Tr}(C, th)$  minimizes not only  $\text{PosBits}(C, Tr, th)$  but also  $\text{ValBits}(\text{IsoC}(C, Tr, th))$ . As a result, we have

$$\widehat{Tr}(C, th) = \widehat{Tr}(C, th) = \arg \min_{Tr} (\text{Bits}(C, Tr, th)). \quad (3)$$

In many zerotree coding methods such as EZW and SPIHT, a large threshold is set at the beginning and divided by two after each recursive coding pass. Therefore, we have to find a tree which is optimal for any threshold, but not just one. We argue that the longest sorted tree is such a tree. We define the longest sorted tree  $Tr_{ls}(C)$  as the tree structure of the coefficient set  $\{c_i\}$  which satisfies the following conditions: 1)  $c_{i+1}$  is the only son of  $c_i$ ; 2)  $c_1$  is the root without parent; and 3)  $c_N$  is the deepest

TABLE II  
SUBBAND DEPENDENCE. ROW INDEX IS  $i$  AND COLUMN INDEX IS  $j$

$P(S_i, S_j)$	1	2	3	4	5	6	7	8
1( $L_X L_Y L_X L_Y L_T L_T V$ )		0.82	0.87	0.95	0.94	0.99	0.99	1.00
2( $H_X L_Y L_X L_Y L_T L_T V$ )	0.06		0.50	0.89	0.70	0.94	0.96	0.98
3( $L_X H_Y L_X L_Y L_T L_T V$ )	0.05	0.41		0.88	0.74	0.93	0.95	0.98
4( $L_X L_Y L_X L_Y H_T L_T V$ )	0.04	0.30	0.41		0.57	0.93	0.97	0.99
5( $H_X H_Y L_X L_Y L_T L_T V$ )	0.04	0.30	0.42	0.87		0.91	0.94	0.97
6( $H_X L_Y L_X L_Y H_T L_T V$ )	0.03	0.28	0.38	0.86	0.54		0.94	0.96
7( $L_X H_Y L_X L_Y H_T L_T V$ )	0.03	0.28	0.37	0.86	0.53	0.91		0.96
8( $H_X H_Y L_X L_Y H_T L_T V$ )	0.03	0.27	0.37	0.85	0.53	0.90	0.93	

leaf without son. Fig. 4 shows the longest sorted tree. For any given threshold  $th$ , since  $\{c_i\}$  has been sorted, we know that those  $M$  zero coefficients are  $\{c_i | i \in [N - M + 1, N]\}$ . In the longest sorted tree, all the zero coefficients are clustered as a zerotree with root at  $c_{N-M+1}$ . Therefore,  $Tr_{ls}(C)$  is optimal for any threshold  $th$

$$Tr_{ls}(C) = \arg \min_{Tr} (\text{Bits}(C, Tr, th)), \quad \text{for all } th. \quad (4)$$

Unfortunately, it is not possible to apply  $Tr_{ls}(C)$  to every coefficient set  $C$ , because there is no way to guarantee that all the coefficient sets have the same orders of magnitude, and a lot of extra bits must be used to describe the order of coefficients. A simple way to solve this problem is to associate a coefficient position with the information of its number subband, and assume all the coefficients in one coefficient block to be neighbors/brothers spatially. This restriction and the definition of the coefficient set enable us to use a subband tree to represent all the coefficient trees. A subband tree is a tree structure in which each node is a subband in the wavelet transform domain. *Algorithm 2* describes how to derive the coefficient tree structure from the subband tree structure for the 2-D case.

If the magnitude orders of the coefficient sets are completely random, the zerotree idea may not work at all, since there is not a single subband tree structure which can claim to be efficient for every coefficient set. Fortunately, the magnitude order of the wavelet coefficients is highly related to the subband that they are in. In general, the lower (frequency) a subband is, the larger the magnitude of the coefficients is in the subband, and the coefficients in the subbands at the same level tend to have similar magnitudes. This general observation of the magnitude order gives us a possibility to construct a subband tree which can give the best average performance among all the coefficient sets.

*Algorithm 2* Derive the coefficient tree from the subband tree.

$ps$ : matrix representation of the parent subband.

$pc = ps(m, n)$ : a coefficient in  $ps$ .

$css$ : array holding  $ps$ 's  $L$  child subbands.

$ccs$ : array holding  $pc$ 's child coefficients.

$ttlsons = 0$

for  $i = 0 : L - 1$

$sh = \text{height}(css(i)) / \text{height}(ps)$

$sw = \text{width}(css(i)) / \text{width}(ps)$

for  $j = 0 : j < sh - 1$

for  $k = 0 : sw - 1$

$ccs(\text{ttlsons}) = css(i, m * sh + j, n * sw + k)$

$ttlsons = \text{ttlsons} + 1$

end

end

end

We already know that for the zerotree coding of one coefficient set, the longest sorted coefficient tree is the best. This result has two implications: 1) the magnitude of the parent coefficient should be larger than that of the child coefficient; and 2) a longer tree is better for compression. We assume that the same rules can also be applied in the design of the subband tree. However, the first rule needs to be modified since we now have subbands rather than coefficients. Instead of comparing the magnitude of the parent and child coefficients, we calculate the dependence between two subbands  $S_i$  and  $S_j$ . The dependence  $P(S_i, S_j)$  is defined as the fraction of the parent coefficients in  $S_i$  whose magnitude is equal to or larger than that of all its children's coefficients in  $S_j$ . Here, the magnitude is that after the quantization which we do to remove the effect of small noisy coefficients. Table II is an example of the values of the subband dependences. The decoupled 3-D wavelet transform in Fig. 2 is used where the wavelet transform is applied three times along each of the three axes. The video sequence used is Hall Monitor, the frame rate is 10 frames per second (fps), and the quantization step size is 16.

From the above discussion, we draw three rules for the subband tree construction.

