Robust wavelet zerotree image compression with fixed-length packetization*

Jon K. Rogers Pamela C. Cosman
Dept. of Electrical and Comp. Engineering, Univ. of California at San Diego
La Jolla, CA 92037-0407 {jkrogers,pcosman}@code.ucsd.edu

Abstract

We present a novel robust image compression algorithm in which the output of a wavelet zerotree-style coder is manipulated into fixed-length segments. The segments are independently decodable, and errors occurring in one segment do not propagate into any other. The method provides both excellent compression performance and graceful degradation under increasing packet losses. We extend the basic scheme to perform region-based compression, in which specified portions of the image are coded to higher quality with little or no side information required by the decoder.

1 Introduction

Wavelet zerotree image coding techniques developed by Shapiro (EZW) [1], and further refinements by Said and Pearlman (SPIHT) [2] provide high performance, low complexity image compression. These algorithms are highly dependent on the *state* of the system and are, therefore, highly susceptible to bit errors. A single bit error could potentially lead to decoder derailment. If the output streams from these algorithms were directly divided into packets, loss of a single packet (due either to packet misrouting, or to corrupted bits which might be detected with some error detection scheme) would lead to uncontrolled degradation of the image quality.

Various strategies exist to address these issues. Retransmission protocols (ARQ) allow the decoder to request that the encoder send a packet again. Forward error correction (FEC) techniques allow a certain number of errors to be corrected. Both of these strategies entail some cost. Retransmission protocols introduce delay; the retransmitted packets might not arrive soon enough to be useful. Error correction

*This work was supported by ARO Grant No. DAAH04-95-1-0248 and by an NSF Career Award under Grant MIP-9624729.

schemes reduce the compression achievable because extra bits are added. Furthermore, there is no graceful degradation in image quality if the error correction capacity of the code is exceeded.

Several researchers have recently used various modifications of wavelet zerotree compression to compose noise-robust coders. Man et al., [3] modified the SPIHT algorithm to contain a larger portion of fixed rate symbols. The bitstream is then broken into sub-streams which receive different amounts of error protection based on their noise sensitivity. Creusere [4], using a variation of EZW, partitioned wavelet coefficient trees into groups which were each independently coded. The resulting sub-streams were interleaved for transmission. Any single bit error will only corrupt one sub-stream. Although not using zerotree-style quantization, the video codec in [5] involving entropy coded scalar quantization of subband coefficient trees together with run-length and Huffman coding, is related in spirit to the current work. The bitstream in [5] is divided into variable-length independent packets which allow complete subband coefficients trees to either be lost or received as a unit.

In this paper, we present a wavelet zerotree compression and packetization method that is robust against packet erasure without the use of FEC or ARQ schemes. It differs from previous schemes in that we do not use FEC, and in that the output stream is composed of fixed-length segments. This is suitable for packet-switched networks operating with fixed-length packets, or, for circuit-switched networks, it can be seen as a method to provide resynchronization points to the decoder at fixed intervals. We discuss its extension into a region-based codec in which a small region of an image is coded to a high rate (high quality) while the remainder is coded to a low rate (low quality).

2 Packetizing the Zerotree Bitstream

Wavelet zerotree coders can operate with different numbers of decomposition levels, and with different structures for the parent-child relationships. We here describe an encoder with particular choices for the decomposition levels and the parent-child structures; these choices were motivated by our goal of operating with a packet length of 53 bytes (48 payload) and target bit rates in the range of 0.1 to 0.4 bpp, but other choices could be used as well. The basic idea is to modify the SPIHT/EZW zerotree-style coder such that the final bitstream can be manipulated into fixed-length segments (packets) which are independently decodable.

The encoder begins with the basic SPIHT [2] algorithm with no arithmetic encoding, and only four levels of wavelet decomposition. The coefficient tree structure used is that of Shapiro [1], in which each "head" coefficient in the low-low band has 3 children, one in each of the next directional bands. 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. The encoder encodes the image out to a target bit rate (e.g., 0.2 bpp) and stores the bits. Each stored bit is associated with exactly one of the 1024 trees.

The SPIHT and EZW coders put out bit streams 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 noise robustness, we sacrifice this progressivity. The output bitstream is de-interleaved and re-organized into 1024 variable-length sub-streams, where each sub-stream contains information pertaining to only one tree of coefficients.

