SPIE's 1996 Symp. on Visual Communications and Image Processing '96

Three-Dimensional Subband Coding of Video Using the Zero-Tree Method

Yingwei Chen and William A. Pearlman

Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12180
yingwei@ipl.rpi.edu, pearlman@ecse.rpi.edu

ABSTRACT

In this paper, a simple yet highly effective video compression technique is presented. The zero-tree method by Said [3,4], which is an improved version of Shapiro's [1,2] original one, is applied and expanded to three-dimension to encode image sequences. A three-dimensional subband transformation on the image sequences is first performed, and the transformed information is then encoded using the zero-tree coding scheme. The algorithm achieves results comparable to MPEG-2, without the complexity of motion compensation. The reconstructed image sequences have no blocking effects at very low rates, and the transmission is progressive.

Key word list: video compression, subband coding, video coding, progressive transmission, image sequence coding

1 INTRODUCTION

A video compression scheme needs to exploit both spatial and temporal correlations to be effective. Among many of the techniques, motion compensation is widely used to remove interframe redundancies. Its effectiveness varies depending on the motion properties of the sequence, but in any case it is time consuming. Three-dimensional (3-D) subband decomposition becomes a natural alternative after the zero-tree method was proposed by Shapiro [1], because a 3-D pyramid constructed on a sequence of images fits nicely into the tree structure used in the zero-tree method.

Subband coding of images was introduced by Woods et al. in 1986 [5], and later in 1987 a variation, namely the image pyramid, was first used by Adelson et al. [6]. The underlying idea of coding from subband is to split up the frequency domain of the signal, and by subsampling to flatten each subband so that parameters more accurately matched to the statistics of that subband can be used. In fact, the coding gain from subbands is largely proportional to the ratio between the arithmetic and the geometric means of the energy in subbands. Greater variation of energy distribution results in higher coding gain.

Subband coding of image sequences has been intensively investigated. One of those research efforts is a 2-D subband coding scheme with motion compensation using the zero-tree method by Bhutani et al. [7], which first performs subband decomposition on each frame, then estimates motion vectors within each subband, and codes the Displaced Frame Difference (DFD) using the zero-tree method. Another approach is by Taubman et al. [9], which is a progressive transmission technique using 3-D subband transformation with camera pan compensation. A third scheme is by Podilchuk et al. [8], which uses 3-D subband decomposition together with a new adaptive differential pulse code modulation scheme (ADPCM) and adaptive bit allocation.

Since subband transformed coefficients are uncorrelated across subbands, linear redundancies are removed through the subband transformation. In high frequency bands of the transform domain, those coefficients corresponding to smooth areas of the original signals tend to have small magnitudes. Because for still images, large areas of slow spatial changes are often observed, 2-D subband transform is very efficient in putting the high magnitude transform coefficients in lower frequency subbands. For video signals without much interframe motion, 3-D subband decomposition performs equally well . However, for sequences with large amount of motion, such as "football", there exists high energy in high frequency subbands and that reduces the orderliness of the coefficients in terms of their magnitude distribution from low to high frequency subbands. Various subband filtering schemes with motion compensation in the temporal domain were introduced to solve this problem, and achieved good results in terms of transferring large high frequency coefficients to lower subbands. In our work, we tried to overcome the same problem by applying the zero-tree method, which sends the transform coefficients with partial order by magnitude, and hence is able to locate larger coefficients in other subbands and send them prior to smaller coefficients in the same subband. Therefore, despite unfavorable energy distribution across subbands, 3-D zero-tree works well even for highly dynamic sequences.

The rest of the paper is organized as follows. Section 2 will describe the 3-D subband decomposition and the tree structure needed for the zero-tree method. Section 3 will address the revised zero-tree method. The complete 3-D zero-tree coding system, including implementation details, are presented in Section 4. Section 5 contains simulation results. Section 6 summarizes the work and concludes the paper.

2 3-DIMENSIONAL SUBBAND FRAMEWORK AND TREE STRUCTURE

In this work, a separable 3-D subband filtering is used, i.e., the same filter is used for all spatial and temporal axes. In order to obtain a pyramid of some desired level, a group of three subband filtering operations on three axes respectively is repeated several times on the lowest frequency subband. Figure 1 illustrates the steps to obtain a 3-level 3-D pyramid using 1-D separable Quadrature Mirror Filter.

The operations result in a 3-D pyramid with 15 subbands, which is illustrated in Figure 2.