- 1) Every subband is a node of the subband tree, and the coefficient tree structure is uniquely determined by the subband tree.
- 2) There is good dependence between the parent subband  $S_i$  and the child subband  $S_j$ .  $P(S_i, S_j) > 0.9$ .
- 3) Always attempt to construct a longer subband tree.

It should be pointed out that the above rules are not the result of a rigorous mathematical derivation but that of a heuristic analysis. We feel that this is a realistic and effective approach considering the random nature of the contents of images. Since assumptions about the magnitudes of the coefficients and their relations are correct in most cases, the rules are effective in constructing an

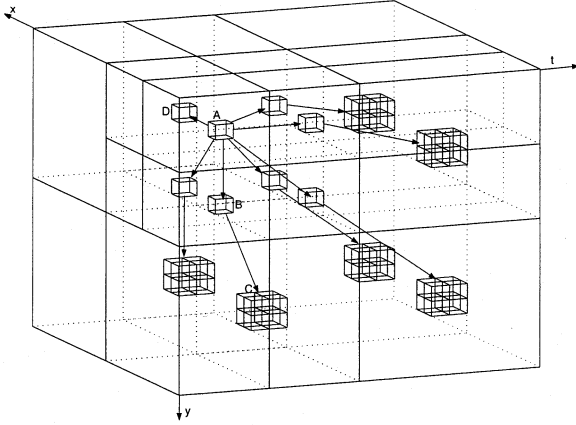


Fig. 6. Symmetric 3-D zerotree after decoupled 3-D wavelet transform.

efficient subband tree, which can be proved by results of applications. Our experimental results also prove that these rules are effective.

In the 2-D image coding, one can see that the symmetric tree structure in Fig. 3 satisfies all the three rules, and is, therefore, the best. We can certainly try different tree structures such as attaching the  $H_X H_Y L_X L_Y V$  subband to the  $H_X L_Y L_X L_Y V$  subband. However, the experiment results show that any other tree structure cannot consistently improve the coding performance, and the improvement, if any, is very small. We thus conclude that the symmetric tree is the best. In the 3-D case, the problem becomes more complex, which will be studied in the next subsection.

### B. New and More Efficient 3-D Coefficient/Subband Tree Structure

For the symmetric 3-D wavelet transform, it is natural and reasonable to use the symmetric 3-D subband tree structure [4], [11], [17], [28]. However, the decoupled 3-D wavelet transform performs better than the symmetric 3-D wavelet transform as mentioned earlier. Unfortunately, the construction of the 3-D subband tree is difficult when one uses the decoupled 3-D wavelet transform, since the latter generates asymmetric subbands, as shown in Fig. 2.

In [25], a 3-D coefficient tree is defined for the decoupled 3-D wavelet transform. When a coefficient is zero, a symmetric 2-D coefficient tree is generated in the same frame. Then this 2-D tree is copied to all the following frames. Finally, the 3-D coefficient tree is the combination of all the 2-D trees in the current and following frames. Unfortunately, this coefficient tree structure violates the first rule. The reason is that the coefficients in such a tree structure do not correspond to the same local region. References [12], [13], [19], [20], [29], and [30] just use the symmetric 3-D coefficient tree structure without considering the fact that the subbands are no longer symmetric after the decoupled 3-D wavelet transform. In the sample coefficient tree shown in Fig. 6, the coefficient  $B$  is the parent of the coefficient block  $C$ . The position of  $B$  shows that it is related to the frames in the middle of the GOF because it is in the center of the subband  $L_X H_Y L_X L_Y L_T L_T V$ , but its child coefficient block  $C$  locates at the boundary of two subbands, which shows that it is related

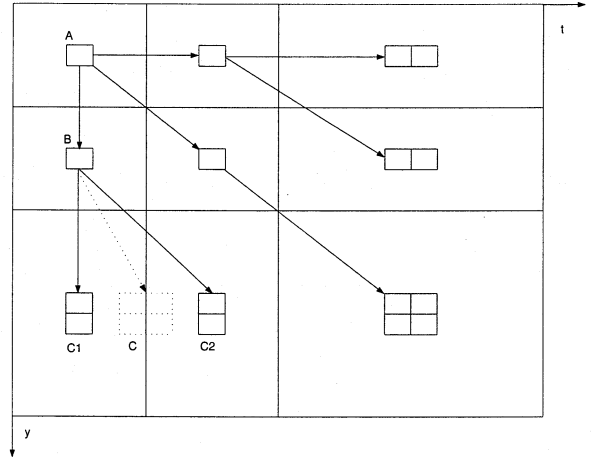


Fig. 7. Symmetric-alike 3-D zerotree after decoupled 3-D wavelet transform.

to the beginning and ending frames of the GOF. In [14], a symmetric-alike 3-D tree structure is used to solve this problem. The coefficient tree structure is shown in Fig. 7. For simplicity, only two dimensions,  $Y$  and  $T$ , are shown. Instead of assigning the coefficients block  $C$  as the child of the coefficient  $B$  (shown as dot arrow), two child blocks  $C_1$  and  $C_2$  are assigned as the children of  $B$ . This structure fits the subband structure, and therefore follows rule 1.