A cumbersome but straightforward fixed-length packetization method can now be seen, which serves as an introduction to our method. The 1024 sub-streams are ordered in some fixed order (e.g., a raster scan) known to both encoder and decoder. We assume initially that we are using standard ATM packets (48-byte payload) and that no sub-stream has more than 48 bytes. The encoder concatenates sub-streams into a packet until no more will fit. If only n trees fit, the encoder pads out any space remaining in the packet with null bits, and tree $n + 1$ starts the next packet. A substantial amount of overhead would be required in each packet. Firstly, 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. Since there are 1024 head coefficients, ten bits are required in each packet to say which tree starts the packet. Secondly, the decoder must be able to parse out the concatenated sub-streams. 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 pre-determined target rate is reached. However, when the sub-streams are de-interleaved and concatenated, a separate stop code would be needed to indicate the terminating point of each sub-stream; alternatively the encoder can inform the decoder how many bits are in each sub-stream, 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 re-interleaving the set of sub-streams in any one packet, the decoder can decode each tree in the packet without stop codes or additional information regarding each tree size. A small piece of additional overhead (say, 4 bits) 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.

Each packet gets filled exactly. To fill the next packet, the encoder examines the upcoming ordered sub-streams. Suppose the next n trees would underfill the packet; the encoder can grow them out by encoding them at a rate higher than the initial target rate. As necessary, more sorting and refinement passes are conducted for those trees alone, and the results interleaved, until the packet is exactly filled. Alternatively, the $n{+}1$ trees overfilling the packet can be pruned back until the packet is filled exactly (see Figure 1). The encoder currently chooses between growing a smaller set of trees

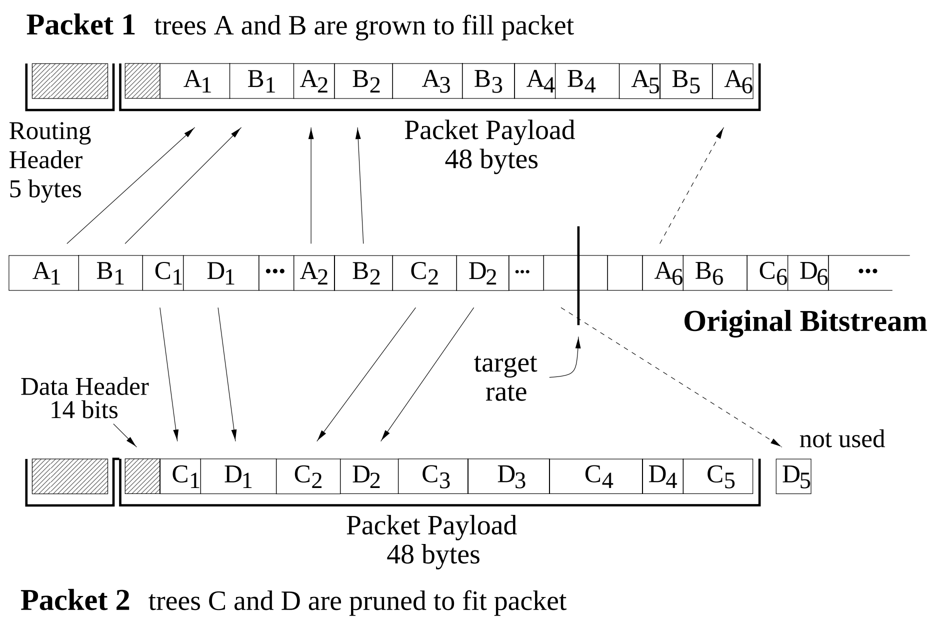

Packet 1 trees A and B are grown to fill packet
A1 B1 A2 B2 A3 B3 A4 B4 A5 B5 A6
Routing Header 5 bytes
Packet Payload 48 bytes
A1 B1 C1 D1 ... A2 B2 C2 D2 ... A6 B6 C6 D6 ...
Original Bitstream
target rate
Data Header 14 bits
not used
C1 D1 C2 D2 C3 D3 C4 D4 C5 D5
Packet Payload 48 bytes
Packet 2 trees C and D are pruned to fit packet

Figure 1: The encoder decides to put trees A and B into packet 1, and trees C and D into packet 2. Because trees A and B do not quite fill the packet, they are grown by using additional bits from beyond the target rate. Trees C and D together overfill their packet, and so get pruned.

and pruning a larger set by taking 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.

