On Boundary Effects in 3-D Wavelet Video Coding*

Jizheng Xu[†], *Zixiang Xiong*[‡], *Shipeng Li*[†], *and Ya-Qin Zhang*[†]
[†]Microsoft Research, China, No. 49, Zhichun Rd, Haidian District, Beijing, 100080
[‡]Dept of Electrical Engineering, Texas A&M University, College Station, TX 77843
Tel: (979) 862-8683; Fax: (979) 862-4630; Email: zx@lena.tamu.edu

July 2, 2000

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 group of picture (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.[1–5] The main advantage of 3-D wavelet video coding is its rate, PSNR, spatial, and temporal scalabilities. 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 (GOPs), sometimes up to several decibels. This results in very annoying jittering artifacts during video playback. The reason for this is

*Z. Xiong's work was supported in part by the NSF CAREER grant 0096070 and by a grant from the Microsoft Corporation.

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 GOP boundary effects in 3-D wavelet video coding

In a typical 3-D wavelet video coder (e.g.,[4,5]), the 2-D spatial transform and the temporal transform are done separately, i.e., the spatial and temporal transforms are decoupled. This allows us to focus only on the 1-D temporal wavelet transform for the analysis in this section. We exclude common orthogonal transforms like DFT, DCT because they do not easily offer frame rate scalability and spatial resolution scalability like the wavelet transform does. Furthermore, we only consider biorthogonal wavelet transforms with symmetric extensions over GOP boundaries because they perform better in our coding experiments than orthogonal wavelet transforms with periodic extensions over GOP boundaries or boundary filters.[11]

Let $\mathbf{x} = \{x_0, ..., x_{N-1}\}$ be a length-N signal and $\mathbf{X} = \{X_0, ..., X_{N-1}\}$ be its biorthogonal wavelet transform coefficients (assuming symmetric extensions over the boundaries in the wavelet transform.) Denote the corresponding N basis vectors as $\mathbf{b_i} = \{b_{i,0}, ..., b_{i,N-1}\}$, $i = 0, ..., N-1$. Assume uniform quantization of wavelet coefficients X_i with the quantization errors $E_i = X_i - Q(X_i)$ being i.i.d. with zero mean and variance σ^2 for all $i = 0, ..., N-1$, we have the time-domain error vector after inverse wavelet transform as:

$$\epsilon_{\mathbf{x}} = \{\epsilon_0, ..., \epsilon_{N-1}\} = \sum_{i=0}^{N-1} E_i \mathbf{b_i}$$

with

$$\epsilon_j = \sum_{i=0}^{N-1} E_i b_{i,j}, \;\; j = 0, ..., N-1.$$

Then, the mean-squared error at x_j in the time domain will be:

$$||\epsilon_j||^2 = \sigma^2 \sum_{i=0}^{N-1} b_{i,j}^2, \;\; j = 0, ..., N-1, \tag{1}$$

where we have used the i.i.d. property of E_i to obtain the above expression. Fig. 1 shows the boundary effects by plotting $k - 10\log||\epsilon_j||^2$ (PSNR in dB) for $N = 16$ and $N = 32$, where $k = 10\log\frac{255^2}{\sigma^2}$ is a constant (assumed to be 30). Three levels of Mallat (dyadic) wavelet transforms are used with the Daubechies 9/7 filters in both cases. The plot for $N = 16$ is repeated once in Fig. 1, which clearly depicts the boundary effects (wide range fluctuation in PSNR across boundaries) if one compresses a length-32 signal in two separate length-16 segments[1].

One possible way to alleviate the boundary effects is to use scaling prior to uniform quantization (or equivalently non-uniform quantization.) Instead of quantizing the original wavelet coefficients $\{X_0, ..., X_{N-1}\}$, we now apply uniform quantization on $\{\frac{X_0}{\alpha_0}, ..., \frac{X_{N-1}}{\alpha_{N-1}}\}$, where the α_i's are scaling factors. After inverse quantization and inverse scaling, the mean-squared error at x_j in this case will be:

$$||\epsilon_j||^2 = \sigma^2 \sum_{i=0}^{N-1} \alpha_i^2 b_{i,j}^2, \;\; j = 0, ..., N-1.$$

The aim here is to find a set of α_i^2's in the above equation such that $||\epsilon_j||^2$ is a constant for all $j = 0, ..., N-1$, which will eliminate PSNR variations. It turns out that there is no such solution for a set of odd-length biorthogonal filters including the popular Daubechies 9/7 filters. A possible remedy is to use adaptive boundary filters[12] at signal boundaries. The drawback of this approach is the extra buffer and computation needed to handle the boundary filters. The wavelet transform scheme presented in the next section effectively handles infinitely long signals (or video sequences) with finite memory, thus eliminating boundary effects across GOP boundaries.