The symmetric-alike subband tree structure follows rules 1 and 2, but does not consider rule 3. Next, we will show how to generate a better 3-D subband tree which follows all the rules. We use Fig. 2 as example. First of all, it should be made clear that the root subband is the lowest subband  $L_X L_Y L_X L_Y L_T L_T V$ . Secondly, for all the subbands calculated from the intermediate subband  $L_T L_T V$ , it is reasonable to follow the way for constructing the 2-D symmetric tree structure since they are the result of the 2-D wavelet transform of the subband  $L_T L_T V$ , and it has been shown that the 2-D symmetric tree is very efficient for the 2-D wavelet transform. Then, the subband  $L_X L_Y L_X L_Y H_T L_T V$  should be the child of the root subband, since it is the low resolution representation of the subbands  $L_T L_T V$ , and  $H_T L_T V$ , which are low- and high-pass subbands after applying  $W_T$  to the subband  $L_T V$ , respectively. Now we consider the positions of the subbands  $L_X H_Y L_X L_Y H_T L_T V$ ,  $H_X L_Y L_X L_Y H_T L_T V$ , and  $H_X H_Y L_X L_Y H_T L_T V$ . These subbands are attached to the root subband by the previous works, because they have very good dependence on the root subband. For example, they are 0.99, 0.99, and 1.00 in Table II. The dependence of these three subbands on the subband  $L_X L_Y L_X L_Y H_T L_T V$  is also very good because they are the result of the 2-D wavelet transform of  $H_T L_T V$ . In fact, they are 0.93, 0.97, and 0.99 in Table II. As a result, the parent of these three subbands could be selected according to rule 3, which results in a longer and narrower tree. If we choose the root subband as the parent, the depth of all the three subbands is one. If we choose  $L_X L_Y L_X L_Y H_T L_T V$  as the parent, the depth is two. Clearly,  $L_X L_Y L_X L_Y H_T L_T V$  is the choice of the new tree structure.

Now consider how to put the subbands  $L_X H_Y H_T L_T V$ ,  $H_X L_Y H_T L_T V$ , and  $H_X H_Y H_T L_T V$  in the tree. In [14], they

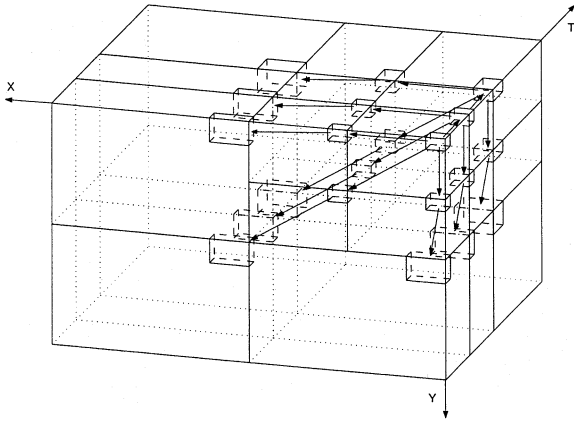


Fig. 8. New 3-D zerotree structure.

are attached as the sons of the subbands  $L_X H_Y L_X L_Y L_T L_T V$ ,  $H_X L_Y L_X L_Y L_T L_T V$ , and  $H_X H_Y L_X L_Y L_T L_T V$ , respectively. Similarly, we find that the dependences of the subbands  $L_X H_Y H_T L_T V$  on  $L_X H_Y L_X L_Y H_T L_T V$ ,  $H_X L_Y H_T L_T V$  on  $H_X L_Y L_X L_Y H_T L_T V$ , and  $H_X H_Y H_T L_T V$  on  $H_X H_Y L_X L_Y H_T L_T V$  are all very good. On the other hand, if we attach  $L_X H_Y H_T L_T V$ ,  $H_X L_Y H_T L_T V$ , and  $H_X H_Y H_T L_T V$  to  $L_X H_Y L_X L_Y L_T L_T V$ ,  $H_X L_Y L_X L_Y L_T L_T V$ , and  $H_X H_Y L_X L_Y L_T L_T V$ , respectively, we get a longer tree. Therefore, based on rule 3, these new parent–child relationships are adopted as the new tree structure.

The same idea is used to put all the remaining subbands in the subband tree structure. The final result is shown in Fig. 8. The detailed coefficient parent–child relationship of the new zerotree is described in *Algorithm 3*. This tree is longer and narrower, while keeping the good dependence between the parent and child subbands. In the symmetric-alike 3-D coefficient tree, a coefficient has up to eight child coefficients and the tree depth is three. In the new 3-D tree, the numbers are five and five, respectively. As a result, this tree should provide a better zero clustering ability than the symmetric-alike tree. There is a similarity between our tree and the tree used in [6] for multispectral image compression, but one may find three major differences between the two by a careful comparison. First, [6] is for multispectral image compression, not for video. Second, [6] uses Karhunen–Loeve decomposition in the spectral domain (corresponding to the time domain in video compression). Therefore, it is not completely wavelet transform. Third, since the Karhunen–Loeve decomposition does not have the dyadic feature of the wavelet transform, the tree structures along the time (spectral) domain are different. In [6], a coefficient has, at most, one son along the spectral domain. In our tree, it is two.

*Algorithm 3* Coefficient parent–child relationship for the new 3-D zerotree.

$WV$ : wavelet coefficients of the video.  
 $Fs$ : total frames of the lowest subband.  
 $Rs$ : total rows of the lowest subband.  
 $Cs$ : total columns of the lowest subband.

$PC = WV(i, j, k)$ : a coefficient at frame  $i$ , row  $j$ , and column  $k$  of  $WV$ .  
 $O(PC)$ : a set of child coefficients of  $PC$ . if  $i < Fs$  and  $j < Rs$  and  $k < Cs$   
 $O(PC) = \{WV(i, j + Rs, k), WV(i, j, k + Cs), WV(i, j + Rs, k + Cs), WV(i + Fs, j, k)\}$   
elseif  $j < Rs$  and  $k < Cs$   
 $O(PC) = \{WV(i, j + Rs, k), WV(i, j, k + s), WV(i, j + Rs, k + Cs), WV(2i, j, k), WV(2i + 1, j, k)\}$   
else  
 $O(PC) = \{WV(i, 2j, 2k), WV(i, 2j + 1, 2k), WV(i, 2j, 2k + 1), WV(i, 2j + 1, 2k + 1)\}$

end

\* if the child coefficient's position is out of  $WV$ , it is removed from  $O(PC)$ .

