MEMORY-CONSTRAINED 3D WAVELET TRANSFORMS FOR VIDEO CODING WITHOUT BOUNDARY EFFECTS

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

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

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

ABSTRACT

3-D wavelet-based scalable video coding provides a viable alternative to standard MC-DCT coding. However, many current 3-D wavelet coders experience severe boundary effects across GOP boundaries. This paper proposes a memory efficient transform technique via lifting that effectively computes wavelet transforms of a video sequence continuously on the fly, thus eliminating the boundary effects due to limited length of individual GOPs. Coding results show that the proposed scheme completely eliminates the boundary effects and gives superb video playback quality.

1. INTRODUCTION

As an alternative to predictive approaches in video coding standards (e.g., H.261/3 and MPEG-1/2/4), 3-D wavelet video coding has been investigated recently by several researchers [1, 2, 3, 4]. The main advantage of 3-D wavelet video coding is its scalabilities (including, rate, PSNR, spatial and temporal). The obvious applications are video delivery over heterogeneous networks (e.g., the Internet) and future wireless video services, where the encoder should be able to seamlessly adapt to different channel conditions, such as bandwidth fluctuation and packet errors/losses, and the decoder should be able to adapt to different computational resources. The latest results indicate that 3-D wavelet coding outperforms MPEG-2 or HDTV at high bit rates (>2 Mbps) while being slightly worse than H.263+ at low bit rates (<40 kbps).

However, there is still a problem common in many current 3-D wavelet coders. That is, frame quality or PSNR drops severely at the boundaries of group of pictures (GOP), sometimes up to several decibels. This results in very annoying jittering artifacts in video playback. The reason for this is limited GOP length due to delay or memory constraints. With enough memory, one could potentially buffer the whole video sequence and process it as a whole in 3-D wavelet transform and bit-plane coding.

In this paper, we propose a lifting-based scheme[6] that utilizes the locality property of wavelet transforms to implement the memory-constrained wavelet analysis and synthesis. A finite buffer is used to process one part of the sequence at a time continuously — creating the effect of having infinite memory — by buffering coefficients at intermediate lifting steps towards the end of one GOP and finishing the job until intermediate coefficients from beginning of the next GOP are available. This is much like the classic “Overlap-save” approach in Oppenheim & Schafer[7] to implementing DFTs of 1-D (e.g., speech) signals. Similar filter-bank factorization based approach for discrete wavelet transform was also used in [8, 9]. Our proposed wavelet transform scheme does not physically break the sequence into GOPs but processes the sequence without intermission, so the boundary effect can be completely eliminated. Moreover, the required buffering in our implementation is very small and the proposed approach can be used to implement other decomposition structures (e.g., Spacl and Packet in [10].) Coding results show that the proposed scheme indeed gives superb video playback quality without any boundary effects.

2. MEMORY-CONSTRAINED 3D WAVELET TRANSFORMS VIA LIFTING

In a typical 3-D wavelet video coder (e.g., [4]), the 2-D spatial transform and the temporal transform are done separately. This allows us to focus only on the 1-D temporal transform for artifact elimination while maintaining the traditional approach to computing the spatial transform with symmetric extensions over the boundaries. Without loss of generality, we will assume in the sequel that the Daubechies 9/7 biorthogonal filters are used (shorter filters can be handled more easily with less memory.)

2.1 One-level wavelet decomposition

It was shown in [11] that every FIR wavelet or filter bank can be decomposed into lifting steps (see Fig. 1) and that each lifting step can be further split into elementary operations.

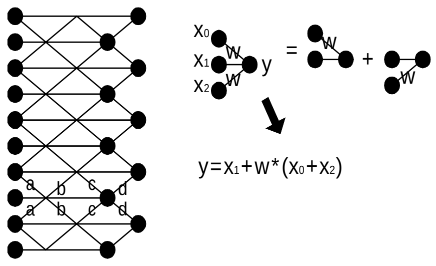

Figure 1. Temporal wavelet decomposition via lifting,each node denotes a frame. The nodes of left column are input frames and the right ones are output frames. High pass and low pass frames are output alternately.