Growing and pruning lead to spatially varying image quality. Coefficients are not all coded 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 first reading the 14-bit packet header (which tells how many trees are in the packet, and where the first one is located spatially) and then repeatedly cutting the threshold in half and marching through the bit planes until it reaches the end of the packet. Because trees are grown or pruned to fit within the fixed-length segment, no trees span across packet boundaries. Together with the small packet headers, this makes each packet independently decodable and provides robustness against packet loss.

In transmission over packet switched networks, packets which are not received within a specified time because of network traffic or mis-routing are considered “lost”. Those packets will not be available to the decoder. Also, a packet may be discarded by the decoder (considered lost) if errors are found in the packet payload after arrival. By dedicating 16 bits of the payload to a CRC, the decoder could have a high probability of error detection over the payload. Because of the packet independence, in the event of losses, the decoder fills in as many positions as it can in the wavelet coefficient array using all successfully received packets; missing wavelet coefficients are then replaced by zeros. If errors were not detected, the wavelet coefficients corresponding to trees in the erroneous packet would not be replaced by zeros, but would be placed with wrong values in the array; nonetheless the undetected errors cannot propagate beyond the packet boundary, and trees of coefficients in other packets would be unaffected.

2.1 Refinements

In practice, refinements to this basic idea are needed for the algorithm to work well:
• The effects of a lost packet can be mitigated by interpolation. We used a simple averaging: the decoder interpolates missing coefficients in the low-low band by averaging together as many immediate 8-neighbors as are available. Missing coefficients in other bands are replaced by zeros prior to inverse transforming the entire group.
• A raster scan order for the 1024 trees packs neighboring trees of coefficients together; there will be fewer 8-neighbors present for interpolation in the event of packet loss. So we order the 1024 trees with a recursive tessellation technique [6] used to generate dispersed-dot dither patterns, ensuring that trees in each packet come from widely dispersed locations in the image (see Figure 2).
• If four bits specify the number of trees in the current 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-database images tested, the number of trees per packet remained strictly between 1 and 15 for the bit rates of interest (0.1 to 0.4 bpp), but the algorithm can handle trees of any size.
• Header information such as the image size and the starting threshold can be handled in a number of ways. If the system always operates on a fixed-size image, the size does not need to be stated. Otherwise, that information can be provided redundantly in several different packets. Each packet could use its own starting threshold. For example, two bits within each packet could 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 would be used to indicate this special condition.

2.2 Results

The PZW algorithm was used to compress the 512×512 8 bpp grayscale images Lena and Peppers. The initial (progressive) wavelet coding target rate was 0.2 bpp. After packetization, the actual rates achieved were 0.209 for Lena and 0.208 for Peppers. These higher rates include the 14-bit overhead for each packet, as well as the effects of growing and pruning trees within each packet. The PSNRs achieved at these rates (for 4 cases: all packets arriving, 1%, 10%, and 20% packets erased) are shown in Table 1 along with the PSNRs for the SPIHT algorithm with and without arithmetic

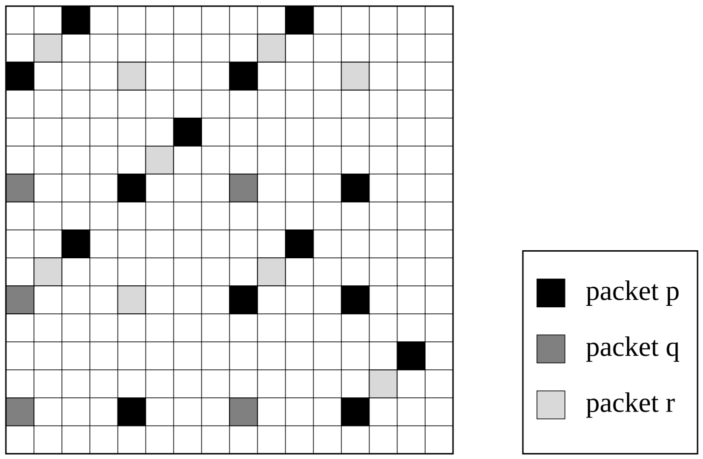


Figure 2: Position of tree heads in a low-low band of size 16x16. Packet p contains 14 trees, packet q contains 5 trees, and packet r contains 9 trees. Packets contain trees that are not spatial neighbors. In the event of packet loss, more 8-neighbors of the missing trees will be present, allowing for better interpolation.