Another advantage of the new 3-D subband tree is its consistency with the transform sequence of the 3-D wavelet transform, which means that the subband tree can “grow automatically” by following the transform sequence of the 3-D wavelet transform. At the beginning, there is only one subband (original video cuboid  $V$ ), and the subband tree has only one node. In the first step, the video is 1-D wavelet transformed along the temporal direction. Following this transform, the original video cuboid is partitioned into three small cuboids (subbands) and the subband tree grows to a 1-D tree

$$L_T L_T V \rightarrow H_T L_T V \rightarrow H_T V. \quad (5)$$

In the second step, every subband undergoes a 2-D wavelet transform in the spatial domain and is partitioned into seven smaller subbands. These seven subbands follow the symmetric tree structure, since they are generated by the 2-D wavelet transform. As a result, every node in the 1-D subband tree shown in (5) grows along the  $X$  and  $Y$  dimensions, and the 3-D subband tree is finally obtained as shown in (6) at the bottom of the next page, which is identical to the tree we construct in Fig. 8. This “automatically growing” feature makes the new tree structure easily adaptive to different 3-D wavelet or wavelet packet transforms. For any pattern of the transform, we can always find a consistent subband tree structure. For example, we can apply different levels of the wavelet transform along  $X/Y$  and  $T$  directions and construct a 3-D subband tree accordingly. More adaptively, we can even apply different levels of the 2-D wavelet transforms for different temporal subbands after the cascaded  $W_T$  transforms and construct the corresponding 3-D subband tree.

#### IV. IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

We apply the new 3-D subband tree structure to the 3-D SPIHT video codec to test its efficiency, since the 3-D SPIHT has been studied extensively [14]. We use the same sample color video sequences in the experiment which are Carphone, Mother and Daughter, and Hall Monitor sequences, respectively. The same parameters are used during the test. That is, the video size is quarter common intermediate format (QCIF)

TABLE III

PSNR RESULTS. THE SECOND COLUMN (3-D DWT) DESCRIBES LEVELS OF THE TRANSFORMS ALONG  $T/X/Y$  DIRECTION. CU MEANS COEFFICIENT UNIT. SAT IS SYMMETRIC-ALIKE TREE, AND OT IS OUR NEW TREE STRUCTURE. AC IS ARITHMETIC CODING. BPS IS BIT RATE. PSNR IS DISPLAYED IN ORDER OF YUV IN LAST 3 COLUMNS. ROWS 3 AND 9 CORRESPOND TO THE ORIGINAL 3-D SPIHT.

Test ID.	3-D DWT	CU	Tree	bps	AC	PSNR Car	PSNR Mtdr	PSNR Hall
1	4/4/4	$1 \times 1 \times 1$	SAT	30k	No	29.60 36.66 37.41	32.14 38.68 38.39	32.19 37.59 39.89
2	4/4/4	$1 \times 1 \times 1$	OT	30k	No	30.41 37.28 38.02	32.67 39.46 39.20	34.51 38.70 40.96
3	3/3/3	$2 \times 2 \times 2$	SAT	30k	Yes	30.15 36.76 37.65	32.65 38.81 38.47	32.79 37.66 40.01
4	3/3/3	$2 \times 2 \times 2$	OT	30k	Yes	30.26 36.79 37.74	32.74 38.99 38.71	33.39 37.78 40.09
5	4/3/3	$1 \times 2 \times 2$	SAT	30k	Yes	30.66 37.22 38.07	33.06 39.60 39.34	34.76 38.81 41.03
6	4/3/3	$1 \times 2 \times 2$	OT	30k	Yes	30.74 37.49 38.26	33.08 39.69 39.45	35.16 39.17 41.31
7	4/4/4	$1 \times 1 \times 1$	SAT	60k	No	32.02 38.34 38.95	34.70 40.58 40.42	36.63 39.67 41.63
8	4/4/4	$1 \times 1 \times 1$	OT	60k	No	32.88 39.16 39.67	35.23 41.03 40.89	39.06 40.91 42.83
9	3/3/3	$2 \times 2 \times 2$	SAT	60k	Yes	32.83 38.48 39.31	35.41 40.59 40.36	37.61 39.83 41.78
10	3/3/3	$2 \times 2 \times 2$	OT	60k	Yes	32.91 38.62 39.42	35.47 40.72 40.57	37.94 40.18 42.11
11	4/3/3	$1 \times 2 \times 2$	SAT	60k	Yes	33.17 39.26 39.74	35.68 41.21 41.09	39.31 40.99 42.91
12	4/3/3	$1 \times 2 \times 2$	OT	60k	Yes	33.21 39.37 39.84	35.69 41.26 41.14	39.72 41.10 43.05

which is  $176 \times 144$ , the frame rate is 10 fps, the color space is  $4 : 2 : 0$ , and the length of GOF is 16. The implementation of the 3-D SPIHT follows the algorithm presented in [23]. Since the source code of the SPIHT is not available, our 3-D SPIHT implementation is largely based on the 2-D SPIHT in QccPack [7]. As mentioned in [7], QccPack's implementation is up to 0.17 dB worse than the reported PSNR in [23] due to different arithmetic coding implementation. Our result shows that the 3-D SPIHT with the new 3-D tree outperforms the original 3-D SPIHT. The SPIHT algorithm has an option to apply the arithmetic coding to further improve the compression ratio. We compare our tree structure with the symmetric-alike tree structure in two cases: with and without arithmetic coding. In the experiments without the arithmetic coding, we use the same decoupled 3-D wavelet transform, but the levels of the wavelet transforms along all the dimensions are four instead of three, since we do not need to keep  $2 \times 2 \times 2$  ( $T \times Y \times X$ ) coefficient units for the arithmetic coding. In the experiments with the arithmetic coding, the decoupled 3-D wavelet transform with three levels of transform along each direction, and the symmetric  $2 \times 2 \times 2$  coefficient unit are used. Furthermore, since the new 3-D tree does not have any restriction on the 3-D wavelet transform structure, we can use different levels of wavelet transforms along each direction. Therefore, we do

not keep the 2 coefficient width in the lowest subband along the temporal direction because we believe the correlation in the temporal direction should be removed by the wavelet transform, not by the arithmetic coding. We run another test using  $1 \times 2 \times 2$  coefficient block and a 3-D wavelet transform with four levels of temporal transforms. The symmetric-alike tree can also use such a configuration after small modification. The biorthogonal 9/7 wavelet is used along all the directions except the last temporal direction transform, which uses a simple Haar wavelet. Our experimental results show that the Haar wavelet can reduce the computational complexity, and improve the PSNR slightly simultaneously.