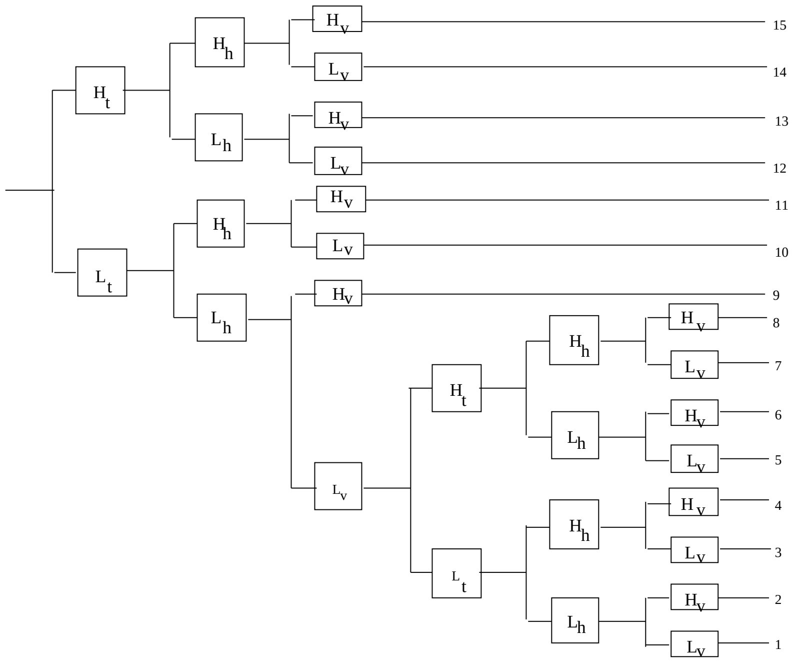

Figure 1: 3-D octave subband splitting using 1-D separable filter

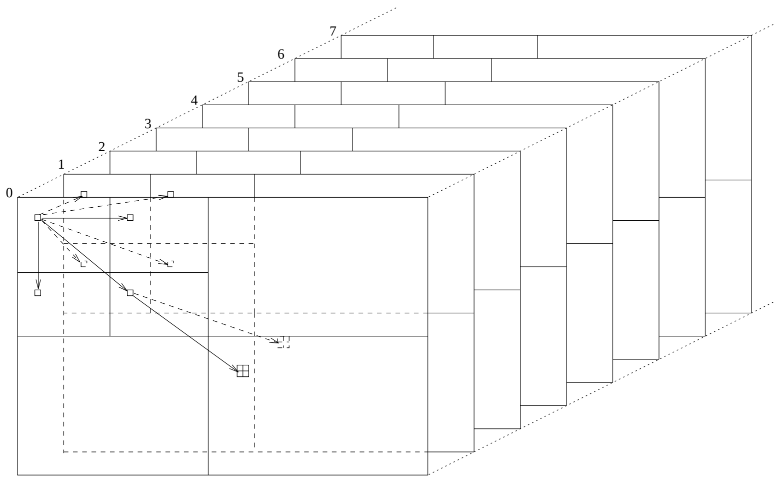

Figure 2: 3-D pyramid tree structure

To come up with the tree structure used in the zero-tree method, let us start with the simpler 2-D case. In two dimensions, a node at the highest level of the pyramid, or root node of the tree, has three children, a node at the middle level (internal node) has four children, while a node at the bottom level (leaf node) has none. We will denote the coordinates of the parent node as (m, n), the dimensions of the image as M and N, and the level of the pyramid as p. Then the dimensions of the highest level of the pyramid is $M_h = M/2^p$ and $N_h = N/2^p$. We further denote the set of root nodes as $\mathcal{R}$, the set of internal nodes as $\mathcal{I}$, and the set of leaf (bottom) nodes as $\mathcal{B}$. The parent-children linkage can then be defined as:

$$\begin{aligned}
(m,n) &\rightarrow \{(m+M_h,n),\ (m,n+N_h),\ (m+M_h,n+N_h)\} \text{ if } (m,n)\in\mathcal{R};\\
(m,n) &\rightarrow \{(2m,2n),\ (2m+1,2n),\ (2m,2n+1),\ (2m+1,2n+1)\} \text{ if } (m,n)\in\mathcal{I};\\
(m,n) &\rightarrow \phi \text{ if } (m,n)\in\mathcal{B}.
\end{aligned}$$

The arrows in the above equations indicate the parent-children relationship.