From Fig. 1, we also note that, for a one-level wavelet decomposition, each output frame is at most related to the next four frames. Thus, a minimum buffer of five frames is needed. Our algorithm uses exactly five frames for buffering. We push video frames into the buffer one by one and output the wavelet transform frame immediately when it is available. Let B0, B1,..., B4 denote the buffered five frames. Our proposed wavelet transform algorithm is as follows:

- **Initialization**: The first five frames are pushed into the buffer, and the first wavelet frame is computed and output (symmetric extension is used at the left boundary).
- **Pipeline computation**: After initialization, the input frames can be processed in a pipeline. That is, once a wavelet frame is output, a buffer frame can be released and a new frame can be pushed into the buffer, a new wavelet frame can then be computed. Specifically, once getting a frame, we perform a buffer updating:

  $B0 \leftarrow B1$,
  $B1 \leftarrow B2$,
  $B2 \leftarrow B3$,
  $B3 \leftarrow B4$,
  $B4 \leftarrow$ a new frame.

  If the input frame is odd-numbered, we perform the following elementary operations:

  $B4 \leftarrow B4+\alpha*B3$,
  $B3 \leftarrow B3+\beta*B2$,
  $B2 \leftarrow B2+\gamma*B1$,
  $B1 \leftarrow B1+\delta*B0$
  Output B0;

  otherwise, we perform the following:

  $B3 \leftarrow B3+\alpha*B4$,
  $B2 \leftarrow B2+\beta*B3$,
  $B1 \leftarrow B1+\gamma*B2$,
  $B0 \leftarrow B0+\delta*B1$,
  Output B0;

  where α, β, γ and δ are the lifting factors corresponding to the 9/7 filters in [11].
- **Flush**: When the last frame is pushed into the buffer, the last five wavelet frames can be computed and output (symmetric extension is also used at the right boundary.)

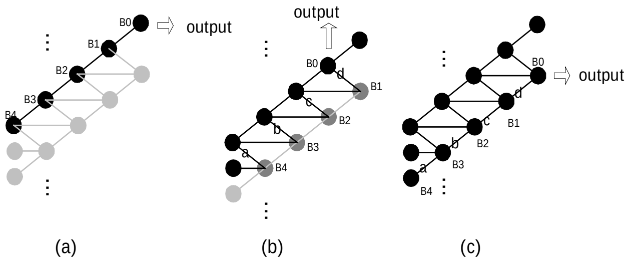

Figure 2. Pipeline computation. (a) illustrates the buffers' state before a frame is pushed in. (b) and (c) show the operations when an odd- or an even-number frame is pushed into the buffers respectively.

Figure 2 shows these operations. Black nodes represent frames that have been processed, while gray ones correspond to those that have not been fully processed. Similarly, black lines represent processed operations, while gray ones correspond to those that are not processed yet Note that, in each group of operations, the content of B0 is output before reuse. Also, because B0 is output in the previous operation, this pipeline process can proceed correctly with only a five-frame buffer.

2.2 N-level wavelet decomposition

For an N-level decomposition, we use a ***push*** model in which input frames in one level are pushed into the buffer of that level and once an output is ready, it is pushed into the next level buffer or the final output buffer. One method is to allow each decomposition level to use its own independent buffer and to process decomposition levels sequentially, i.e., for multi-level decomposition, we use the output of one level as the input to the next level. For example, in a two-level Mallat (dyadic) decomposition, the high pass frames of level one are output directly and the low pass ones are pushed into the level two buffer. Since a five-frame buffer is needed for each level, an N-level Mallat wavelet decomposition will need a 5N-frames buffer. The buffer sizes for other decomposition structures are listed in Table 1.

Table 1. Buffer requirements (in terms of frames) for different decomposition structures with the independent-buffer method.

Level	Mallat	Spacl	Packet
1	5	-	-
2	10	15	-
3	15	20	35
4	20	25	40

Another method is to use shared buffer. That is, all the buffers are allocated at the beginning of transforms and all decomposition levels share a common buffer. To avoid inter-level interference, more space should be allocated. Fig. 3 shows a two-level Mallat decomposition structure, in which B1 to B12 are not changed until the node *i* is ready for output. Once B0 is ready, B0 to B3 can be output one by one and the memory will be available for reuse. With this method, if we define Buf(i) as the minimal buffer requirements (in terms of frames) we need to implement an i-level decomposition, then Buf(i) is given by the following recursive formula:

$$Buf(i+1)=Buf(i)+2^{i+2},\ i\in Z^{+}, \text{ with } Buf(0)=1.$$

The buffer requirements for the Spacl, Packet or other decomposition structures are the same because they are determined only by the decomposition level and by the filter lengths used. Table 2 summarizes these requirements.