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

Table 1: PSNR results for compressing Lena and Peppers to 0.21 bpp.

coding. In the case of no loss, the PZW algorithm pays a penalty of about 1.1 dB relative to the original SPIHT algorithm. With 1% loss, the quality remains high, and even at loss rates as high as 20% to 40% the images remain recognizable and catastrophic derailment is prevented. The PSNRs were obtained by averaging the mean squared errors for 10,000 random realizations of the packet erasures. The burstiness of the packet erasures makes no difference, since all packets are equivalent a priori. Figure 3 shows a comparison between the original 512×512 grayscale Lena image and reconstructions compressed and packetized at 0.209 bpp for packet erasure rates of 0%, 1% and 20%.

3 Region-Based Compression

We can take advantage of PZW's spatially varying quality to create a region-based compression scheme in which a small region of interest (ROI) gets reproduced with higher quality than the rest of the image. We discuss two methods for this; both assume that a user has specified the ROI and its desired (approximate) quality level. In method 1, all trees with coefficients in the ROI are coded to a high rate (to achieve the desired quality), while trees outside the ROI are coded to a low rate (to provide context). Because the final threshold (encoded bit plane) must be the same for all trees within one packet, a packet which contains both trees from inside and outside the ROI must increase the bit rate of the non-ROI trees in order to maintain the specified quality for the ROI trees. This spreads the higher quality to parts of the image outside the ROI, an effect which can be minimized by changing the tree packing order to a row and column interlaced raster scan (which is more likely to keep ROI trees together in a packet, but still does not have any immediate neighbors in the same packet). Because the decoder does not need to know the final threshold for any tree, no side information is required to indicate the region of interest. The decoder merely decodes some packets which contain only 1 or 2 trees (coded at high quality, presumably from the ROI), and other packets with a dozen trees (low bit rate per tree, presumably not from the ROI) but no vulnerable side information ever needs to be sent to the decoder to explicitly state the location or quality of the ROI.

A second method for achieving this goal is a packetized version of Shapiro's region-enhancement EZW algorithm [7]. All coefficients in ROI trees are pre-multiplied by some factor > 1.0, increasing their apparent significance. They will be coded using more bits. Side information required by the decoder includes the pre-multiplication factor and the parameters necessary to specify the region. In this method, although the actual terminating threshold for each tree in any one packet is the same, the *effective* terminating threshold for trees within the ROI is larger (scaled by the pre-multiplication). This allows for a better distribution of bits when packing trees from both inside and outside the ROI in the same packet. Region-based compression and packetization with method 2 are illustrated in Figure 4. At 0.06 bpp, the name of the ship (the ROI) is illegible. With a scaling factor of 4.75 for ROI trees, the final rate is 0.063 bpp, and we get high quality in the ROI (PSNR for ROI = 30.52 dB) while maintaining context information. Table 2 shows numerical results of both methods for the Ship image at different overall bit rates. The results illustrate the problem described for method 1; images have less overall distortion, but the quality of the ROI is not as high. Method 2 more effectively dedicates bits to the ROI, giving it higher quality and leaving the non-ROI area at low quality.

			PSNR (dB)		
	Rate (bpp)	ROI factor	overall	within ROI	outside ROI
Method 1	0.086	0.40	28.44	30.86	28.41
Method 1	0.116	0.40	30.29	31.61	30.26
Method 2	0.086	4.75	28.06	32.04	28.02
Method 2	0.116	4.75	29.66	33.68	29.62

Table 2: PSNR results inside and outside the region of interest for the Ship image. The region of interest included pixels $[300-380, 300-370]$. The ROI factor represents the ROI rate (bpp) for method 1, and the multiplicative factor for method 2.

4 Conclusions

The wavelet zerotree compression and packetization algorithm presented here is robust against packet erasure without a requirement for retransmission or FEC. This might be useful for channels with long round-trip delays, or for real-time interactive systems. By grouping and interleaving coefficient trees of adjusted lengths, the method produces fixed-length segments that provide resynchronization points for the decoding algorithm at fixed intervals, preventing catastrophic derailment. A controlled degradation in image quality results as more packets are dropped. The algorithm also is easily extended to perform region-based compression. The additional rate required is low, and little or no side information is required by the decoder.

PZW may be useful in a non-packet-based network as well, since the sequence of 48-byte payloads can be streamed together and would guarantee re-synchronization between the decoder and encoder after any bit errors. The method could be combined with FEC to provide both error correction and also prevention of derailment when the error correction capability is exceeded. By adjusting the payload length (resynchronization interval), the number of decomposition levels, and the amount of FEC, a variety of trade-offs between compression performance, noise robustness, and resilience to error burstiness could potentially be obtained.

