A WAVELET VIDEO CODER USING THREE-DIMENSIONAL EMBEDDED SUBBAND CODING WITH OPTIMIZED TRUNCATION (3-D ESCOT)

Jizheng Xu[+], *Shipeng Li*[+], *Ya-Qin Zhang*[+] *and Zixiang Xiong*[#]

[+]Microsoft Research, China
No. 49, Zhichun Rd, Haidian District,
Beijing, 100080, P.R.C
Phone:+86-10-62617711, Fax:+86-10-6255-5337
Email: xu_jz@hotmail.com,{spli,yzhang}@microosft.com

[#]Dept of Electrical Engineering
Texas A&M University
College Station, TX 77843, USA
Phone:(409) 862-8683, Fax(409) 862-14630,
Email: zx@lena.tamu.edu

ABSTRACT

In this paper, we present an efficient three-dimensional wavelet video coding algorithm called 3-D Embedded Subband Coding with Optimized Truncation (3-D ESCOT). In the proposed ESCOT algorithm, the coefficients in each subband are coded independently using fractional bit-plane coding approach. This feature makes it very easy to achieve frame-rate scalability and resolution scalability for the coded video stream. Moreover, context-based adaptive arithmetic coder with elaborated context assignment as well as global rate-distortion optimization is used in the ESCOT algorithm to achieve high compression efficiency. Compared with 3-D SPIHT algorithm, 3-D ESCOT preserves the scalability of the compressed bit-stream and shows its higher compression performance. Experiments show that, 3-D ESCOT performs measurably and visually better than 3-D SPIHT. Depending on the complexity of motion and video content, the PSNR gain over 3-D SPIHT can be 0.8-1.8dB.

1. INTRODUCTION

Three-dimensional wavelet video coding schemes, as an alternative to standard motion compensated predictive coding approaches, have been investigated recently by several researchers [1, 2, 3, 4]. An important feature of wavelet transforms is the support of scalabilities in the compressed video. With the embedded coding techniques, wavelet video coding achieves continuous rate scalability. Furthermore, because of the multi-resolution nature of wavelet analysis, both resolution scalability and temporal (frame rate) scalability can be easily supported.

As in any video coder, the issue of how to exploit the motion information is an important one in 3-D wavelet coding. Both block- and frame-based motion estimation/compensation methods are investigated. Taubman and Zakhor [1] use a simple pan compensation and Wang et al. [5] use image warping to do the global motion compensation. Ohm [3] and Choi [4] employ block matching motion compensation in 3-D wavelet video coding. The results show that a motion model can improve the compression performance of 3-D wavelet coding significantly.

In this paper, however, we focus on the coding algorithm to deal with the wavelet coefficients generated by any 3-D wavelet transformations. We propose an algorithm called three-dimensional embedded sub-band coding with optimized truncation, or in short 3-D ESCOT.

2. CODING ALGORITHM

A video sequence, due to the delay and storage/computation capability constraint, is normally separated into group of pictures (GOP) for the 3-D wavelet transforms. Encoder and decoder then process the video sequence GOP by GOP, separately and independently.

Pictures in a GOP are decomposed with a 3-D (2D+T) wavelet decomposition scheme. Then the wavelet coefficients are coded to form the compressed bit-stream. Before wavelet decomposition, a global or local motion estimation/ compensation approach may be used to process the pictures to exploit the temporal correlation. Ideally, the coding algorithm should adapt to the motion model too to take further advantage of the temporal correlation.

2.1 Three-dimensional wavelet decomposition

In a typical 3-D wavelet video coder (e.g., [4]), the 2-D spatial transform and the temporal transform are done separately. Frames in a GOP, say with size of 32, are first decomposed in temporal direction. Then spatial decompositions, with mallat or packet decomposition structure, are applied to process the temporal low sub-bands and sometimes even high sub-bands. After wavelet transformation, the coefficients of a special sub-band have their own statistical probability corresponding to the property of that sub-band. Generally, low pass sub-band contains the low frequency signals, thus has more energy. Based on the low/high pass properties of a sub-band, eight types of sub-bands can be obtained: LLL, LLH, LHL, HLL, LHH, HLH, HHL and HHH sub-bands (The three letters stand for the properties in temporal, horizontal and vertical directions respectively, L denotes low pass and H denotes high pass). If a sub-band is low pass in a direction, it normally means that the sub-band contains more correlation in that direction. The coding algorithm should exploit this correlation and use different contexts to code different sub-bands.