In three dimensions, we add to the 2-D pyramid a third dimension. If we denote the length of the sequence as L, then the corresponding length at the highest level of the pyramid is $L/2^p$. The coordinates of a node now are (l, m, n). A root node has 7 children, an internal node has 8, and a leaf node has none. Similar to the 2-D case, the parent-children linkage is now defined as:

$$\begin{aligned}
&(l,m,n)\rightarrow\\
&\quad\{(l+L_h,m,n),\ (l,m+M_h,n),\ (l,m,n+N_h),\\
&\quad\ (l+L_h,m+M_h,n),\ (l+L_h,m,n+N_h),\ (l,m+M_h,n+N_h),\\
&\quad\ (l+L_h,m+M_h,n+N_h)\}\\
&\quad\ \text{if } (l,m,n)\in\mathcal{R};\\
&(l,m,n)\rightarrow\\
&\quad\{(2l,2m,2n),\\
&\quad(2l+1,2m,2n),\ (2l,2m+1,2n),\ (2l,2m,2n+1),\\
&\quad(2l+1,2m+1,2n),\ (2l+1,2m,2n+1),\ (2l,2m+1,2n+1)\\
&\quad(2l+1,2m+1,2n+1)\}\\
&\quad\text{if } (l,m,n)\in\mathcal{I};\\
&(l,m,n)\rightarrow\phi\\
&\quad\text{if } (l,m,n)\in\mathcal{B}.
\end{aligned}$$

3 THE 2-DIMENSIONAL ZERO-TREE CODING SCHEME

Zero-tree coding belongs to the family of progressive transmission techniques. By progressive transmission, we mean that the initial segment of a bit stream contains necessary information to reconstruct a coarse version of the original image, and as more bits arrive from the same bit stream, the decoder can refine the image either until a desired quality is achieved or the end of the bit stream is reached. It contrasts with non-progressive transmission in that the latter can not recover a whole image until the whole bit stream is seen. Progressive transmission is desired in many applications, such as video database, where an image or a piece of video will be discarded if one finds out by its contents it is not wanted, or future HD-TV, where receivers with different spatial resolutions or scanning frequencies will be able to watch programs from a single transmitted bit stream.

The simplest progressive transmission approach is to scan through the pixels sequentially, send the highest bit of each pixel, followed by the second bit, and so on. However, little compression can be achieved, because it does not make use of redundancies inside or between images. Usually an orthogonal transform , such as a subband transform, is first performed on the images to remove linear correlations among pixels in different subbands. If we denote the orthogonal transform by T, and assume the original image is defined by a set of pixel values $p_{i,j}$, where (i,j) is the pixel coordinate, then the transform coefficient matrix c can be written as

$$c = T(p).$$

Since the transform is orthogonal, it follows immediately that

$$p = T^{-1}(c).$$

However, in lossy transmission, only an approximation of c is available to the decoder. Hence the decoder can only reconstruct the image as

$$\hat{p} = T^{-1}(\hat{c}).$$

If we use mean squared error as the distortion measure, the distortion between the reconstructed and the original images can be written as

$$D_{mse}(p-\hat{p}) = \frac{||p-\hat{p}||^2}{N},$$

where N is the total number of pixels for the image. Using the fact that Euclidean norm is invariant to orthogonal transforms, we obtain

$$D_{mse}(p-\hat{p}) = D_{mse}(c-\hat{c}) = \frac{1}{N}\sum_i\sum_j (c_{i,j}-\hat{c}_{i,j})^2.$$

From the above equation, it is clear that if initially all $\hat{c}_{i,j}$'s are set to zero, then upon receiving the exact value of $c_{i,j}$, the distortion is reduced by $|c_{i,j}|^2/N$. An alternative interpretation of this result in terms of bits is a '1' in higher order leads to greater distortion reduction than a '1' in lower order. In this sense we say the higher order bits are more important than lower order bits, and should be sent prior to the latter. Ideally, the '1's of the highest bits of all coefficients should be sent first, followed by the '1's of all second bits, and so on, so that whenever the transmission or decoding stops, the image with the lowest distortion can be recovered with the same bit rate. However, the order in which those bits are sent needs to be known to the decoder, and this ordering information will use a substantial number of bits available. Therefore, complete ordering of bits is infeasible. In practice, such as in the zero-tree method, partial ordering is used, and the ordering information can be sent with a moderate number of bits compared to the number of bits used to send the coefficients themselves.