3 Memory-constrained 3-D wavelet transform via lifting

Due to the decoupling of spatial and temporal transforms, we again focus on the 1-D temporal transform for artifact elimination while maintaining the traditional filtering or lifting approach to computing the spatial transform with symmetric extensions over the boundaries. Without loss of generality, we will assume in the sequel that the 9/7 biorthogonal filters are used (shorter filters can be handled more easily with less memory.)

3.1 One-level wavelet decomposition

It was shown in[13] that every FIR wavelet or filter bank can be decomposed into lifting steps and that each lifting step can be further split into elementary operations. Specifically,[13] gives the following factorization of the dual polyphase matrix $\tilde{P}(z)$ corresponding to the Daubechies 9/7 biorthogonal filters:

$$\tilde{P}(z) = \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{bmatrix} \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix},$$

[1]If orthogonal wavelet transform is used, then $||\epsilon_j||^2$ in Eq. (1) will be a constant for all $j = 0, ..., N-1$, i.e., there will be no boundary effects. This is confirmed in our experiments with high bit rate coding. But the overall coding performance in this case is worse than that from using biorthogonal wavelet transforms at the same bit rate. In addition, when the bit rate is low, the i.i.d. assumption of E_i is not true any more, and we observe even worse boundary effects than using biorthogonal wavelet transforms.

where $\alpha = -1.5861, \beta = -0.05298, \gamma = 0.8829$, $\delta = 0.4435$, and $\zeta = 1.1496$ are the lifting factors. The above factorization leads to a standard five-step lifting implementation as shown in Fig. 2, where symmetric extension is assumed over the boundaries of a 1-D signal of length ten. From Fig. 2, we see that, for one-level wavelet decomposition, each output coefficient is at most related to the next four samples (given current and past samples are already available). Thus, a minimum buffer of five samples (the current sample plus four future samples) is needed if the memory is limited.

Based on this observation, we propose a memory-constrained algorithm to compute 1-D temporal wavelet transform of a long video sequence. Our algorithm uses exactly five frames for buffering (each sample in Fig. 2 now corresponds to a video frame.) We push video frames into the buffer one by one and output a wavelet transform frame immediately whenever it is available. More specifically, 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 (symmetric extension is used at the left boundary.)
- **Pipeline computation**: After initialization, the input frames can be processed in a pipeline, which means that, once a frame is pushed into the buffer, a new wavelet frame can be computed and a buffer frame can be released. Specifically, once getting a frame, we perform a buffer updating:

  $$\begin{aligned}&B0 \leftarrow B1,\\&B1 \leftarrow B2,\\&B2 \leftarrow B3,\\&B3 \leftarrow B4,\\&B4 \leftarrow \text{a new frame.}\end{aligned}$$

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

  $$\begin{aligned}&B4 \leftarrow B4 + \alpha * B3,\\&B3 \leftarrow B3 + \beta * B2,\\&B2 \leftarrow B2 + \gamma * B1,\\&B1 \leftarrow B1 + \delta * B0,\\&\text{output } B0,\end{aligned}$$

  otherwise, we perform the following:

  $$\begin{aligned}&B3 \leftarrow B3 + \alpha * B4,\\&B2 \leftarrow B2 + \beta * B3,\\&B1 \leftarrow B1 + \gamma * B2,\\&B0 \leftarrow B0 + \delta * B1,\\&\text{output } B0.\end{aligned}$$

- **Flush**: When the last frame is pushed into the buffer, the last five wavelet frames can be computed (symmetric extension is also used at the right boundary.)

Fig. 3 shows the pipeline implementation of our proposed algorithm. Note that, in each group of operations, the content of $B0$ is discarded before reuse. Also, because $B0$ is output in the previous operation, this pipeline process can proceed correctly with only a five-frame buffer.