2.2 Sub-band coding

After transformation, we code the coefficients of each sub-band independently. There are several reasons to do so.

1. Coding each sub-band independently means that the date in a sub-band can be extracted individually. Therefore, resolution

and frame rate (temporal) scalabilities can be easily implemented.

2. Unlike the EBCOT [7] in JPEG2000, our encoder takes a sub-band as a whole entity. The season is that normally a video frame has lower resolution compared with still image. To guarantee the coding efficiency of the context-based adaptive arithmetic coder, there is no need to split a sub-band further into many small 3-D blocks.

3. Taking a sub-band as a whole entity is also convenient for incorporating possible motion model in the coding process, since there is no motion vector from one sub-band to another sub-band. But within the same 3-D sub-band, the motion vector may point from any coefficients on a temporal plane to any other coefficients on other temporal planes.

2.2.1 Transposition

The coefficients in a sub-band, after transformation and before being coded, are first transposed. The objective of such a transposition is to reduce the number of context models the sub-bands. As described above, after transformation there are eight different types of sub-bands. But because of the symmetry among the temporal, horizontal and vertical directions, the coder does not have to handle so many types of sub-bands. For examples, LLH and LHL sub-band could share the same context model if we could rearrange the horizontal and vertical order. Here transposition is just to rearrange the order of t- y- and x-axis. In the end, there are only four types of sub-bands, namely, HHH, HHL, HLL and LLL sub-bands.

2.2.2 Bit-plane coding

Let $\sigma[i,j,k]$ be a binary-valued state variable, which illustrates the significance of the sample at position [i,j,k]. $\sigma[i,j,k]$ is initialized to 0 and toggled to 1 when the first non-zero bit-plane value of the corresponding sample is encoded. We also define $\chi[i,j,k]$ as the sign of that sample, which is 0 when the sample is positive and 1 when the sample is negative. We use three primitive coding operations to code the bit-plane of the samples.

Zero Coding: When a sample is not yet significant in the previous bit-plane, i.e. $\sigma[i,j,k]=0$, this primitive operation is utilized to code the new information of the sample, which indicates whether the sample become significant or not in the current bit-plane. The Zero Coding (ZC) operation uses the information of the neighbors of the current sample as the context to encode the significance information of the current sample.

Here we consider three categories of neighbors of a sample (see Figure 1):

- Immediate horizontal neighbors. These are the two immediate neighbors of the current sample in the horizontal (i) direction. They are at positions [i-1,j,k] and [i+1,j,k]. We denote the number of these neighbors that are already significant by h, $0 \le h \le 2$.
- Immediate vertical neighbors. These are the two immediate neighbors of the current sample in the vertical (j) direction. They are at positions [i,j-1,k] and [i,j+1,k]. We denote the number of these neighbors that are already significant by v, $0 \le v \le 2$.
- Immediate temporal neighbors. These are the two immediate neighbors of the current sample in the temporal (k) direction. They are at positions [i,j,k-1] and [i,j,k+1]. We denote the number of these neighbors that are already significant by a, $0 \le a \le 2$.
- Immediate diagonal neighbors. They are the 12 neighboring samples that have the least distance from the current sample (except those immediate neighbors). They are samples at diagonal positions, [i,j-1,k-1], [i,j-1,k+1], [i,j+1,k-1], [i,j+1,k+1], [i-1,j,k-1], [i-1,j,k+1], [i+1,j,k-1], [i+1,j,k+1], [i-1,j-1,k], [i-1,j+1,k], [i+1,j-1,k], and [i+1,j+1,k]. We denote the number of these neighbors that are already significant by d, $0 \le d \le 12$.

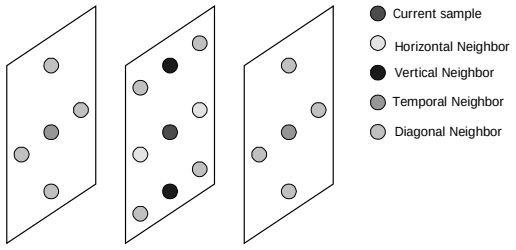

Figure 1: Immediate neighbors of a sample.

