Editor: Harrick Vin
University of Texas at Austin

Multimedia Broadcasting over the Internet: Part II—Video Compression

Borko Furht
Florida Atlantic University

Raymond Westwater
Future Ware

Jeffrey Ice
Pipe Dream

In the first part of our project report, published in the October-December 1998 issue, we described a new technique for multimedia broadcasting over the Internet, called IP Simulcast. In this article we introduce a new video compression technique, called *XYZ* compression, which suits ultra-low bandwidth applications very well.

Current technology permits delivering video over the Internet. Video applications on the Internet divide into the following groups: videophone and videoconferencing applications, viewing video on the Web, and video broadcasting. Delivering video via the Internet requires a combination of some compromises and clever engineering, summarized as

- *Picture size*: Typical Web-based videos use a quarter (240 × 320 pixels) or a sixteenth (120 × 160 pixels) of a screen.

- *Frame rate*: A normal television picture delivers 30 full pictures (or frames) per second (25 for the European video standard PAL systems). Most computer video reduces this to 15 or 10 or even fewer frames per second.

- *Compression*: A television picture includes a lot of redundant information—neighboring areas tend to have the same color and lightness, and neighboring frames often don't change much. Video compression techniques compress the data with varying amounts of image degradation and artifacts.

- *Quality compromises*: A few other parameters must be sacrificed due to bandwidth limitations such as video noise and color fidelity.

The *XYZ* video compression technique

The *XYZ* compression extends discrete cosine transform (DCT) encoding to moving pictures. Sequences of eight frames are collected into a 3D block, referred to as a video cube, to which a 3D DCT is applied. The 3D DCT has been used in the past to encode errors after motion estimation. Previously developed, 3D DCT-based compression algorithms have fallen into disfavor because they required excessive computation and large memory buffers. Therefore, they were considered less effective than motion estimation.

The resurrected 3D DCT-based video compression technique takes advantage of the statistical behavior of video data in both the spatial and temporal domains. The algorithm also includes a new method for adaptive quantization that generates optimal quantizers based on statistics gathered for the eight consecutive frames in a video sequence. The human eye's sensitivity to various DCT coefficients helps modify the quantizers and create a "visually equivalent" cube with still greater energy concentration.

The *XYZ* video compression algorithm takes a full-motion digital video stream and divides it into groups of eight frames. Each group of eight frames forms a 3D image, where *X* and *Y* are spatial components, and *Z* is the temporal component. Each frame in the image is divided into 8 × 8 blocks (like the Joint Photographic Experts Group JPEG format), forming 8 × 8 × 8 cubes, as illustrated in Figure 1 (next page). The three blocks of the *XYZ* video encoder—3D DCT, Quantizer, and Entropy encoder independently encode each 8 × 8 × 8 cube. Figure 2 (next page) shows a block diagram of the *XYZ* compressor.

The original unsigned pixel sample values—typ-

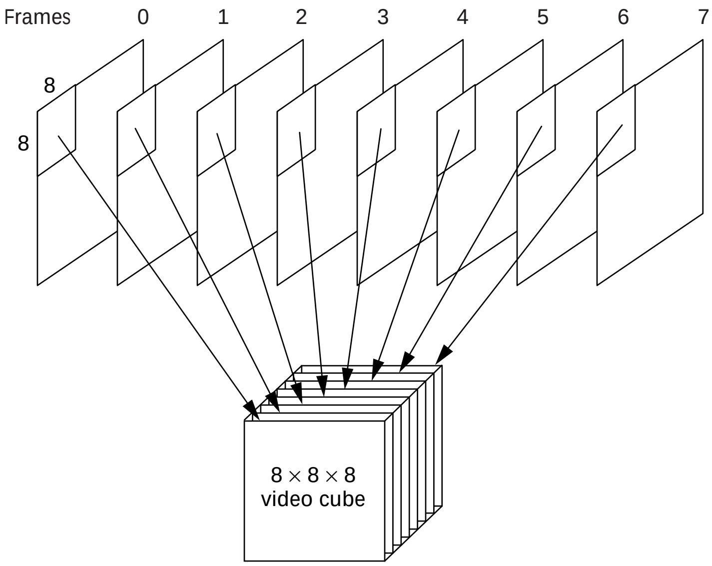

Figure 1. Forming an 8 × 8 × 8 video cube for XYZ compression.

ically in the range [0, 255]—are first shifted to signed integers, say in the range [–128, 127]. Then the Forward 3D DCT transforms each 8 × 8 × 8 cube of 512 pixels into the frequency domain using

$$F(u,v,w) = C(u)C(v)C(w)* \sum_{x=0}^{7}\sum_{y=0}^{7}\sum_{z=0}^{7} f(x,y,z)* \frac{\cos\left(\left(2x+1\right)u\pi\right)}{16} \frac{\cos\left(\left(2y+1\right)v\pi\right)}{16} \frac{\cos\left(\left(2z+1\right)w\pi\right)}{16}$$

where x, y, z are index pixels in pixel space; $f(x, y, z)$ is the value of a pixel in pixel space; u, v, w are index pixels in DCT space, $F(u, v, w)$ is a transformed pixel value in DCT space; and

$$C(i) = \frac{1}{\sqrt{2}} \quad \text{for } i = 0$$
$$C(i) = 1 \quad \text{for } i > 0$$

A function in three dimensions, the transformed 512-point discrete signal contains both spatial and temporal information. Most of the energy is contained in a few low-frequency coefficients, while the majority of the high-frequency coefficients have zero or near-zero values.

