Robust Transmission of JPEG2000 Codestreams over Packet Erasure Channels *

Zhenyu Wu, Ali Bilgin, Lingling Pu and Michael W. Marcellin

Dept. of Electrical and Computer Engineering,
The University of Arizona, Tucson, AZ, 85721

ABSTRACT

In this paper, a robust image transmission scheme based on JPEG2000 is proposed for packet erasure channels. Error resilience functionalities provided by JPEG2000 are utilized to control the source coding efficiency and the robustness according to channel conditions. Furthermore, together with the proposed interleaving scheme, some erasures can be recovered. Experimental results show the effectiveness of the scheme.

Keywords: JPEG2000, error resilience, packet erasure channel

1. INTRODUCTION

With the fast development of computer networks, more and more data, including compressed images, are transmitted through packet switched networks. However, during transmission, channel packets could be lost at random when the number of packets sent exceeds transmission capacity. New generation of image codecs, such as SPIHT[1] and JPEG2000 [2] , can provide efficient compression for transmission purposes. Through the extensive use of context-based entropy coding, their codestreams are very sensitive to these packet losses. So it is important to make these image coders more error resilient.

Recently, several methods have been proposed for wavelet based image coders to improve their error resilience and packetization efficiency over noisy channels. In [3] , a modified embedded zerotree wavelet (EZW) [4] image compression algorithm was proposed. Its encoder partitions the wavelet coefficients into several groups and then quantizes and codes each of them independently. The resulting multiple codestreams are interleaved and then sent through the channel. The corresponding decoder can use all of the bits received before the occurrence of the first error in each codestream to reconstruct the image. So the impact of a single bit error is localized to only one codestream and this codestream is decoded partially to its reduced fidelity, while the other codestreams are unaffected and can be decoded to their full fidelity. By forming more independent codestreams, the correlation among wavelet coefficients is exploited in a smaller region by the encoder, but a single bit error affects a smaller part of the total codestream. Therefore, error robustness and source coding efficiency can be traded off by forming differing numbers of independent codestreams.

In [5] , the authors proposed a combined wavelet zerotree coding and packetization method (PZW) for packet erasure channels. It modifies the set partitioning in hierarchical trees (SPIHT) [1] coder such that complete trees of wavelet coefficients are contained within each fixed-length network packet. It provides graceful degradation in image quality as packet losses increase because each packet is independently decodable. This comes at the cost of source coding efficiency. Efficient packetization for embedded bitstreams is further studied in [6] . It solves the problem of minimizing packetization inefficiency due to bitstream alignment, which is brought up by partitioning encoded bitstreams into channel packets for error resilience purposes. Theoretically optimal packetization (OP) schemes were proposed for packet erasure channels, along with suboptimal schemes with less complexity.

All the schemes discussed above combat channel noise by modifying source coders to introduce error resilience to the coded bitstreams. The underlying idea is to localize the error influence to a small segment of the bitstream so that it will not propagate to derail the entire decoding. This is achieved by various data partitioning and resynchronization techniques.

*This work was supported in part by the National Science Foundation under Grant No. CCR-9979310 and ANI-0325979.

Visual Communications and Image Processing 2004, edited by Sethuraman Panchanathan, Bhaskaran Vasudev, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 5308

Another approach to combat noise is to use channel coders concatenated with source coders. A channel coder adds controlled redundancy into the source-encoded bitstreams [7] [8] [9] . With this extra redundancy, errors or erasures can be not only detected but also recovered. However, within a total transmission rate, channel coding decreases the rate available to the source coding, and it also adds extra complexity to both sender and receiver due to the channel coding.

In this paper, a new scheme is proposed for robust image transmission over ATM packet erasure channels with JPEG2000. It achieves source coder robustness against packet loss by utilizing some error resilience functionalities provided by JPEG2000. It does not require any modification of standard-compliant JPEG2000 coders, and no channel coding technique is employed. This scheme results in a high complexity decoder. However, the encoder is spared the extra complexity of channel coding as commonly used to improve robustness. Experimental results show the effectiveness of the scheme compared with other schemes.

The paper is organized as follows: in Section 2, the proposed scheme is described as well as the related concepts of JPEG2000. Section 3 gives the experimental results and in Section 4 we draw conclusions.

2. ALGORITHM DESCRIPTION

JPEG2000 coding is based on the extensive use of context-based arithmetic coding. Thus it is crucial to maintain the synchronization between encoder and decoder. However, a bit error in a codestream can easily destroy the synchronization and thus derail the the decoding thereafter, resulting in disastrous effects on the decompressed image. To combat this problem, JPEG2000 provides some useful mechanisms for error resilience. They generally serve two purposes [10]: (1) hierarchical data partitioning and resynchronization, (2) error detection and isolation.