The context assignment map for Zero Coding is listed in Table 1. If the conditions of two or more rows are satisfied in the same time, the lower-numbered context is selected. An adaptive context-based arithmetic coder is used to code the significance symbols of Zero Coding.

Table 1: Context assignment map for Zero Coding

LLL and LLH sub-band	h	2	1	1	1	0	0	0	0	0	0	0
	v	x	≥1	0	0	2	1	0	0	0	0	0
	a	x	x	≥1	0	0	0	≥1	0	0	0	0
	d	x	x	x	x	x	x	x	3	2	1	0
	con-text	0	0	1	2	3	4	5	6	7	8	9
LHH sub-band	h	2	1	1	1	1	1	0	0	0	0	0
	v+a	x	≥3	≥1	≥1	0	0	≥3	≥1	≥1	0	0
	d	x	x	≥4	x	≥4	x	x	≥4	x	≥4	x
	con-text	0	0	1	2	3	4	5	6	7	8	9
HHH sub-band	d	≥6	≥4	≥4	≥2	≥2	≥2	≥0	≥0	≥0	≥0	
	h+v+a	x	≥3	x	≥4	≥2	x	≥4	≥2	1	0	
	con-text	0	1	2	3	4	5	6	7	8	9	

Sign Coding: Once a sample becomes significant in the current bit-plane, Sign Coding operation is called to code the sign of the sample. Sign Coding also utilizes an adaptive context-based arithmetic coder to compress the sign symbols. Similarly, we define three quantities, h, v and a, by

$$h=\min\{1, \max\{-1, \sigma[i-1,j,k] \times (1-2\chi[i-1,j,k])+\sigma[i+1,j,k]\times(1-2\chi[i+1,j,k])\}\}$$

$$v=\min\{1, \max\{-1, \sigma[i,j-1,k] \times (1-2\chi[i,j-1,k])+\sigma[i,j+1,k] \times (1-2\chi[i,j+1,k])\}\}$$

$$a=\min\{1, \max\{-1, \sigma[i,j,k-1] \times (1-2\chi[i,j,k-1])+\sigma[i,j,k+1] \times (1-2\chi[i,j,k+1])\}\}$$

The context assignments are showed in Table 2. $\hat{\chi}$ means the sign symbol prediction in the given context. And the symbol sent to the arithmetic coder is $\hat{\chi}$ XOR χ.

Table 2: Context assignment and sign prediction map for Sign Coding

h=-1	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	0	0	0	0	0	0	0	0	0
	context	0	1	2	3	4	5	6	7	8
h=0	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	0	0	0	0	0	1	1	1	1
	context	9	10	11	12	13	12	11	10	9
h=1	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	1	1	1	1	1	1	1	1	1
	context	8	7	6	5	4	3	2	1	0

Magnitude Refinement: Magnitude Refinement (MR) is used to code the new information of a sample that has already become significant in the previous bit-planes. This operation has three contexts. If MR operation is not yet used for this sample, the context is 0. If MR has been used for the sample and the sample has at least one significant immediate neighbor in the previous bit-planes by now, then the context is 1, otherwise the context is 2.

For each bit-plane, the coding procedure consists of three distinct passes, which are applied in turn. Each pass processes a "fractional bit-plane". And in each pass, the scanning order is along i-direction firstly, then j-direction and k-direction lastly. The three passes are described as follows.

1. Significant propagation pass: In this pass, the samples that are not yet significant but have "preferred neighborhood" are processed. We call that a sample has "preferred neighborhood" if and only if the sample has at least a significant immediate diagonal neighbor for HHH sub-band or a significant immediate horizontal, vertical, or temporal neighbor for the other types of sub-bands. For the sample that satisfies these conditions, we apply the ZC primitive to code the symbol in the current bit-plane of this sample. If the sample becomes significant in the current bit-plane, then SC primitive is used to code the sign.