The results on the three sequences at 30 and 60 kb/s are shown in Table III. The results show that the new asymmetric tree structure consistently outperforms the symmetric-alike tree. Table IV shows the PSNR improvement of the new structure in various settings. It can be noticed that if there is no arithmetic coding, the improvement is quite big (rows 1 and 2), but if there is arithmetic coding, the improvement is very limited for the first two sequences (rows 3–6). That shows the arithmetic coding is very efficient in removing the redundant information which is not removed by the baseline zerotree coding. One can see from the table that the new 3-D tree generates a larger improvement for the Hall Monitor sequence. The reason is that the Hall Mon-

$$\begin{aligned}
& \nearrow L_X H_Y L_X L_Y L_T L_T V \rightarrow L_X H_Y L_T L_T V \\
L_X L_Y L_X L_Y L_T L_T V & \rightarrow H_X L_Y L_X L_Y L_T L_T V \rightarrow H_X L_Y L_T L_T V \\
& \downarrow \searrow H_X H_Y L_X L_Y L_T L_T V \rightarrow H_X H_Y L_T L_T V \\
& \nearrow L_X H_Y L_X L_Y H_T L_T V \rightarrow L_X H_Y H_T L_T V \\
L_X L_Y L_X L_Y H_T L_T V & \rightarrow H_X L_Y L_X L_Y H_T L_T V \rightarrow H_X L_Y H_T L_T V \\
& \downarrow \searrow H_X H_Y L_X L_Y H_T L_T V \rightarrow H_X H_Y H_T L_T V \\
& \nearrow L_X H_Y L_X L_Y H_T V \rightarrow L_X H_Y H_T V \\
L_X L_Y L_X L_Y H_T V & \rightarrow H_X L_Y L_X L_Y H_T V \rightarrow H_X L_Y H_T V \\
& \searrow H_X H_Y L_X L_Y H_T V \rightarrow H_X H_Y H_T V
\end{aligned} \tag{6}$$

TABLE IV  
PSNR IMPROVEMENT FROM THE NEW TREE STRUCTURE

Test	3-D DWT	CU	bps	AC	Car	Mtdr	Hall
1 vs 2	4/4/4	1x1x1	30k	No	0.81 0.62 0.61	0.53 0.78 0.81	2.32 1.11 1.07
7 vs 8	4/4/4	1x1x1	60k	No	0.86 0.82 0.72	0.53 0.45 0.47	2.43 1.24 1.20
3 vs 4	3/3/3	2x2x2	30k	Yes	0.11 0.03 0.09	0.09 0.18 0.24	0.60 0.12 0.08
9 vs 10	3/3/3	2x2x2	60k	Yes	0.08 0.14 0.11	0.06 0.13 0.21	0.33 0.35 0.33
5 vs 6	4/3/3	1x2x2	30k	Yes	0.08 0.27 0.19	0.02 0.09 0.11	0.40 0.36 0.28
10 vs 11	4/3/3	1x2x2	60k	Yes	0.04 0.11 0.10	0.01 0.05 0.05	0.41 0.11 0.14
3 vs 5			30k	Yes	0.51 0.46 0.42	0.41 0.79 0.83	1.97 1.15 1.02
9 vs 11			60k	Yes	0.34 0.78 0.43	0.27 0.62 0.73	1.80 1.17 1.13
3 vs 6			30k	Yes	0.59 0.73 0.61	0.43 0.88 0.98	2.37 1.51 1.30
9 vs 12			60k	Yes	0.38 0.89 0.53	0.28 0.67 0.78	2.11 1.27 1.27

TABLE V  
COMPUTATION TIME COMPARISON FOR COMPRESSING 16 FRAMES OF THE CARPHONE VIDEO SEQUENCE TO 48 000 BITS

CPU Time (sec)	Test ID	Encoder			Decoder		
		Load Video	DWT	SPIHT encode	SPIHT decode	IDWT	Save Video
3D-SPIHT w/o AC	0	0.26	19.29	4.05	0.33	19.57	0.20
3D-SPIHT w AC	3	0.25	18.23	1.41	0.30	18.37	0.21
Our method w/o AC	1	0.24	19.38	4.69	0.33	19.57	0.21
Our method w AC	6	0.25	19.19	2.15	0.32	19.43	0.21

itor sequence is more asymmetric, as shown in Table I. Its STD differs more between the temporal direction and the other two directions than that of the other two sequences.

The other interesting result is the differences between tests 3 and 5, and between tests 9 and 11, respectively. Tests 3 and 9 correspond to the original 3-D SPIHT in [14]. The results from tests 5 and 11 show larger improvements over 3 and 9, which prove that the  $1 \times 2 \times 2$  coefficient unit with an extra Haar wavelet transform in temporal direction works much more efficiently than the  $2 \times 2 \times 2$  coefficient unit. Although the improvement is not directly from the new asymmetric tree structure, it is consistent with our initial consideration: the zerotree coding does not need to be applied symmetrically. In summary, the best result comes from tests 6 and 12, where the 4/3/3 3-D wavelet,  $1 \times 2 \times 2$  coefficient block, and the new tree structure are used.