References

[1] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41:3445–3462, December 1993.

[2] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Ciruits and Systems for Video Technology*, 6(3):243–249, June 1996.

[3] H. Man, F. Kossentini, and M.J.T. Smith. Robust EZW image coding for noisy channels. *IEEE Signal Processing Letters*, 4(8):227–229, August 1997.

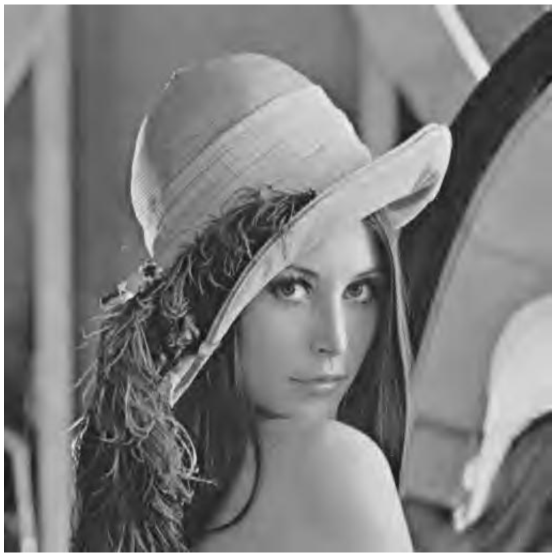
[4] C.D. Creusere. A new method of robust image compression based on the embedded zerotree wavelet algorithm. *IEEE Transactions on Image Processing*, 6(10):1436–1442, October 1997.

[5] V.J. Crump and T.R. Fischer. Intraframe low bitrate video coding robust to packet erasure. In J.A. Storer and M. Cohn, editors, *Proceedings: DCC '97, Snowbird, Utah*, page 432, Los Alamitos, CA, 1997. IEEE Computer Society.

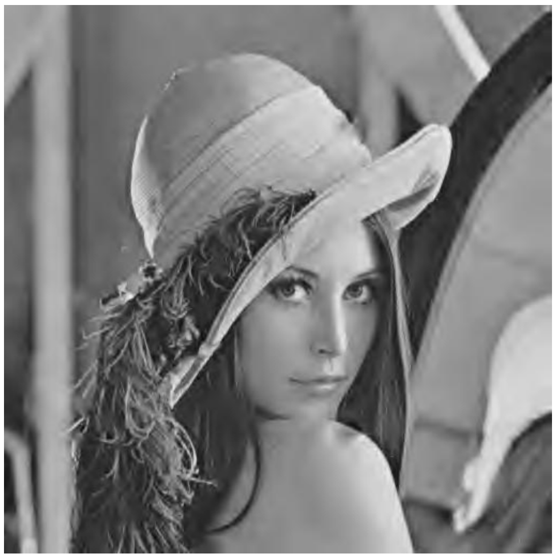
[6] R. Ulichney. *Digital Halftoning.* MIT Press, Cambridge, Mass, 1987.

[7] J.M. Shapiro. US Patent # 5563960: Apparatus and method for emphasizing a selected region in the compressed representation of an image, October 1996.

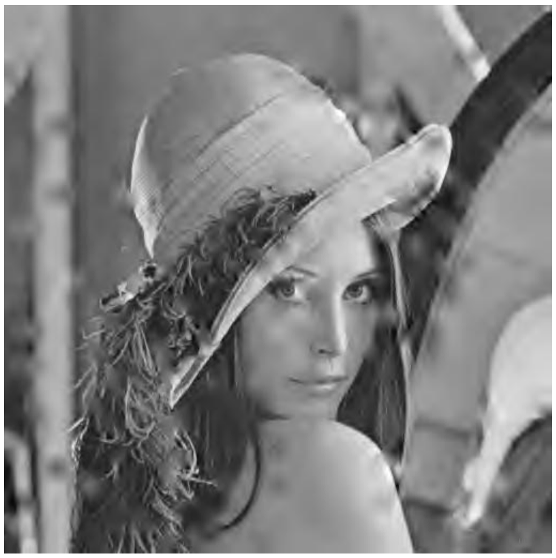
(a)



(b)