2. Magnitude Refinement pass: We code the samples that have been significant in the previous bit-planes in this pass. And the symbols of these samples in the current bit-plane are coded by MR primitive.

3. Normalization pass: During this pass, those samples that have not yet been coded in the previous two passes are coded. These samples are insignificant. So ZC and SC primitives are applied in this pass.

2.2.3 Global rate-distortion optimization

In the previous stage, each sub-band is coded separately and each forms a independent bitstream. Assume that there are *n* sub-bands: S_1, S_2, …, S_n. Given a specific bit-rate $R_{\max}$, our objective is to construct a final bit-stream which satisfies the bit-rate constraint and makes the overall distortion minimal. As in the EBCOT algorithm [7], the end of each pass at each "fractional" bit-plane is a candidate truncation point. And the value on R-D curve at each candidate truncation point can be obtained because we can calculate the bit length and the distortion reduction at the end of each pass. So we can approximate the R-D curve and find the convex hull of the R-D curve, and truncation can only be performed at the candidate truncation points that are at the convex hull of the R-D curve. The reason to do so is to guarantee that at every truncation point the bit-stream is rate-distortion optimized.

Given a rate-distortion slope threshold λ, one can find those truncation points of a sub-band at which the rate-distortion slopes are greater than λ. To satisfy the bit-rate constraint and make the distortion minimal, the smallest value of λ such that $R_\lambda \le R_{\max}$ is desired. The algorithm to find such threshold can be found in [7].

To achieve quality scalability, a multi-layer bit-stream is formed and each layer represents a quality level. To make a N-layer bit-stream, we first select $\lambda_1 > \lambda_2 > \mathsf{L} > \lambda_N$ which satisfy $R_{\lambda_N} \le R_{\max}$. So with every threshold, we can find a truncation point and obtain a layer of bit-stream from each sub-band. The corresponding layers of all the sub-bands constitute the layers of the final bit-stream. Depending on the available bandwidth and the computational capability, the decoder can select to decode up to the layer it can handle. The fractional bit-plane coding ensures that the bit-stream is finely embedded.

Since we code each sub-band independently, the bit-stream of each sub-band is separable. The encoder can choose to construct a bitstream favoring spatial scalability or temporal scalability. And the decoder can easily extract only a few sub-bands and decode only these sub-bands. Therefore, the implementation of resolution scalability and temporal scalability is natural and easy.

If there exists a motion model involved in the 3D wavelet transforms, the proposed coding algorithm can also take advantage it too. The only adjustment to this algorithm is to change the temporal neighbor relationship. By appointing the temporal neighbor as the corresponding samples along the motion vector, more correlation thus more coding efficiency is expected. On the other hand, object-based coding can be support by skipping the samples out of the object region. All of these demonstrate the great expansibility of the proposed 3-D ESCOT coding algorithm. Details of such extensions will be addressed in a separated paper [8].

3. EXPERIMENTAL RESULTS

To compare the performance of 3-D ESCOT with 3-D SPIHT, low motion (Class A in MPEG-4) sequences "Akiyo" and "Mother & Daughter" at 20/40kbps and high motion (Class B in MPEG-4) sequence "Coast guard" at 40/80kbps are used as our test sequences. All of these sequences are in QCIF resolution sampled at 30fps. Ninety-six (96) frames were coded for all these sequences with GOP of size 32, i.e. first

three GOPs are coded. For each GOP in the sequences, firstly three-level temporal wavelet decompositions are applied and then three-level spatial mallat wavelet decompositions within each transformed frame are applied using Daubechies 9/7 biorthogonal filters. With this specific wavelet decomposition structure, 40 sub-bands are generated in a subband. Table 3 shows some of our experimental results.

Table 3: Comparison of 3-D ESCOT with 3-D SPIHT in term of Average PSNR (dB) for coding Akiyo, Mother & Daughter and Coast guard video sequence.