3.2 L-level wavelet decomposition

For an L-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 its own independent buffer and to process decomposition levels sequentially, i.e., for multi-level decompositions, we use the output of one level as the input to the next level. For example, in a two-level Mallat decomposition, the highpass frames of level one are output directly and the lowpass ones are pushed into the level two buffer. Since a five-frame buffer is needed for each level, an L-level Mallat wavelet decomposition will need a $5N$-frame buffer. The buffer sizes for other decomposition structures are listed in Table 1.

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

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

Another method is to use a shared buffer. That is, a common buffer is allocated at the beginning for sharing by all decomposition levels. To avoid inter-level interference, more space should be allocated. Fig. 4 shows a two-level Mallat decomposition structure, in which $B1$ to $B12$ are not changed until $B0$ is ready for output. Once it is ready, $B0$ to $B3$ can be output one by one and the memory will be available for reuse. Thus, to output one lowpass frame in the second-level wavelet transform, we need five first-level lowpass wavelet frames; but because of the interleaving nature of wavelet frames, we have to buffer the four first-level highpass wavelet frames in between, hence, we actually compute nine first-level wavelet frames, which in turn require a buffer of 13 (nine plus four) original frames. In summary, if we define $Buf(i)$ as the minimal buffer requirement in terms of number of frames for implementing an i-level wavelet decomposition with this method, then $Buf(i)$ can be calculated from the following recursive formulae:

$$Buf(i+1) = Buf(i) + Buf(i) - 1 + 4 = 2Buf(i) + 3, i \in Z^+, \text{ with } Buf(0) = 1.$$

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

From Tables 1 and 2, we see that the independent-buffer method is more memory efficient in most cases. However, a wavelet frame is 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 ESCOT[4] with order requirements (3-D ESCOT exploits inter-band correlation so the wavelet frames should be rearranged according to subband rather than interleaved as in Fig. 4.) We have to use extra buffering for wavelet frame ordering in this case. In the shared-buffer method, however, the extra buffer can also be used for frame rearrangement, thus guaranteeing the required order. In addition, the shared-buffer method is more suitable for other decomposition structures. In fact the buffer requirements are the same for different decompositions (see Fig. 4). Finally, both methods give the same minimum delay in the wavelet transform. For example, the delay is four frames for one-level decomposition and 12 frames for two-level decomposition.

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

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

3.3 Wavelet synthesis

The synthesis structure is much like the analysis one. The only difference is that we now use 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 in the wavelet analysis, there are again two ways for implementing an L-level wavelet synthesis: independent-buffer method and shared-buffer method. The buffer and delay requirements are the same as in the analysis case.

4 Experimental results

To test our proposed memory-constrained wavelet transform algorithm, we applied it on the QCIF "Akiyo" and "Coast_guard" test sequences (288 frames) with a three-level temporal decomposition followed by a three-level spatial decomposition on each resulting 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.)

We use the embedded 3-D ESCOT coder[5] to compare our proposed transform scheme with independent GOP transform in real coding scenarios. When the proposed transform scheme is used, we separate frames into GOPs of 16-frames *after* wavelet transform. Note that doing so will not introduce any boundary effect. When independent GOP transforms are 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 ESCOT coding), inverse transform is used to decode the sequence.

Fig. 5 shows PSNR curves from the two transform schemes using 3-D ESCOT coding at 40.846 kbps and 44.213 kbps for "Akiyo" and "Coast_guard", respectively. Uniform bit rate allocation is used over different GOPs in 3-D ESCOT. From Fig. 5, we can see that the manifestation of boundary effects, i.e., the pattern of PSNR variation, in the independent GOP coding case matches that in Fig.

1 from the analysis in section 2. In contrast, our proposed transform scheme solves the PSNR dipping problem across GOP boundaries. The overall PSNR curve is smoother, with average PSNR being about 0.35 to 0.48 dB higher than that corresponding to independent GOP coding. In video playback, the proposed transform scheme gives much better visual quality because the boundary effects are completely eliminated.