The zero-tree method incorporates the ideas of progressive transmission, partial ordering of coefficients, and self-similarity in the pyramid tree structure. A coefficient is defined to be significant with regard to a threshold if its magnitude exceeds the threshold, and otherwise it is deemed as insignificant. The method is largely based on the observation that if a coefficient is insignificant, then it is probable that all of its descendants are insignificant as well. In the following description, we will denote a coefficient as $c(n)$, and the maximum magnitude of its descendants as $d(n)$, where n is the index of the coefficient. Four states relative to a threshold are defined for each coefficient, which are:

ZRT (Zero-tree RooT): if both $c(n)$ and $d(n)$ are insignificant;
ISZ (ISolated Zero): if $c(n)$ is insignificant but $d(n)$ is significant;
SGR (SiGnificant Root): if $c(n)$ is significant but $d(n)$ is not;
SGT (SiGnificant Tree): if both $c(n)$ and $d(n)$ are significant.

The algorithm maintains two lists: the dominant list (examination list) and the subordinate list (transmission list). Initially the dominant list contains all lowest frequency subband coefficients, whose states are initialized as ZRT, and the subordinate list is empty. The initial threshold is set to the largest number to an integer power of 2 which is smaller than the maximum magnitude of all coefficients. During the first pass of the program, the encoder scans through the dominant list, looking for coefficients whose states change from ZRT to any other states compared to the current threshold. If the new state is ISZ or SGT, meaning there are significant coefficients down in the branches, the immediate descendants are added to the dominant list, to be examined later. If the new state is SGR or SGT, indicating significance of the coefficient, its sign bit is sent, and the coefficient is removed from the dominant list and appended to the subordinate list. Once the encoder reaches the end of the dominant list, it switches to the subordinate list, transmitting the bits compared to half of the current threshold (threshold for the next pass), because from the state information it is already known that the bits compared to the current threshold of coefficients added to the subordinate list during this pass is '1'. Upon finishing this, the encoder halves the current threshold, and repeats the above steps until some stopping condition is met.

The algorithm used by the decoder is almost the same, with only a few differences. First, every output operation is replaced by an input operation. Second, during a dominant pass, the new states are directly obtained through state transition information from the bit stream and the old states. Third, during a subordinate pass, the coefficients are refined using the bits from the bit stream. The complete algorithm can be found in Said's report [4].

Because coding or decoding can stop any time either inside the dominant list or subordinate list, the order in which the coefficients are added is important. By following the above steps, at the same level in the pyramid, significant coefficients are processed prior to insignificant coefficients, unless the latter have significant descendants. In addition, significant coefficients are guaranteed to be located during the current pass, although they will not join the dominant list until their

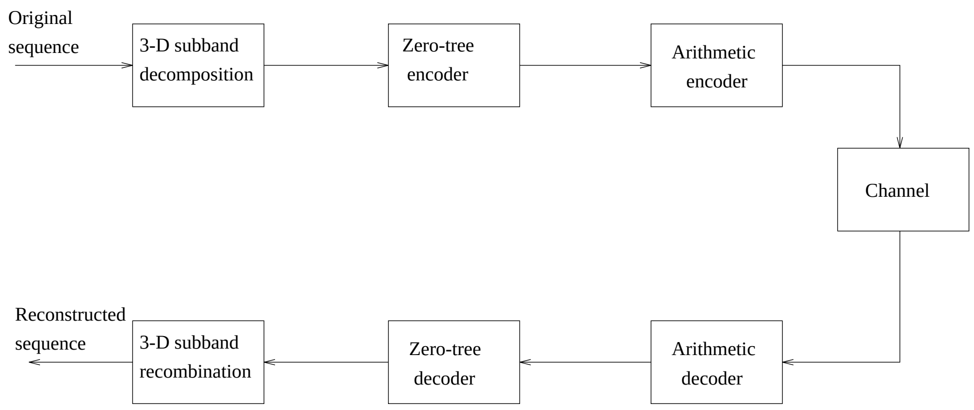

Figure 3: 3-D zero-tree video coding system

parents do. Hence zero-tree method is capable of locating large coefficients quickly, or in other words, with few overhead in transmitting the ordering information.