	Akiyo		Mother & Daughter		Coast guard	
Rate(kbps)	20	40	20	40	40	80
3-D ESCOT	31.32	35.55	31.91	34.90	26.61	28.39
3-D SPIHT	29.43	33.38	30.09	33.33	25.88	27.65
PSNR gain	1.89	2.17	1.82	1.57	0.73	0.74

From Table 3, we note that 3-D ESCOT always performs much better than 3-D SPIHT, especially for the low motion sequences. With Akiyo or Mother & Daughter sequences, the PSNR gain can be as high as 1.57 to 2.17 dB. The relative lower PSNR gain compared with 3-D SPIHT for the Coast guard sequence is probably due to lack of a motion model in our experiments. Because the coding efficiency of the 3-D ESCOT depends more on the spatial and temporal correlations of the wavelet coefficients (via the contexts), but 3-D SPIHT relies on the inter-sub-band correlations of the wavelet coefficients.

The visual effects of the two coding schemes are compared in Figure 2. Note that 3-D ESCOT preserves more detail information than 3-D SPIHT although both are blur due to the low bit-rate coding.

Figure 2: Akiyo frame 11 encoded at 20 kbps using 3-D ESCOT (left) and 3-D SPIHT (right).

Figure 3 compares 3-D ESCOT and 3-D SPIHT schemes in term of PSNR versus frame number at 20kbps for the Mother & Daughter sequence. Clearly, the 3-D ESCOT scheme is significantly better than the 3-D SPIHT. The PSNR drops somewhat abruptly at the beginning and the end of each GOP's video both for 3-D ESCOT and 3-D SPIHT. We have investigated this problem and found that this is due to the boundary effect of the wavelet transformation within a limited length of GOP. Further study is needed to solve this common problem in 3-D wavelet coding.

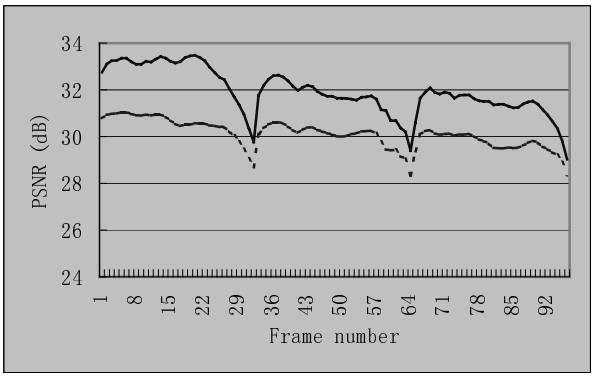

Figure 3: PSNR vs. frame number (1-96) for "Mother & Daughter" at 20kbps by 3-D ESCOT (solid line) and 3-D SPIHT (dash line).

4. SUMMARY

In this paper, we proposed a new three-dimensional wavelet video coding algorithm, namely 3-D ESCOT. Independent sub-band coding and multi-layer coding enable 3-D ESCOT with many features, such as flexible temporal, spatial, rate scalabilities. Compared with 3-D SPIHT, the coding efficiency advantage of 3-D ESCOT is significant both measurably and visually. Moreover, the flexibility of our coding algorithm makes it easy to be used in various three-dimensional coding systems, with or without motion model, frame-based or object-based. The boundary effect of 3-D wavelet coding for a limited length GOP still needs to be solved.

5. REFERENCES

[1] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 572-589, Sept. 1994.

[2] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," IEEE Trans. Image Processing, vol. 3, no. 5, pp. 559-571, Sept. 1994.

[3] S. J. Choi and J. W. Woods, " Motion-compensated 3-D subband coding of video," IEEE Trans. Image Processing, vol. 8, pp. 155-167, Feb. 1999.

[4] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Very low bit-rate embedded video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," to appear in *IEEE Trans. Circuits and Systems for video Technology*, 2000.

[5] A. Wang, Z. Xiong, P. A. Chou, and S. Mehrotra, "Three-dimensional wavelet coding of video with global motion compensation," *Proc. DCC'99*, Snowbird, UT, Mar. 1999.

[6] B.-J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," *Proc. DCC'97*, 1997.

[7] D. Taubman, "JPEG2000 Verification Model: Version VM4.1", ISO/IEC JTC 1/SC 29/WG1 N1286.

[8] J.-Z. Xu, S. Li and Y.-Q. Zhang, "Three-dimensional shape-adaptive discrete wavelet transforms for efficient object-based video coding," to be appeared in Proceedings of SPIE VCIP'2000, Perth, Australia, July, 2000.