For comparison purposes, we use MPEG-4 to compress the two test sequences under the same experimental conditions (we actually run MPEG-4 first to obtain a target bit rate for 3-D ESCOT to match since the latter coder is embedded.) MPEG-4 (MS version 16.0) gives an average of 34.18 dB for "Akiyo" at 40.846 kbps and 26.04 dB for "Coast_guard" at 44.213 kbps, respectively. These average PSNRs are about 0.6 and 1 dB worse than those shown in Fig. 5. Finally, we point out that our proposed new transform scheme offers coding improvements that are independent of the actual coding method. Thus, it is equally applicable to other coders like 3-D SPIHT.[4]

5 Conclusions

This paper presents a 3-D wavelet transform scheme with very low memory and minimal delay. More importantly, the proposed scheme completely eliminates boundary effects in 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.

6 REFERENCES

[1] D. Taubman and A. Zakhor, "Multirate 3-D subband coding of video," *IEEE Trans. Image Processing*, vol. 3, pp. 572–589, September 1994.

[2] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Processing*, vol. 3, pp. 572–589, September 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, February 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," to appear in *IEEE Trans. Circuits and Systems for Video Technology*, 2000.

[5] J. Xu, S. Li, Y.-Q. Zhang, and Z. Xiong, "A wavelet video coder using 3-D embedded subband coding with optimized truncation (3-D ESCOT)", submitted to ISMIP'00, Sydney, Australia, December 2000.

[6] W. Sweldens and P. Schröder, "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 Processing*, 2000.

[11] C. Herley, "Boundary filters for finite-length signals and time-varying filter banks," *IEEE Trans. Circuits and Systems*, vol. 42, pp. 102–144, February 1995.

[12] A. Mertins, "Optimized biorthogonal shape adaptive wavelets," *Proc. ICIP'98*, Chicago, IL, October 1998.

[13] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, pp. 247-269, 1998.

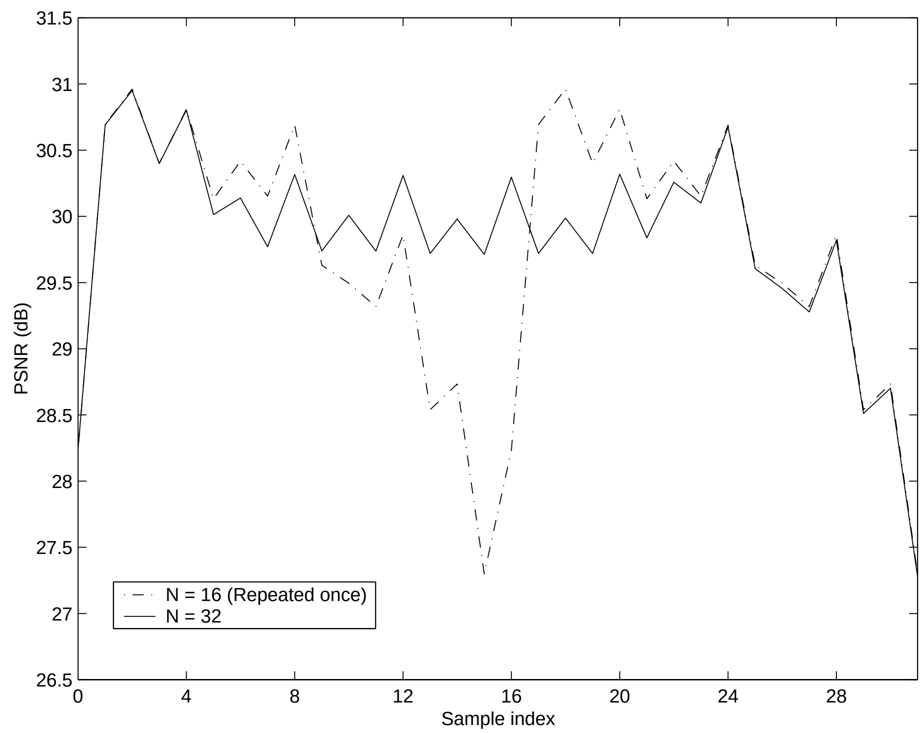


Figure 1: Boundary effects in terms of PSNR variations (in dB) as seen analytically from coding a length-32 signal as a whole vs. coding it as two separate length-16 segments using three-level Mallat (dyadic) wavelet transform and uniform quantization. The Daubechies 9/7 biorthogonal wavelet filters are used.