The new 3-D zerotree increases computational complexity slightly. Table V is the CPU time comparison between our method and the original 3-D SPIHT. It is based on our programs running on a Pentium III 500-MHz computer. The result shows that the computational complexity of our method is close to that of the original 3-D SPIHT in both encoding and decoding. One can see several other interesting points from this table. First of all, most of the computational time is spent on the transform instead of the coding. This is because we use a set of long wavelet filters. Second, the SPIHT encoding spends much more time than the SPIHT decoding. This is because the encoding part needs to check whether a tree is a zerotree (significance checking). Third, the compression with the arithmetic coding uses less computation time, which seems to contradict our expectation. It is also because of the significance checking. In compression without the arithmetic coding, we apply one more level of wavelet transform along each direction. Therefore, the

subband tree is longer and the significance checking needs more time even than arithmetic coding. Besides the computational efficiency, the new 3-D subband tree supports bit rate, PSNR, resolution, and frame-rate scalability in the same way and as the original 3-D SPIHT does.

The above comparisons are the results of two fixed bit rates. In order to study the behavior of the tree structures in different bit rates, we have tested the video sequences with different bit rates and plot a rate-distortion curve in Fig. 9. In this figure, we compare the performance of three different tree structures, symmetric tree (ST), symmetric-alike tree (SAT), and our new tree (OT), respectively. The symmetric tree works with the symmetric 3-D wavelet transform, while the other trees work with the decoupled 3-D wavelet transform. The transform levels along all the directions are four, and no arithmetic coding is used. The three color video sequences are coded at 30 fps with the GOF size of 16. Only the average PSNR of the  $Y$  color space is displayed. From the figure, we can conclude that the new 3-D tree is consistently better than the symmetric and the symmetric-alike trees.

We also compare the PSNR of each frame in the video. Fig. 10 is the comparison of the PSNR of the  $Y$  color space for each frame of the Hall Monitor sequence. The video is coded at 30 fps, 75 kb/s, 16 frames per GOF, and four levels of wavelet transforms along each direction without arithmetic coding. The small circle on the curves indicates the position of the first frame in each GOF. The figure clearly shows the PSNR degrading of the boundary frames in a GOF which comes from two sources. First, the temporal wavelet transform needs signal padding at the boundary. Second, there are multilevel temporal wavelet transforms. If we replace all the 9/7 wavelet transforms along the temporal direction with simple Haar

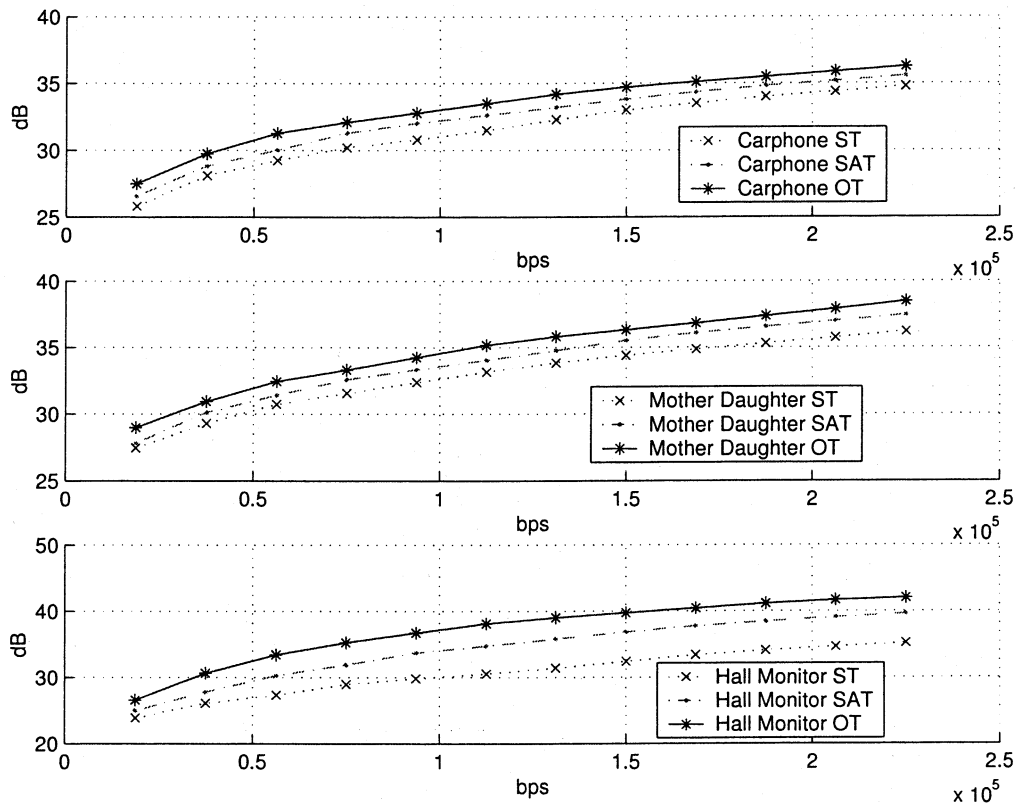


Fig. 9. Rate distortion comparison of different tree structures.

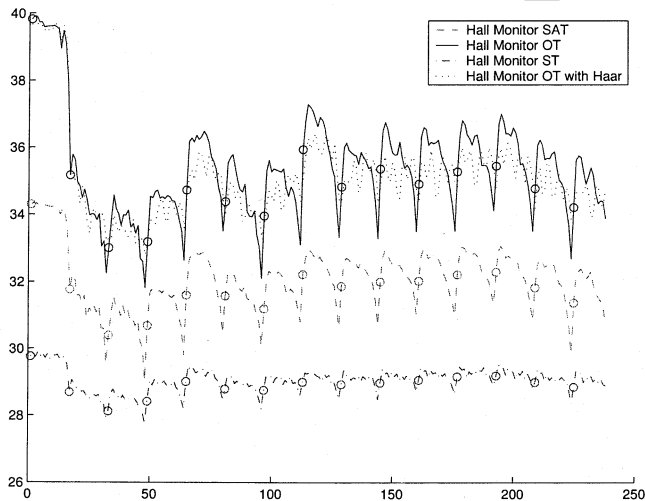


Fig. 10. Frame-by-frame PSNR comparison of the Hall Monitor sequence encoded at 30 fps, 75 kb/s, and 16 frames per GOF.

wavelet transforms, we can reduce the PSNR degrading, since the Haar transform does not need padding at the boundary, but at the same time, the overall PSNR drops about 0.2 dB.