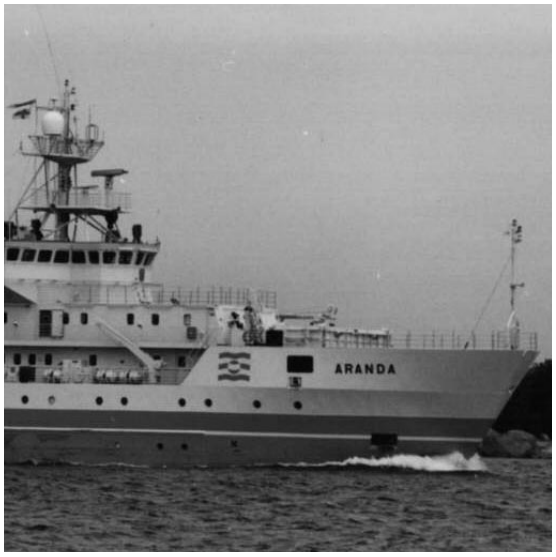


(c)

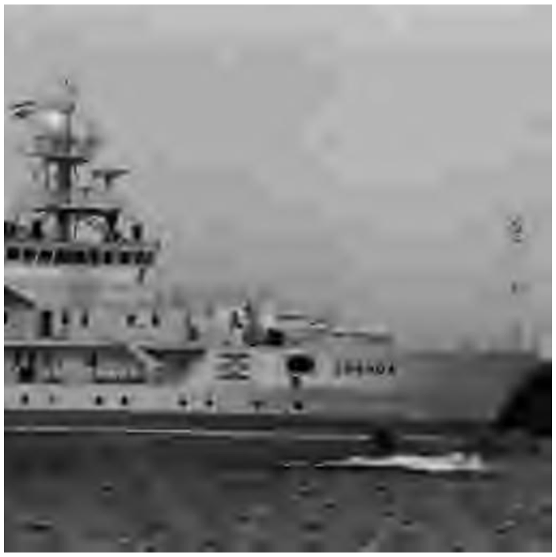


(d)

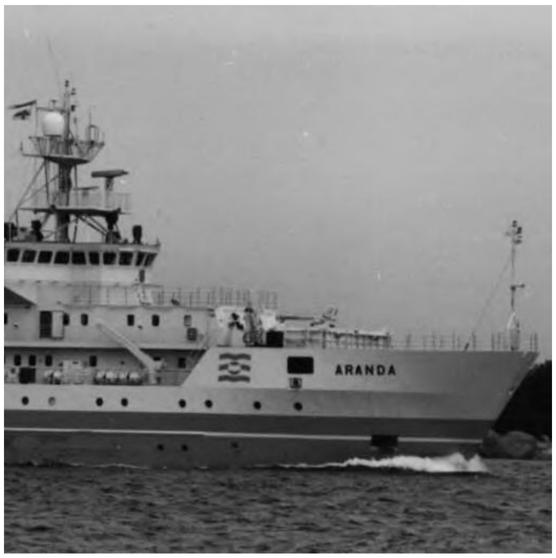
Figure 3: (a) Original Lena image. (b), (c) and (d) are compressed and packetized at 0.209 bpp with 0%, 1% and 20% of packets erased, respectively. The corresponding PSNRs are 32.2, 31.1, and 25.4 dB.



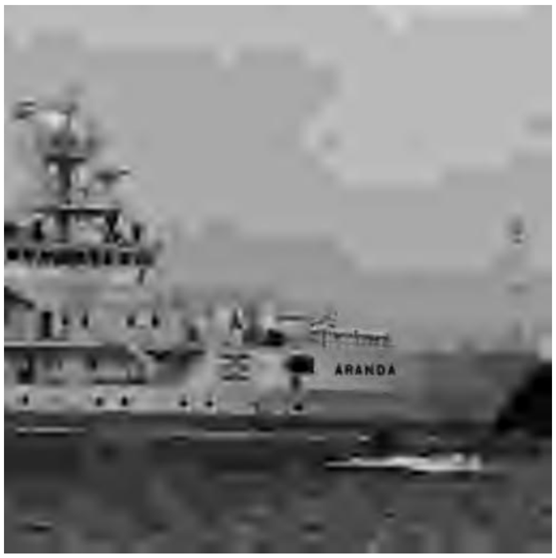
(a)



(b)



(c)



(d)

Figure 4: (a) Original image, (b) PZW compressed at 0.06 bpp, (c) PZW compressed at 0.5 bpp, (d) region-based PZW results: $ROI = [300 - 380, 300 - 370]$ is enhanced with a multiplier of 4.75, requiring an overall rate of 0.063 bpp