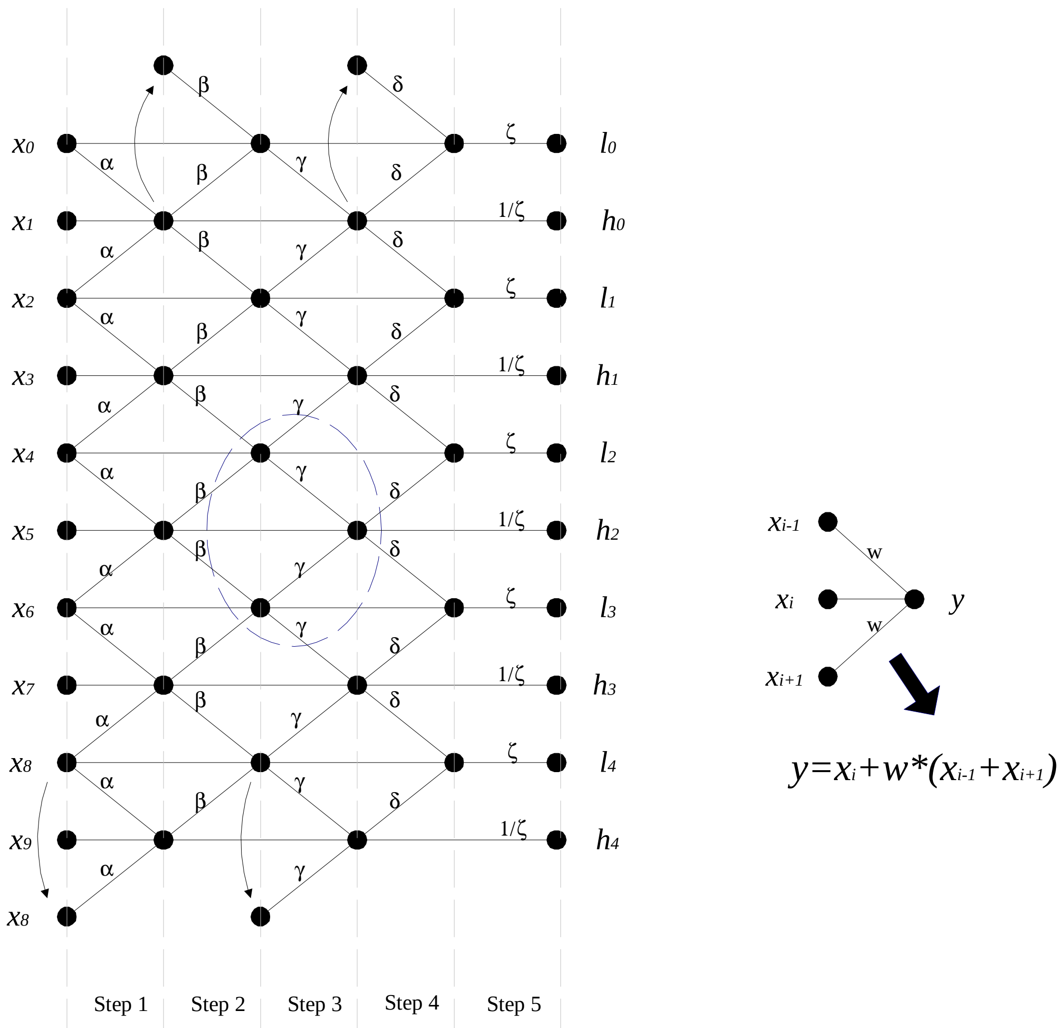


Figure 2: Standard five-step lifting implementation of the Daubechies 9/7 biorthogonal wavelet transform for a 1-D signal of length ten. Symmetric extensions are used over signal boundaries. Except the last scaling step, each basic lifting step involves elementary operations shown in the right.