The new tree structure has no limitation on the levels of transform along each direction. Therefore, the length of the GOF can be chosen flexibly according to the requirement of the delay, the memory cost, and the speed of the coding. Theoretically, a larger GOF length may give a higher compression ratio, but requires more memory space and causes delay. For example, encoding a video at 10 fps, a GOF length of 16 means the encoder delay is

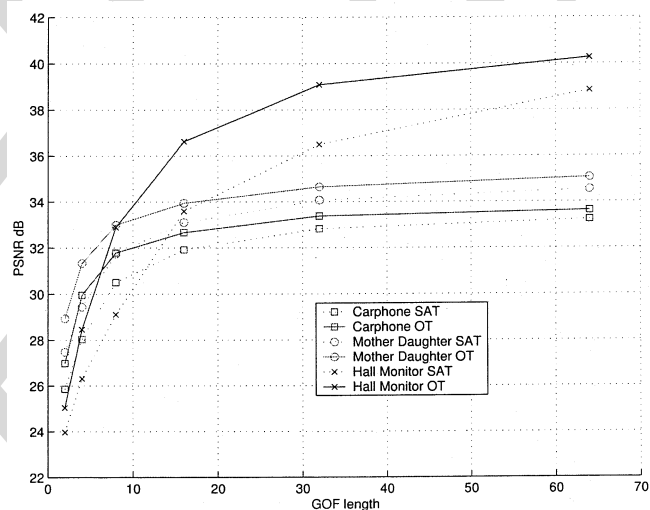


Fig. 11. Performance with different GOF size. Videos are encoded at 30 fps and 90 kb/s.

at least 1.6 s. We test the new tree structure with different GOF sizes, and the result is in Fig. 11. Three color video sequences are coded at 30 fps and 90 kb/s. For different GOF lengths, we always apply the temporal wavelet transform to the end, and use the same levels of the spatial wavelet transforms. It shows that the optimal GOF length for the purpose of the compression is 16 or 32. Any GOF length greater than 32 does not improve the compression ratio too much, but increases the memory requirement and delay significantly.

## V. CONCLUSION

In this paper, we study the optimal 3-D tree structure for the 3-D zerotree wavelet video compression. We present three rules for constructing such a tree for image and video compression. Using these rules, we design a new 3-D coefficient tree structure which can cluster zeros more efficiently, and therefore achieve a higher compression ratio. The new tree structure is applied to the state-of-the-art 3-D SPIHT color video compression scheme to demonstrate its efficiency under different conditions. The experimental results show that the new tree has convincing PSNR improvement for all the tested video sequences when arithmetic coding is not involved. When it is used, the new tree has considerable PSNR improvement for certain video sequences which have limited motions between frames. In both cases, however, the improvement is consistent. We also find that significant PSNR improvements can be obtained by allowing asymmetric coefficient unit in the 3-D SPIHT's arithmetic coding.

With the improvement of the rate-distortion performance, the new tree structure still maintains all the desired features of other 3-D zerotree-based methods, such as computational efficiency and scalability. Furthermore, the new structure is not limited to the 3-D SPIHT algorithm, but also other 3-D zerotree-based methods. The automatically growing feature of the new tree leads to easy implementation of the compression algorithm as well as adaptation to different kinds of 3-D wavelet or wavelet packet transforms.

## REFERENCES

- [1] T. J. Burns, S. K. Rogers, M. E. Oxley, and D. W. Ruck, "A wavelet multiresolution analysis for spatio-temporal signals," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 32, pp. 628–649, Apr. 1996.
- [2] P. Campisi, M. Gentile, and A. Neri, "Three-dimensional wavelet-based approach for a scalable video conference system," in *Proc. Int. Conf. Image Processing*, vol. 3, 1999, pp. 802–806.
- [3] B. Chai, J. Vass, and X. Zhuang, "Significance-linked connected component analysis for wavelet image coding," *IEEE Trans. Image Processing*, vol. 8, pp. 774–784, June 1999.
- [4] Y. Chen and W. A. Pearlman, "Three-dimensional subband coding of video using the zerotree method," in *Proc. SPIE Visual Communications and Image Processing*, vol. 2727, Mar. 1996, pp. 1302–1312.
- [5] S. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 8, pp. 155–167, Feb. 1999.
- [6] P. L. Dragotti, G. Poggi, and A. R. P. Ragozini, "Compression of multispectral images by three-dimensional SPIHT algorithm," *IEEE Trans. Geosci. Remote Sensing*, vol. 38, pp. 416–428, Jan. 2000.
- [7] J. E. Fowler, "QccPack: An open-source software library for quantization, compression, and coding," in *Proc. SPIE Applications of Digital Image Processing XXIII*, vol. 4115, A. G. Tescher, Ed., San Diego, CA, 2000, pp. 294–301.
- [8] K. H. Goh, J. J. Soraghan, and T. S. Durrani, "New 3-D wavelet transform coding algorithm for image sequences," *Electron. Lett.*, vol. 29, pp. 401–402, Feb. 1993.
- [9] W. Hsu and H. Derin, "3-D adaptive wavelet packet for video compression," in *Proc. Int. Conf. Image Processing*, vol. 1, 1995, pp. 602–605.
- [10] G. Karlsson and M. Vetterli, "Three-dimensional subband coding of video," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Apr. 1988, pp. 1100–1103.
- [11] H. Khalil, F. Atiya, and S. I. Shaheen, "Lowering frame-buffering requirements of 3-D wavelet transform coding of interactive video," in *Proc. Int. Conf. Image Processing*, vol. 3, 1999, pp. 852–856.
- [12] B. Kim and W. A. Pearlman, "Fast color-embedded video coding with SPIHT," in *Proc. Data Compression Conf.*, Mar. 1998, p. <AUTHOR: PAGE RANGE?>.