Note that each time an ISZ or a SGT is newly detected, only the direct descendants are added to the dominant list. Because ISZ or SGT only indicates the existence of significant coefficients but not the locations of such coefficients, it is highly possible that insignificant coefficients are added as well. Moreover, the significant nodes may only exist further down in the subtree, therefore they can not be added to the dominant list until their nonsignificant parents are. Those two cases will lead to more bits being spent on transmission of states rather than on bits of coefficients. This does not happen frequently if the transform coefficients are in good descendant order from the root nodes down to the leaf nodes, which is often true for images without many sharp edges. However, this unfavorable situation will occur in other cases, which is to some extent inevitable, because for those types of images, more bits are needed to achieve the same quality.

4 THE COMPLETE 3-DIMENSIONAL ZERO-TREE CODING SYSTEM AND IMPLEMENTATION DETAILS

After the previous preparation, now we can present the 3-D zero-tree scheme. Basically, a segment of video to be coded is first subband transformed, resulting in a 3-D pyramid, and then the zero-tree algorithm is applied to the pyramid. The resulting bit stream is then arithmetically encoded to further compress the bit stream and the resulting bits are sent to the decoder. The decoder does exactly the opposite: first does arithmetic decoding, then zero-tree decoding, and finally subband recombination. The overall system is illustrated in Figure 3.

The QMF filter used is a 9-tap filter. The same filtering operation is performed on each axis of the temporal and spatial domains. The level of the pyramid is chosen to be 4, because it was observed that no further gain can be achieved by using pyramids of higher levels. Moreover, the level of the pyramid determines directly the minimum number of frames supplied to the program. Since the frames fed to the program are processed as a whole unit, memory requirement is a major issue, and it also poses a delay on the outcoming bit stream which is at least that many

Sequence	Rate (bpp)	3D-ZT (dB)	MPEG-2 (dB)
tennis	0.3	30.7	30.3
tennis	1.0	36.7	36.4
football	0.3	27.3	26.9
football	1.0	33.5	33.0

Table 1: Coding Results (PSNR in dB)

frames long. Besides, large memory occupation restricts the speed of the algorithm. This is not so critical if the memory of the processor far exceeds the amount of data contained in the pyramid, in which case the processor needs not to swap data between main memory and secondary memory frequently. However, this is usually not the case for work stations, therefore the program is considerably slower than it should be if sufficient memory is available. The larger the number of frames to be processed, the slower the program will run. For the above reasons we chose the number of images supplied to the codec to be 16, which is the least for constructing a pyramid of level 4. In most cases where the entire sequence is far longer than 16, the sequence is broken into segments of 16 frames, and coded segment-by-segment. Since the subband transform is separable, there are totally $8 + 3 \times 7 = 29$ subbands in the pyramid, because there are 8 subbands in the highest level of the pyramid, and 7 additional subbands in each lower level of the pyramid.

5 SIMULATION RESULTS

We will present simulation results on two SIF sequences 'table tennis' and 'football', which are both of size 352×240, and sampled at rate 30 frames/sec. Both sequences are grey scale with 8 bits per pixel originally. We tested our program on two rates 0.3bpp (bit per pixel) and 1.0bpp, with total bit rates per second at 760 kbps and 2.53 Mbps respectively. These rates are obtained by exact specification of the total number of code bits over 16 frames. We also compared our results with those generated by using MPEG-2 with motion compensation.

For 'table tennis', the first 80 frames were coded. For 'football', the first 48 frames were coded. The PSNR of each reconstructed frame for both sequences at both rates is plotted in Figure 4 through Figure 7, together with the PSNR's obtained using MPEG-2. In all figures, solid lines stand for results of 3-D zero-tree (3D-ZT), while broken lines stand for results of MPEG-2. The average PSNR's are tabulated in Table 1, again with the results of MPEG-2 to serve as a standard reference.

It can be seen from the table that 3-D zero-tree scheme gives results comparable to those of MPEG-2. As for the visual quality, the reconstructed images at 0.3bpp obtained by the 3-D zero-tree scheme have blurring in some regions, while the images obtained by MPEG-2 have blocking effects, which are more irritating to the human visual system. Another point worth mentioning is that while MPEG-2's coding rate varies with every frame, 3-D zero-tree scheme has a fixed rate allocation over 16 frames.