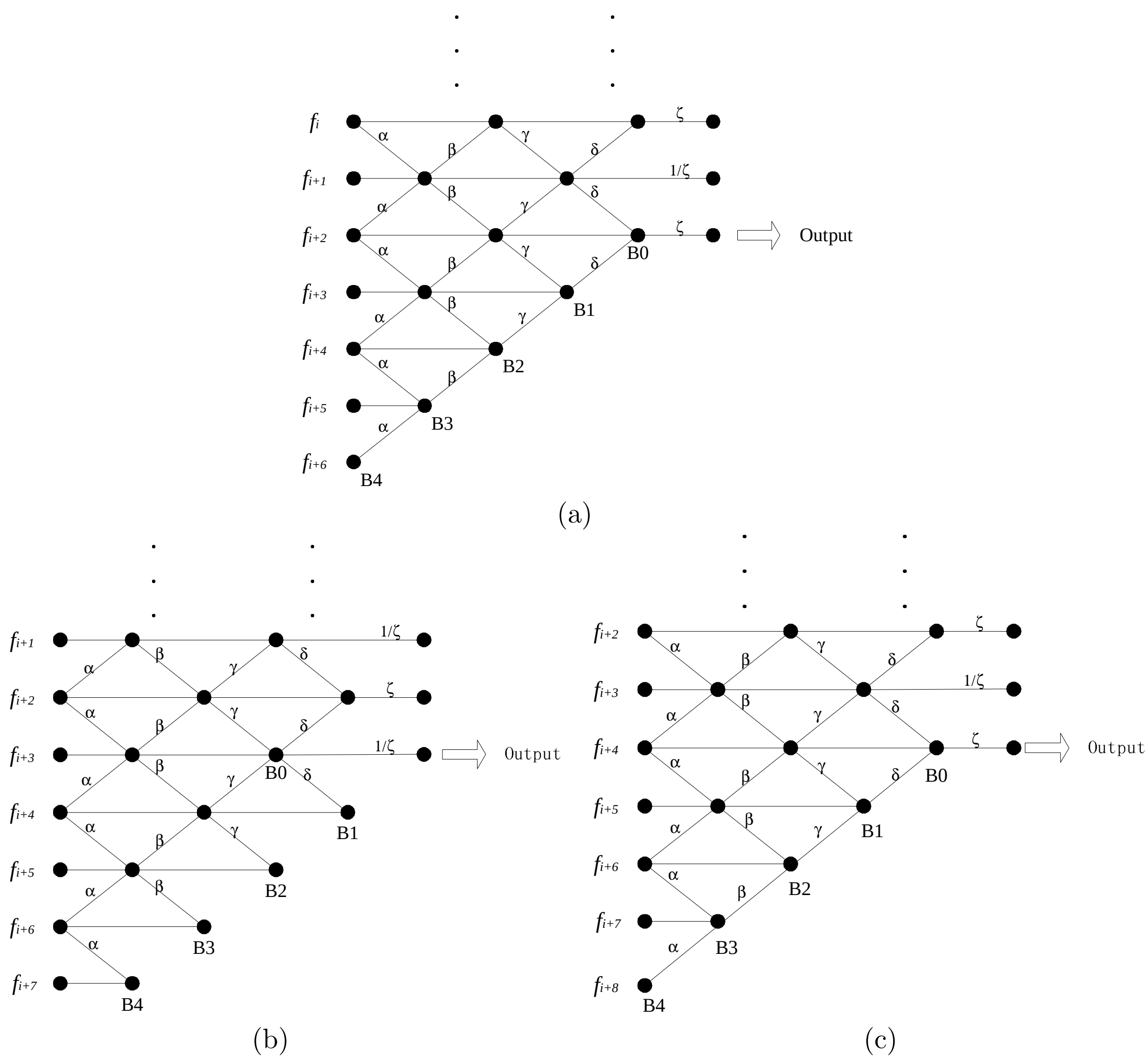


Figure 3: Pineline implementation. (a) An illustration of the buffer state before a frame is pushed into the buffer. (b) and (c) show the pipeline operations when an odd- or even-numbered frame is pushed into the buffer, respectively.