Table 2. Buffer requirements (in terms of frames) for different decomposition structures with the shared-buffer method.

Level	Mallat	Spacl	Packet
1	5	-	-
2	13	13	-
3	29	29	29
4	61	61	61

From Tables 1 and 2, we note that the independent-buffer method is more memory efficient in most cases. However, a wavelet frame is immediately output once it is ready in this method, so the output order is irregular. That is: low-level highpass frames are always output in advance of its original order compared with other wavelet frames due to the process delay. This makes it improper for coding algorithms like 3-D SPIHT [4] with order requirements (3-D SPIHT exploits inter-band correlation so the wavelet frames should be rearranged according to sub-bands as in Fig. 3.) We have to use extra buffers for wavelet frame ordering in this case. In the shared-buffer method, however, the extra buffers can also be used for frame rearrangement, thus they can guarantee the required order. In addition, the shared-buffer method is more suitable for other decomposition structures. In fact the buffer requirement is the same for different decompositions (see Table 2). Finally, both methods give the same delay in the wavelet transform. For example, the delay is four frames for one-level decomposition and 12 frames for two-level decomposition. Since one output frame is related to four frames in one-level decomposition and 12 frames in two-level decomposition, these are minimum delays.

2.3 Wavelet synthesis

The synthesis structure is much like the analysis structure. The only difference is that the former uses a ***pull*** model, which means that a request is sent whenever a wavelet frame is needed and that the synthesis algorithm decides which frames should be loaded into the buffers. The reason for doing so is because the requests are in natural order while the inputs are not. Depending on the buffering method used in the wavelet analysis, there are two ways for implementing an N-level wavelet synthesis: independent-buffer method and shared-buffer method. The buffer requirement and delay are the same as in the analysis case.

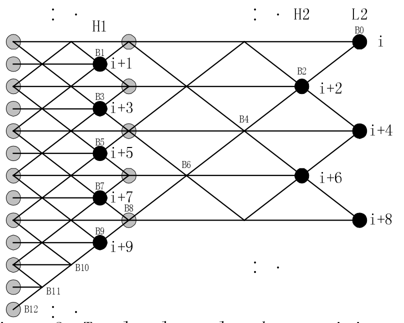

Figure 3. Two-level wavelet decomposition with shared buffer. The black nodes are the final decomposition output nodes and the numbers beside these nodes show their output order.

3. EXPERIMENTAL RESULTS

To test our proposed memory-constrained wavelet transform algorithm, we applied our algorithm on a 288-frame QCIF "Akiyo" (30fps) test sequence with a three-level Mallat temporal decomposition followed by a three-level spatial decomposition for each frame. The shared-memory method is used with a 29-frame memory. The wavelet frames are exactly the same as those obtained by using the conventional transform algorithm that buffers the whole sequence (all 288 frames.)

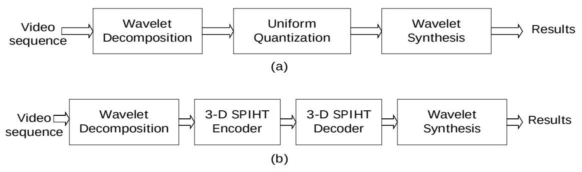

Figure 4. Diagrams of two experiment enviroments.

To compare our proposed transform scheme with the conventional transform scheme in real coding scenarios, we set up two coding experiments: uniform quantization and coding and 3-D SPIHT coding. The diagrams of these two coders are showed in Fig. 4(a) and (b), respectively. When the proposed transform scheme is used, we divide the transformed frames into GOPs. Note that doing so will not introduce any boundary effect. When the conventional transform scheme is used, however, we have to divide the frames into GOPs *before* wavelet transform due to memory constraint. The transform structure is the same in both cases (three-level temporal followed by three-level spatial.) After lossy coding (uniform quantization and coding or 3-D SPIHT coding), inverse transform is used to decode the sequence.