6 SUMMARY

In this paper, a 3-D subband coding scheme for video using the zero-tree method is addressed. The algorithm is simple: it only needs a subband transform, a tree processing algorithm, and an arithmetic coder. Yet it achieves excellent numerical and visual results. It reproduces motion in a sequence effectively. Finally, the bit stream is embedded.

REFERENCES

1. J. Shapiro, "An embedded wavelet hierarchical image coder ", *Proc. IEEE Intl. Conf. on ASSP*, pp. 657-660, vol.4, March 1992.

2. J. Shapiro, "Embedded Image Coding Using Zero-trees of Wavelet Coefficients", *IEEE Trans. Signal Processing*, pp. 3445-3462, vol.41, 1993.

3. A. Said and W. Pearlman, "Image Compression Using the Spatial-Orientation Tree", *IEEE. Intl. Symp. Circuits and Systems (Chicago)*, pp. 279-282, May 1993.

4. A. Said, "An Improved Zero-tree Method for Image Compression", *IPL Technical Report*, TR-122, Image Processing Laboratory, Rensselaer Polytechnic Institute, Nov. 1992.

5. J. Woods and S. O'Neil, "Subband Coding of Images", *IEEE Trans. ASSP*, pp. 1278-1288, Oct. 1986.

6. E. H. Adelson, E. Simoncelli, and R. Hingorani, "Orthogonal Pyramid Transforms for Image Coding", *Proc. SPIE*, vol. 85, pp. 50-58, Cambridge, MA, Oct. 1987.

7. G. Bhutani, "Image Sequence Coding Using the Zero-tree Method," Master's Thesis, Rensselaer Polytechnic Institute, July 1993.

8. C. I. Podilchuk, N. S. Jayant, and N. Farvardin, "Three-Dimensional Subband Coding of Video," *IEEE Trans. Image Processing*, pp. 125-139, VOL. 4, No. 2, Feb 1995.

9. D. Taubman and A. Zakhor, "Multi-Rate 3-dimensional Subband Coding of Video," *IEEE Trans. Image Processing*, pp. 572 - 588, VOL. 3, Sep. 1994.

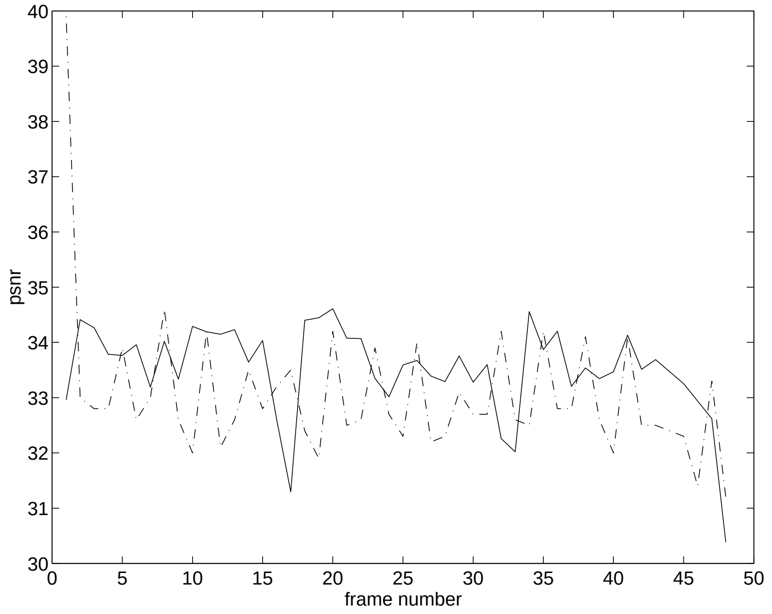

Figure 4: PSNR vs. frame number for "football" at 1.0 bpp.
Solid line: 3D-ZT; broken line: MPEG-2.