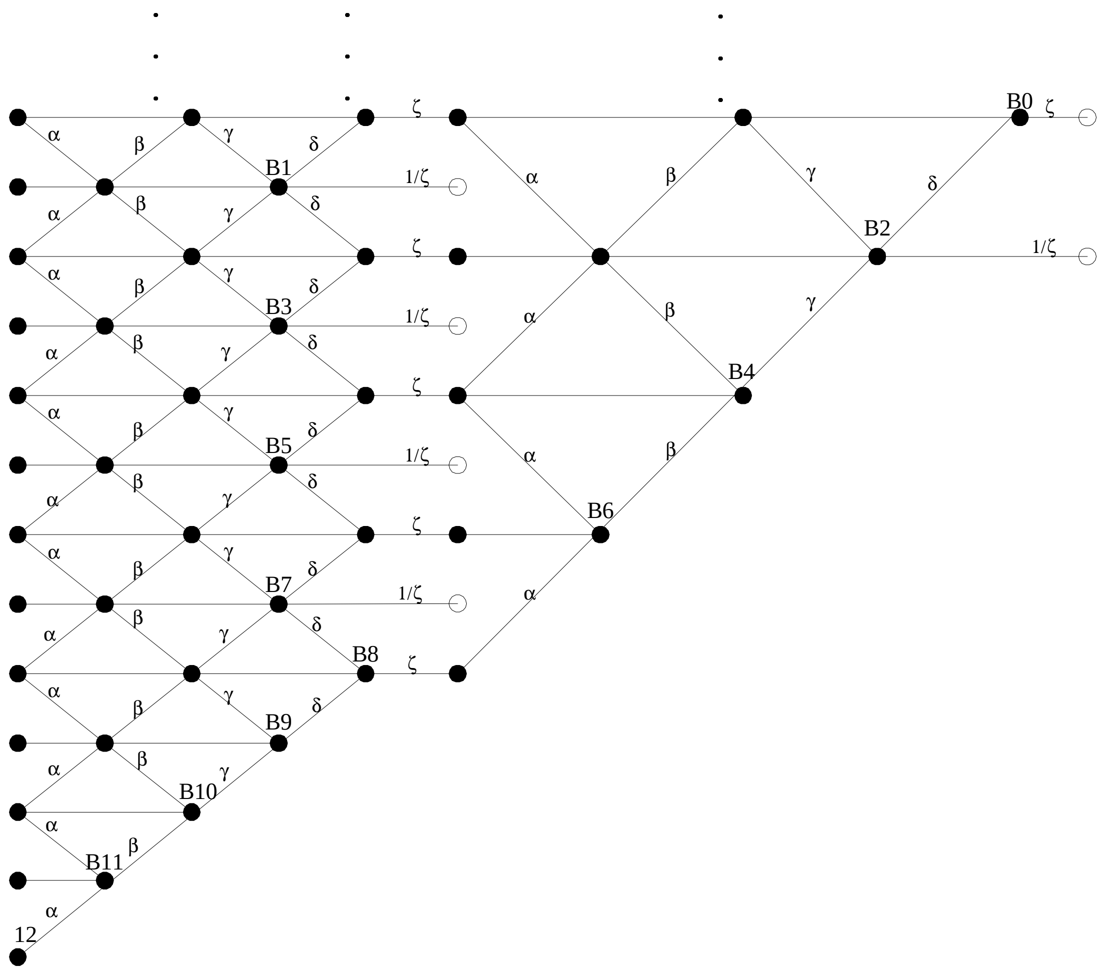
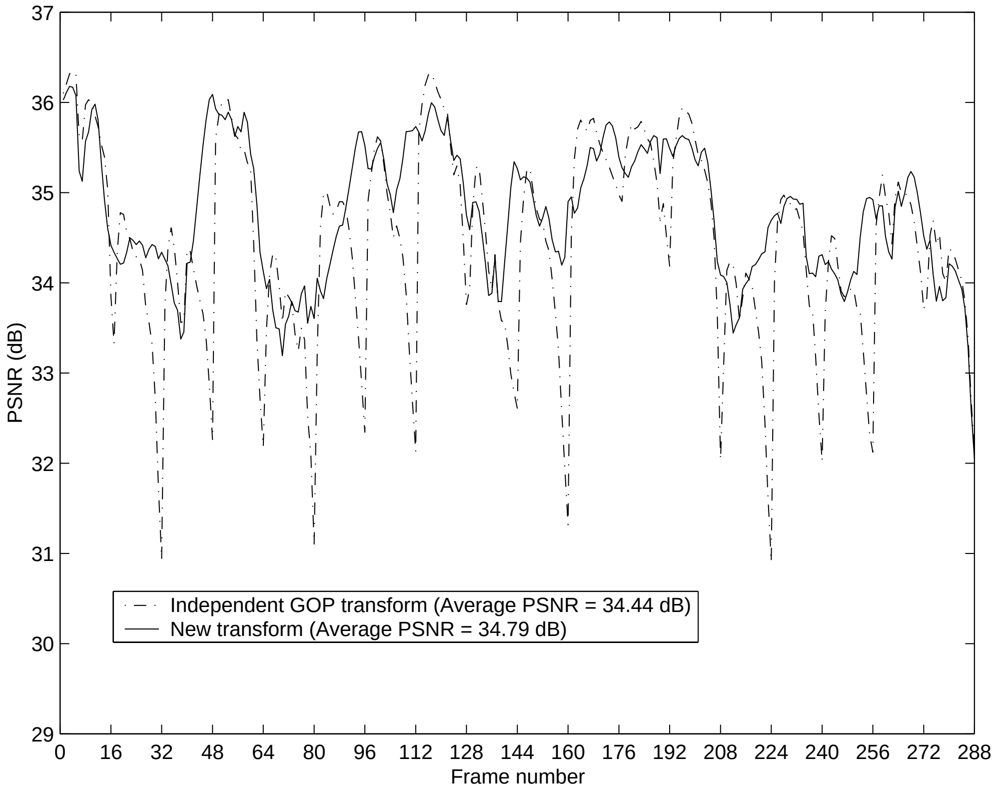