In the next step, all 512 DCT coefficients are quantized using a 512-element quantization table. Quantization introduces minimum error while increasing the number of zero-value coefficients. Quantization may also be used to discard visual information that the human eye may not recognize. Quantizer tables can be predefined, or adaptive quantizers can be developed and transmitted with the compressed data.

Quantization is performed according to the following equation:

$$F_q(u,v,w) = \left\lfloor \frac{F(u,v,w)}{Q(u,v,w)} \right\rfloor$$

where $F(u, v, w)$ are the elements before the quantization; $F_q(u, v, w)$ are the quantized elements; and $Q(u, v, w)$ are the elements from the quantization table.

Each quantizer $Q(u, v, w)$ falls in the range [1, 1024]. The quantization operation results in a collection of smaller valued coefficients, a large number of which equal 0. An entropy coder (in this case, a Huffman coder) converts these coefficients into a compact binary sequence.

The entropy coding operation starts with reordering the coefficients in descending order of expected value. Similarly to JPEG and the Moving Pictures Expert Group (MPEG) format, the coefficients are reordered into a "zig-zag" sequence by placing low-frequency coefficients—more likely to be nonzero—before high-frequency coefficients. This zig-zag sequence is composed as follows:

$\{F(0, 0, 0), F(1, 0, 0), F(0, 1, 0), F(0, 0, 1), F(1, 1, 0), F(1, 0, 1), F(0, 1, 1), F(2, 0, 0), F(0, 2, 0), F(0, 0, 2), \ldots F(7, 7, 7)\}$

This sequence has the benefit of collecting sequentially the largest number of zero-valued coefficients. The run lengths of zero coefficients are computed, and the alphabet of symbols to be encoded becomes the run length of zeros append-

Figure 2. Block diagram of the XYZ compressor.

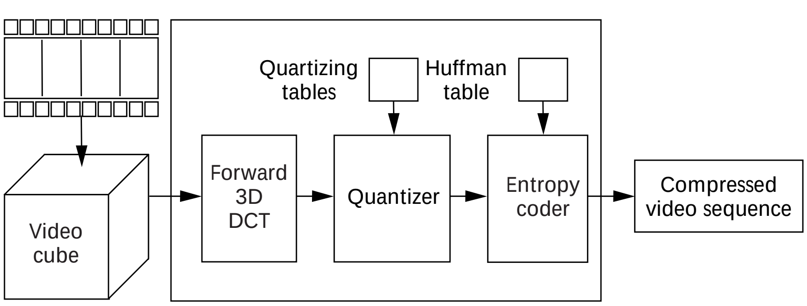

Table 1. Comparison of XYZ and MPEG video compression algorithms.

Compression Algorithm	Compression Ratio	Normalized Root Mean Square (RMS) Error	Encoder/Decoder Complexity	Total Complexity
XYZ	101.7	0.120	240/240	480
MPEG	32.9	0.125	750/100	850

ed to the length of the nonzero coefficient. This binary sequence represents the compressed $8 \times 8 \times 8$ block.

In implementing the *XYZ* video compression encoder and decoder, we developed a fast 3D DCT algorithm. We also created the quantizers based on human visual acuity.

The smaller overhead of the *XYZ* technique, as compared to the MPEG video technique and other standard compression algorithms, suggests that the *XYZ* algorithm suits real-time video applications. The demonstrated results of compression and encoding standard motion video benchmarks also suggests that the *XYZ* video compression technique is not only faster than the MPEG standard and other video compression techniques, but also provides superior compression ratios and higher quality video.

These promising results indicate that *XYZ* video compression can be applied to a variety of video applications including videophone and videoconferencing, consumer broadcast television, video broadcasting over the Internet, and video-on-demand applications. These applications require a real-time video compression and encoding system while maintaining high-quality video. The *XYZ* technique meets all these requirements. Table 1 compares *XYZ* compression to the MPEG compression standard for a video sequence from the movie "Dick Tracy."

Figure 3 presents eight frames of the original video, the *XYZ*-decompressed video, and an RMS error.

The most significant commercial idea embodied in the *XYZ* technique consists of a very powerful video encoding system, which has the following features:

- Produces a high-compression ratio without visible artifacts

Figure 3. Demonstration of the XYZ video compression technique using eight frames from the movie "Dick Tracy." (a) Original sequence, (b) decompressed sequence (compression ratio = 101.7, RMS = 0.120), and (c) error between frames multiplied by 16.