- [13] —, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proc. Data Compression Conf.*, <AUTHOR: MONTH?> 1997, pp. 251–260.
- [14] B. Kim, Z. Xiong, and W. A. Pearlman, "Low bit rate, scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 1374–1387, Dec. 2000.
- [15] I. K. Levy and R. Wilson, "Three-dimensional wavelet transform video compression," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, vol. 2, 1999, pp. 924–928.
- [16] A. S. Lewis and G. Knowles, "Video compression using 3-D wavelet transforms," *Electron. Lett.*, vol. 26, no. 6, pp. 396–398, Mar. 1990.
- [17] G. Lin and E. Liu, "3-D wavelet video codec and its rate control in ATM network," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, 1999, pp. 447–450.
- [18] W. Lin and N. Burgess, "Listless zerotree coding for color images," in *Conf. Rec. 32nd Asilomar Conf. Signals, Systems and Computers*, vol. 1, 1998, pp. 231–235.
- [19] —, "3-D listless zerotree coding for low-bit-rate video," in *Proc. Int. Conf. Image Processing*, vol. 3, 1999, pp. 762–766.
- [20] G. Minami, Z. Xiong, A. Wang, P. A. Chou, and S. Mehrotra, "3-D wavelet coding of video with arbitrary regions of support," in *Conf. Rec. 33rd Asilomar Conf. Signals, Systems, and Computers*, vol. 2, 1999, pp. 1422–1425.
- [21] J. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Processing*, vol. 3, pp. 559–571, Sept. 1994.
- [22] 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.
- [23] A. Said and W. A. Pearlman, "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.
- [24] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
- [25] J. Tham, S. Ranganath, and A. A. Kassim, "Highly scalable wavelet-based video codec for very low bit-rate environment," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 12–27, Jan. 1998.
- [26] 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.
- [27] J. Vass, B. Chai, K. Palaniappan, and X. Zhuang, "Significance-linked connected component analysis for very low bit-rate wavelet video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 630–647, June 1999.
- [28] J. Vass, B. Chai, and X. Zhuang, "3-D SLCCA—a highly scalable very low bit-rate software-only wavelet video codec," in *Proc. IEEE 2nd Workshop Multimedia Signal Processing*, 1998, pp. 474–479.
- [29] A. Wang, Z. Xiong, P. A. Chou, and S. Mehrotra, "Three-dimensional wavelet coding of video with global motion compensation," in *Proc. Data Compression Conf.*, 1999, pp. 404–413.
- [30] Z. Xiong, B. Kim, and W. A. Pearlman, "Multiresolutional encoding and decoding in embedded image and video coders," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 6, 1998, pp. 3709–3712.
- [31] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, pp. 677–693, Jan. 1997.



**Chao He** received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 1995, and the M.S. degree in electrical engineering in 2000 from The Ohio State University (OSU), Columbus, where he is currently working toward the Ph.D. degree.

His research interest includes audio/video compression and streaming.



**Jianyu Dong** (M'02) received the B.E. and M.S. degrees from the University of Science and Technology, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree from The Ohio State University, Columbus, in 2002, all in electrical engineering.

In 2002, she joined the faculty of California State University, <AUTHOR: LOCATION?> where she is currently an Assistant Professor in the Department of Electrical and Computer Engineering. While continuing research on video compression and streaming, she is also involved in the NASA-sponsored project in the SPACE Laboratory, and is leading the research activities of astronomical image processing, compression, and transmission.



**Yuan F. Zheng** (S'82–M'86–SM'90–F'97) received the M.S. and Ph.D. degrees in electrical engineering from The Ohio State University, Columbus, in 1980 and 1984, respectively. His undergraduate education was received at Tsinghua University, Beijing, China in 1970.

From 1984 to 1989, he was with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC. Since August 1989, he has been with The Ohio State University, where he is currently Professor and Chairman of Electrical Engineering. His research interests include two aspects. One is in wavelet transform for image and video compression for internet and satellite communications. Current efforts focus on content-based compression, 3-D wavelet transformation, video object tracking, and content-based retransmission in Internet communications. The other is in robotics, which includes robots for biological applications, multiple robots coordination, legged robots, human-robot coordination, and personal robotics. He is currently on the Editorial Board of the *International Journal of Multimedia Tools and Applications*, on the Editorial Board of *Autonomous Robots*, an Associate Editor of the *International Journal of Intelligent Automation and Soft Computing*, on the Editorial Board of the *International Journal of Intelligent Control and Systems*, and on the Editorial Board of the *International Journal of Control, Automation, and Systems*.

Dr. Zheng was Vice-President for Technical Affairs of the IEEE Robotics and Automation Society from 1996 to 1999. He was an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION between 1995 and 1997. He was the Program Chair of the 1999 IEEE International Conference on Robotics and Automation, Detroit, MI, May 10-15, 1999.



**Zhigang Gao** received the B.S. degree in electronics from Beijing University, Beijing, China, and the M.S. degree in electrical engineering from The Ohio State University, Columbus, in 1997 and 2002, respectively. He is currently working toward the Ph.D. degree in electrical engineering at The Ohio State University.

His research interests are in image/video quality assessment and quality constrained compression using wavelets.