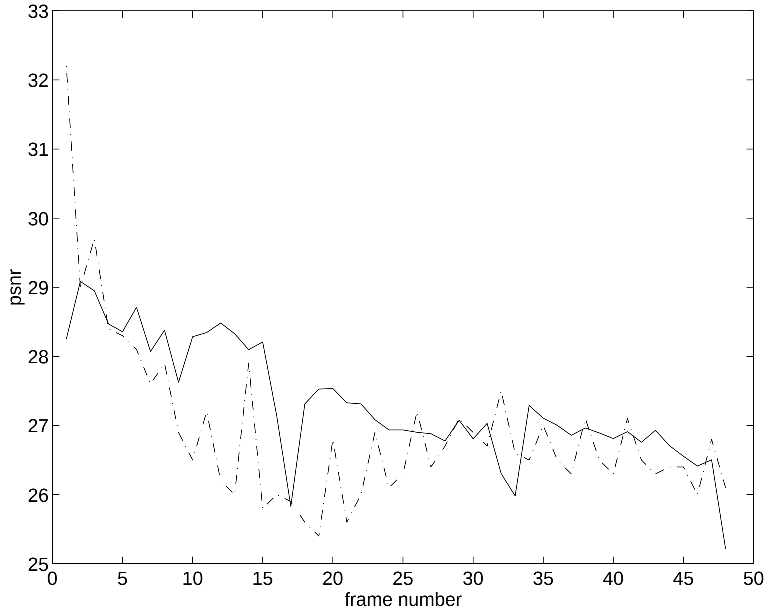

Figure 5: PSNR vs. frame number for "football" at 0.3 bpp.
Solid line: 3D-ZT; broken line: MPEG-2.

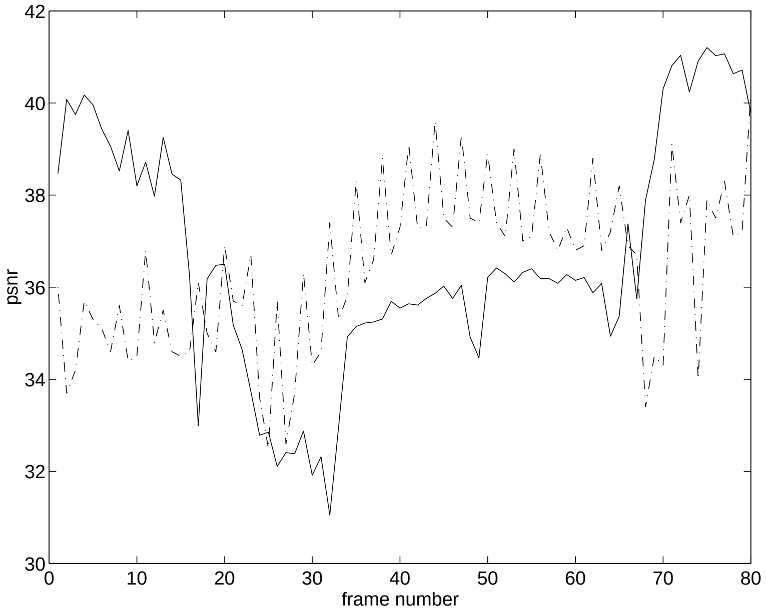

Figure 6: PSNR vs. frame number for "table tennis" at 1.0 bpp.
Solid line: 3D-ZT; broken line: MPEG-2.

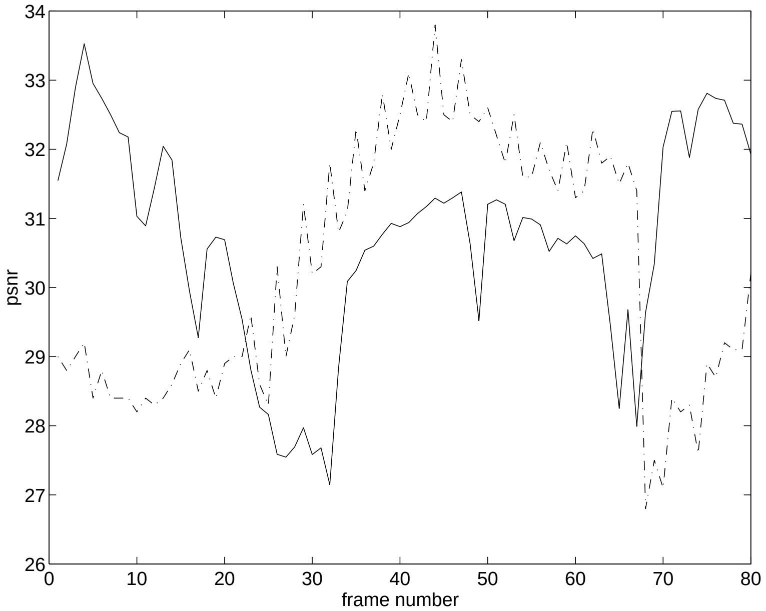

Figure 7: PSNR vs. frame number for "table tennis" at 0.3 bpp.
Solid line: 3D-ZT; broken line: MPEG-2.