Table 2. Overview of video compression techniques.

Compression Algorithm	Typical Compression Ratio	Characteristics
Indeo	23:1	A 128 × 240 data stream interpolated to a 256 × 240 data stream. Color subsampled 4:1. Uses a simple 16-bit codebook without error correction and frame differencing.
Motion JPEG	10:1	Uses 2D DCT to encode individual frames. Extension of the JPEG algorithms for images.
Fractals	10:1	Fractals compress natural science well, but require tremendous computing power.
Wavelets	20:1	Uses 2D and 3D wavelets to compress motion video.
H.323	50:1	Real-time compression and decompression algorithm for video telecommunications. Based on 2D DCT with simple motion estimation between frames.
MPEG	30:1	Uses 2D DCT with motion estimation and interpolation between frames. Complex and expensive to compress, but plays back in real time.
XYZ	50:1 to 100:1	Uses 3D DCT and adaptive quantization. Complex motion estimation replaced by the DCT transform in the time domain. A symmetrical algorithm applied both as a real-time encoder and decoder.

- Provides real-time video compression with inexpensive hardware
- Provides real-time decompression for playback using inexpensive hardware or with a software-only solution
- Degrades easily under network overload

Comparison to other video compression techniques

Many digital video compression algorithms have been developed and implemented. The compression ratios of these algorithms vary depending on the subjective acceptable level of error (or quality of the video) and who makes the claim. Table 2 summarizes video compression algorithms, their typical compression ratios as reported in the literature, and their characteristics.

The trade-off among compression ratio, data rate, and video quality is arguably the most important decision in the entire field of distributed video applications. Higher compression factors and data rates increase costs to users and video producers. For example, a higher compression factor usually requires special-purpose hardware for decompression during playback and for compression during video preparation.

This trade-off applies to transmitting video over telecommunication lines and the Internet even more strongly than it applies to playing video from disks. An expensive T1 line corresponds roughly to a single-speed CD-ROM drive. However, an Integrated Services Digital Network (ISDN) line corresponds to only a 1/10-speed drive, so video over this moderately sophisticated line must have both lower quality and higher compression than on T1 lines or CD-ROMs. To send and receive video over a standard telephone line with a 28.8-Kbps modem—which corresponds to a 1/44-speed drive—the user must be prepared not only to settle for low video quality and lower frame rates, but also to pay for expensive compression and decompression hardware.

Current work

We're presently experimenting with several promising techniques, which will make *XYZ* compression even more effective. An innovative interpolation technique, applied at the decoder side, can enlarge the size of video at the receiver side without introducing visible artifacts. For example, if the original video is 120 × 160 pixels, the reconstructed video could be 240 × 320 pixels or higher. Existing interpolation techniques operate on pixels after decompression, while our technique extends the number of discrete cosine coefficients before performing decompression. It turns out that the proposed technique, based on discrete cosine coefficients, performs much better than pixel interpolation techniques.

In addition, we're also experimenting with video cubes of various sizes (see Figure 1). Preliminary experiments show that with $16 \times 16 \times 8$ and $32 \times 32 \times 8$ cubes we can achieve compression ratios of about 200:1, while the quality of the reconstructed video remains acceptable.

Finally, we're developing the SimulSees system, which uses the IP Simulcast protocol combined with *XYZ* video compression to provide television broadcasters with an efficient and inexpensive solution for broadcasting (Webcasting) television programs to a large number of clients. The initial cost for broadcasters includes a simple and inexpensive server and a small network bandwidth.

We believe these projects will help make the XYZ compression technique a more robust and effective solution for transmitting high-quality video over the Internet. MM

Readers may contact Furht at the Department of Computer Science and Engineering, Florida Atlantic University, 777 Glades Rd., Boca Raton, FL 33431, e-mail, borko@cse.fau.edu.

Contact Project Reports editor Harrick Vin at the University of Texas, Austin, Department of Computer Sciences, e-mail vin@cs.utexas.edu.

Further Reading

The original article on *XYZ* video compression.

R. Westwater and B. Furht, "The *XYZ* Algorithm for Real-Time Compression of Full-Motion Video" *Real-Time Imaging Journal*, Special Issue on Image and Video Processing in Multimedia Systems, Academic Press, Cambridge, UK, Vol. 2, No. 1, February 1996, pp. 19-34.

Presents a number of experiments with the *XYZ* algorithm including its VLSI implementation.

R. Westwater and B. Furht, *Real-Time Video Compression: Techniques and Algorithms*, Kluwer Academic Publishers, Norwell, Mass., 1996.

Presents optimal quantizers for the *XYZ* video compression.

R. Westwater and B. Furht, "Three-Dimensional DCT Video Compression Technique Based on Adaptive Quantizers" *Proc. Second IEEE Int'l Conf. Engineering of Complex Computer Systems*, IEEE Press, Piscataway, N.J., October 1996.