Figure 4: Two-level wavelet decomposition with a shared buffer of 13 frames ($B0$ to $B12$.)



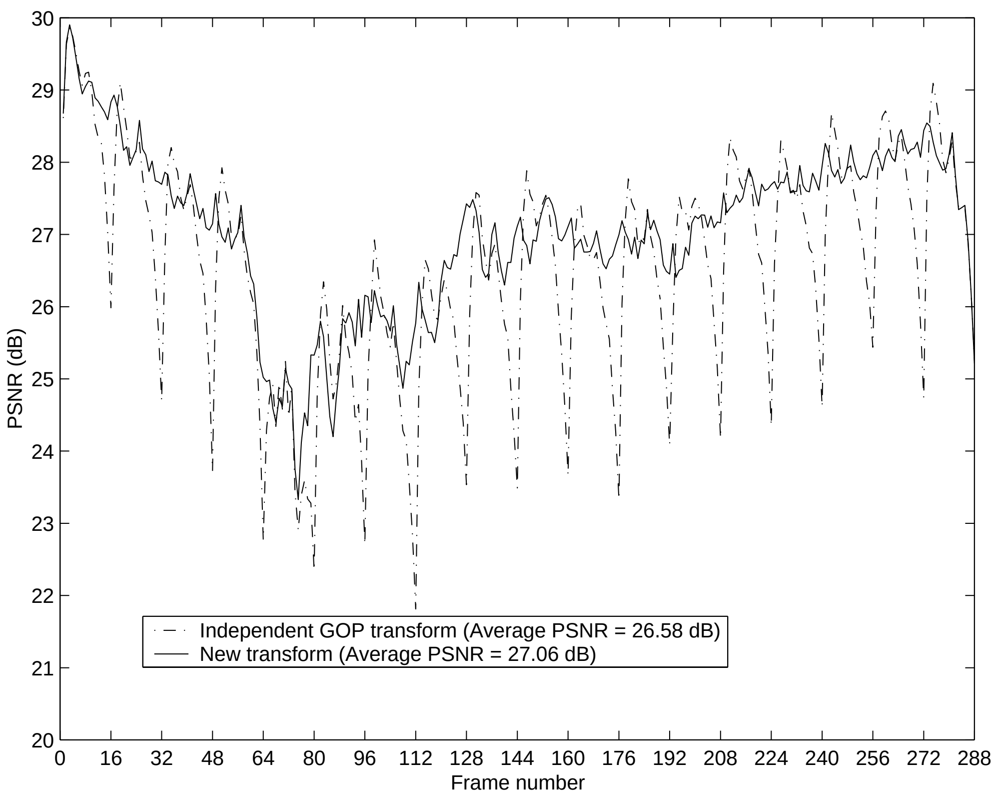


Figure 5: Frame number vs. PSNR plots of 3-D ESCOT coding using two different transform schemes. GOP size = 16. Top: "Akiyo", bit rate = 40.846 kbps; Bottom: "Coast_guard", bit rate = 44.213 kbps. For comparison, MPEG-4 gives an average of 34.18 dB and 26.04 dB, respectively, for these two sequences under the same experimental conditions.