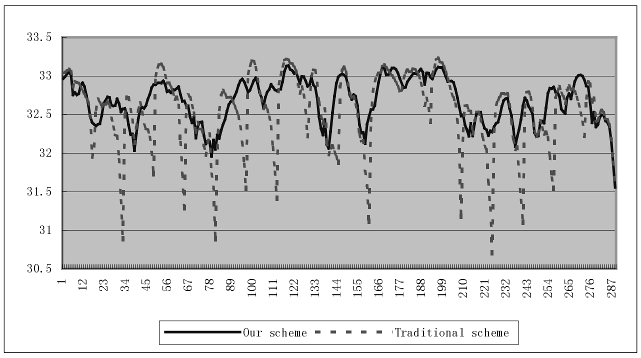

Figure 5. PSNR comparison of two schemes with uniform quantization.

Fig 5. shows PSNR curves from the two transform schemes using uniform quantization (stepsize=64) and coding. PSNR curves from the two schemes using 3-D SPIHT coding at 20 kbps and 40 kbps are shown in Figs. 6 and 7. Uniform bit rate allocation is used for different GOPs in 3-D SPIHT. From these figures, we can see that our proposed scheme solves the PSNR dipping problem at GOP boundaries. The overall average PSNR is about 0.2dB higher than that corresponding to the conventional scheme. In video playback, the proposed transform scheme gives much smoother and better visual quality because the boundary effects are completely eliminated.

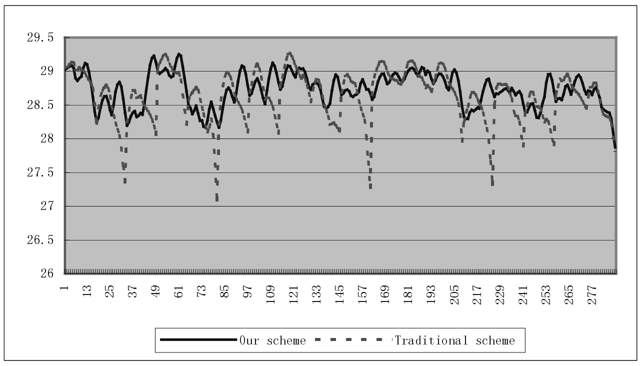

Figure 6. PSNR comparison of two schemes with 3-D SPIHT in 20kbps bit-rate.

4. CONCLUSIONS

This paper presents a 3-D wavelet transform scheme with very small memory and minimal delay. More importantly, the proposed scheme completely eliminates boundary effects in conventional 3-D wavelet video coders. The visual quality of decoded video thus improves substantially with this scheme, making 3-D wavelet coding more competitive to standard MC-DCT video coding.

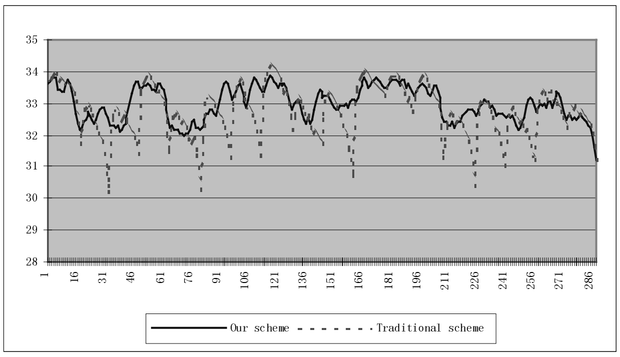

Figure 7. PSNR comparison of two schemes with 3-D SPIHT in 40kbps bit-rate.

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] W. Sweldens and P. Schroder, "Building your own wavelets at home," ACM SIGGRAPH *Course notes on Wavelets in Computer Graphics*, pp. 15-87, 1996.
[7] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.
[8] W. Jiang and A. Ortega, "Efficient discrete wavelet transform architectures based on filterbank factorizations," *Proc. ICIP'99*, Kobe, Japan, October 1999.
[9] W.Jiang and A. Ortega, "Parallel architecture for the discrete wavelet transform based on the lifting factorization," *Proc. SPIE Conf. in Parallel and Distributed Methods for Image Processing III*, Denver, CO, July 1999.
[10] D. Taubman, "High performance scalable image compression with EBCOT", to appear *IEEE Trans. Image Proc*, 2000.
[11] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl*, vol. 4, pp. 247-269, 1998.