Wavelet Zerotree Image Compression with Packetization

Jon K. Rogers and Pamela C. Cosman, *Member, IEEE*

***Abstract*— We describe a combined wavelet zerotree coding and packetization method that provides excellent image compression and graceful degradation against packet erasure. For example, using 53-byte packets (48-byte payload), the algorithm compresses the 512 × 512 gray-scale Lena image to 0.2 b/pixel with a peak signal-to-noise ratio (PSNR) of 32.2 dB with no packet erasure, and 26.3 dB on average for 10% packets erased.**

I. Introduction

VARIOUS problems occur in packet switched networks. Inadequate buffer space at network switches may cause packets to be dropped during periods of congestion (packet erasure). Packets may be received with corrupted bits, and the decoder might make use of the erroneous data in the packet. Error detection coding enables the decoder to discard a corrupted packet, and retransmission protocols (ARQ) allow the decoder to request that missing or discarded packets be sent again. ARQ schemes introduce delay. Forward error correction (FEC) techniques allow the decoder to correct a certain number of errors, but they reduce the compression achievable because extra bits are added. Error concealment techniques seek to approximate (for example, by interpolation) the data from an erased packet. The wavelet zerotree compression and packetization method described in this letter is resilient to packet erasures without the use of FEC or ARQ schemes. Previous related work is [1]–[3].

II. Packetizable Zerotree Wavelet (PZW) Compression

The encoder begins by using a variation on the embedded zerotree wavelet (EZW) and set partitioning in hierarchical trees (SPIHT) coders [4], [5] to encode and store the entire image to the target bit rate. The variation is essentially that of [5], with the differences that arithmetic encoding is not used, only four levels of wavelet decomposition are done, and each "head" coefficient in the low-low band has three children, one in each of the next directional bands, as in [4]. For a 512 × 512 image, there are 1024 head coefficients in the low-low band; each has $3 \times (1 + 4 + 16 + 64) = 255$ descendants in its tree. Each stored bit is associated with exactly one of the 1024 trees. The SPIHT and EZW coders put out bitstreams in which the bits corresponding to different trees are interleaved; this yields progressivity. The most significant bit for every coefficient of every tree is made known to the decoder before transmitting any information about the next significant bit. To achieve robustness, we sacrifice progressivity. The stored bits are deinterleaved, and reorganized into 1024 substreams, where each substream contains information pertaining to only one tree of coefficients.

A cumbersome but straightforward packetization method can now be seen, which serves as an introduction to our method. The 1024 substreams are ordered in some fixed order (e.g., a raster scan) known to both encoder and decoder. We assume initially that no substream has more than 48 bytes. The encoder concatenates substreams into a 48-byte packet until no more will fit. If only n trees fit, the encoder pads out any space remaining in the packet with null bits to the end, and tree $n+1$ starts the next packet. If one were to packetize this way, a substantial amount of overhead would be required in each packet. First, each packet needs to say which tree begins the packet. If the first packet contains trees 1–4, and the second one contains trees 5–11, in the event that the first packet is lost, the decoder would not know that tree 5 begins the second packet. So, for the packets to be independently decodable, each one must include information about which tree starts the packet. Since there are 1024 head coefficients, 10 b are required for this. Second, the decoder must be able to parse out the concatenated substreams. At any point in the original EZW and SPIHT algorithms, by interpreting the bits received up to that point, the decoder can determine to which tree the next bit pertains. The decoder simply marches through the decreasing threshold levels and the trees (sets) until either a stop code is encountered or a predetermined target rate is reached. However, when the substreams are deinterleaved and concatenated, a separate stop code would be needed to indicate the terminating point of each substream; alternatively the encoder can inform the decoder how many bits are in each substream, and this would equally well enable the decoder to parse them out.

It turns out that the null-padding and the need for stop codes or explicit bit counts for the trees can all be avoided. By reinterleaving the set of substreams within any one packet, the decoder can decode each of the trees without stop codes or additional information regarding each tree size. Instead of including null bits at the end of the packet, the encoder fills the packet up with additional useful bits—the n trees are encoded at a higher rate. More sorting and refinement passes are conducted for those trees alone, and the results interleaved, until the fixed-length packet is exactly filled. The

Manuscript received August 29, 1997. This work was supported by ARO under Grant DAAH04-95-1-0248 and by an NSF Career Award under Grant MIP-9624729.

The authors are with the Department of Electrical and Computer Engineering, University of California-San Diego, La Jolla, CA 92093-0407 USA (e-mail: pcosman@celece.ucsd.edu).

Publisher Item Identifier S 1070-9908(98)03726-2.

Fig. 1. Left: original Lena image. Middle: compressed and packetized Lena image at 0.209 b/pixel with no erasure, PSNR = 32.19 dB. Right: 0.209 b/pixel version with 20% of packets erased, PSNR = 25.41 dB.

image quality will thus vary somewhat spatially. Instead of coding all coefficients down to the same bit plane (threshold), each packet has its own terminating threshold which is not told to the decoder. The decoder reads a packet by repeatedly cutting the threshold in half and marching through the bit planes until it reaches the end of the fixed-length packet, at which point all the interleaved substreams in that packet are considered terminated, without the need for an explicit stop code. With packets not arriving, some trees will be missing and require interpolation, but reconstruction of the arrived packets is unimpaired. A small piece of additional overhead (say, 4 b) is required in order to tell the decoder how many trees are interleaved in the current packet. The decoder would be unable to correctly cycle through the round-robin of interleaved trees in the packet if it did not know how many there are. The reason this is not a problem in the original, unpacketized zerotree algorithms is that the number of trees is always fixed and known to both encoder and decoder (given the image size).

A. Refinements

In practice, refinements to this basic idea are needed for the algorithm to work well.

- A raster scan order for the 1024 trees allows neighboring coefficient trees to get put in a packet together, and therefore to get lost together; interpolation of the lost coefficients using the neighbors will thus be less accurate. We order the trees with a recursive tessellation technique [6] used in dispersed-dot dithering, ensuring that trees in each packet come from widely dispersed locations in the image.
- If 4 b specify how many trees fit in a packet, the system cannot handle packets with more than 16 trees or less than one. The binary word `1111` is reserved to signal that there are more than 15 trees in the packet, or that there is less than one, or various other special conditions. Whenever the escape word is used, it is followed by a fixed-length word which specifies what kind of special condition occurred. In practice, for the USC data base images tested, the number of trees per packet remained strictly between one and 15 for the bit rates of interest (0.1–0.4 b/pixel), but the algorithm can handle trees of any size.
- If n trees at the target bit rate fit within the 48-byte payload, but $(n+1)$ do not, a choice is made between putting in n trees (growing them out to fill the packet exactly) and putting in $(n+1)$ trees (pruning them back to fill the packet exactly). We choose whichever is closer to 48 bytes, but the decision could be based on a distortion-rate trade-off, or by using lookahead to see how well future groups of trees will fit into future packets.
- Header information such as the number of rows and columns in the image, the starting threshold, etc. can be handled in a number of ways. If the system will always operate on an image of a fixed size, the size does not need to be stated. Otherwise, that information can be provided redundantly in several different packets. Each packet can use its own starting threshold. For example, 2 b within each packet can specify one of four standard starting thresholds. If the threshold for a given packet is not one of the standard four, the `1111` escape word can be used to indicate this special condition.

TABLE I
PSNR RESULTS FOR COMPRESSING LENA AND PEPPERS TO 0.209 B/PIXEL. PZW REFERS TO THE PACKETIZABLE ZEROTREE WAVELET METHOD DESCRIBED IN THIS PAPER

Image	SPIHT +arith.	SPIHT w/o arith	PZW no loss	PZW 1% loss	PZW 10% loss	PZW 20% loss
Lena	33.37	32.94	32.19	31.33	26.29	24.63
Peppers	32.83	32.35	31.75	30.85	26.38	23.31

III. RESULTS AND CONCLUSIONS

The PZW algorithm was used to compress the 512 × 512 8 b/pixel gray-scale images, Lena and peppers. The initial (progressive) wavelet coding was at 0.2 b/pixel. After packetization, the actual rates achieved were 0.209 for Lena and 0.208 for peppers. These higher rates include the 14-b overhead for

each packet (10 b to specify the starting tree, and 4 b to specify how many trees), as well as the effects of growing and pruning trees within each packet. The PSNR's achieved at these rates (for four cases: all packets arriving, 1, 10, and 20% packets erased) are shown in Table I along with the PSNR's for the SPIHT algorithm with and without arithmetic coding. The PSNR's were obtained by averaging the mean squared errors for 10 000 random realizations of the packet erasures. Note that for PZW the burstiness of the packet erasures makes no difference, since all packets are equivalent *a priori*. That is, if 20% of the packets are lost, it matters not which 20% are lost. The decoder interpolates missing coefficients in the low-low band by averaging together as many of their immediate eight-neighbors as are available. Missing coefficients in other bands are replaced by zeros prior to inverse transforming the entire group. Fig. 1 shows a comparison among the original 512×512 gray-scale Lena image, the version that has been compressed and packetized at 0.209 b/pixel with no erasures, and the 0.209 b/pixel version with 20% packet erasures.

The wavelet zerotree compression and packetization algorithm presented here is resilient to packet erasure without a requirement for retransmission requests. This might be useful for channels with long round-trip delays, for real-time interactive systems, or for situations where a receiver does not want to reveal its location. The method provides a controlled degradation in image quality as more packets are dropped. The utility of PZW is not restricted to the case of packet-switched networks; the sequence of 48-byte payloads can be streamed together and would guarantee resynchronization between the decoder and encoder after any bit errors. Unlike resynchronization strategies that rely on the use of fairly long resynchronization flags, PZW provides resynchronization points at fixed known intervals in the compressed bitstream, and so avoids the use of flags. PZW is particularly useful for burst errors; if the packet employs a CRC check, a single bit error or a hundred errors would still cause the check to fail and the packet to be discarded. In [3], trees of wavelet coefficients are also put in separate streams for error robustness; however, in that work a fixed number of trees are grouped into each substream, and the substreams are interleaved. In PZW, a variable number of trees are grouped into fixed-length segments. Thus PZW is more suitable for the case of burst errors and packet erasures. In conjunction with FEC, PZW may prove to be robust against high levels of both random noise and burst errors.

REFERENCES

[1] H. Man, F. Kossentini, and M. J. T. Smith, "Robust EZW image coding for noisy channels," *IEEE Signal Processing Lett.*, vol. 4, pp. 227–229, Aug. 1997.
[2] V. J. Crump and T. R. Fischer, "Intraframe low bitrate video coding robust to packet erasure," in *Proc. DCC '97*, Snowbird, UT, p. 432.
[3] C. D. Creusere, "A new method of robust image compression based on the embedded zerotree wavelet algorithm," *IEEE Trans. Image Processing*, vol. 6, pp. 1436–1442, Oct. 1997.
[4] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
[5] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–249, June 1996.
[6] R. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1987.