A JPEG2000 codestream can be partitioned into hierarchical structures. Each codestream starts with a main header. It contains critical information for the decoder to decode the entire codestream correctly. The codestream following the main header can be partitioned into a single or multiple tile-streams, depending on whether the input image is divided into multiple tiles before the wavelet transform. Inside each tile-stream, there are one or several layers and each layer contains a certain number of packets. Each packet collects the compressed data from a precinct at a resolution level. Each precinct consists of some codeblocks from all subbands at its resolution level and finally each codeblock contains a certain number of coding passes. JPEG2000 error resilience tools are built upon these different structures.

Within this hierarchy, the smallest independent coding unit is the codeblock. That means bit errors from one codeblock will not propagate to other codeblocks as long as codeblock synchronization is maintained. In other words, resynchronization can be reestablished at the codeblock level. Similar mechanisms exist for resynchronization at the level of precincts and/or tiles. Since they are not used in this work, they are not discussed further.

Besides this hierarchical data partitioning and resynchronization, JPEG2000 provides a set of mode variations for error detection and isolation purposes. In its "default mode", a JPEG2000 bitplane coder produces one contiguous arithmetic codeword for each codeblock. If there is any error inside a codeword segment, undetected erroneous decoding of this entire segment (beyond the error) may occur, causing severe quality degradation of the decompressed image.

When the JPEG2000 mode *ERTERM* is employed, the encoder uses a predictable termination policy for MQ and raw codeword segments. Any error in a codeword segment is very likely to leave the coder in a state which is inconsistent with the predictable termination policy. So the decoder can exploit this property to detect whether errors have been introduced into any individual codeword segment with high reliability. When the *RESTART* mode is employed, the MQ coder is restarted at the beginning of each coding pass. Specifically, an MQ codeword is appropriately terminated and the MQ coder is re-initialized (excluding its probability models). Each coding pass is then represented by a separate MQ codeword segment. The length of each segment is explicitly signalled in the relevant packet header. Both modes come with a cost of some source coding inefficiency.

When *ERTERM* and *RESTART* are used together, an encoder creates a separate, predictably terminated codeword segment for each coding pass. If an error occurs in the codestream, a decoder can detect it with high probability at the end of the coding pass in which it occurred. With the length information signalled in the

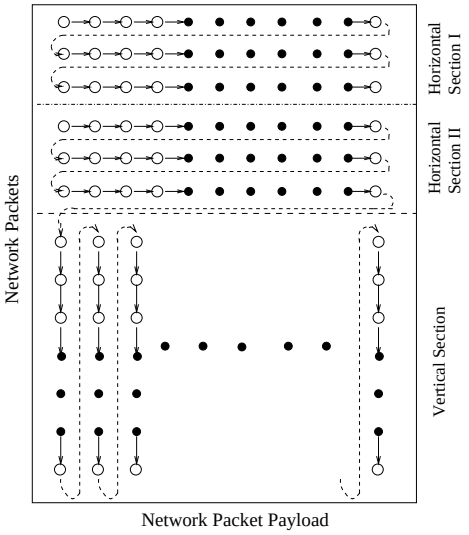


Figure 1. Proposed Interleaving Scheme

packet header, the decoder can discard only the corrupt coding pass (and all subsequent coding passes from that same codeblock), thereby concealing the possible visual artifacts which would otherwise result from such corruption.

Inside a JPEG2000 codestream, besides the main header, tiles, tile-parts and packets each have a header. These headers are important for the decoder to correctly decode the data from their corresponding part of the codestream. Also they contain the information necessary to establish synchronization at their respective levels. JPEG2000 provides a useful functionality referred to as *packed packet headers* (PPM or PPT). They can be used to relocate packet headers from their respective packet bitstreams to the main (or tile-part) headers to facilitate special protection for them.

As discussed above, the JPEG2000 standard contains error resilience mechanisms to detect and isolate errors, and conceal the error impact within a certain coding structure. However, there is no forward error correction capability. Nevertheless, we will show that by appropriately utilizing these error resilience mechanisms, together with our proposed interleaving scheme, erasures can be recovered.

Consider a JPEG2000 codestream formed with modes *ERTERM* and *RESTART*, and with all the headers at different structural levels packed into the main header at the beginning of the codestream by invoking *packed packet header* functionality. The proposed scheme then partitions the codestream into ATM network packets in two different sections, which we call the horizontal section and the vertical section, as shown in Figure 1. In this figure, each row corresponds to one 48-byte ATM packet payload.

In the first horizontal section, the main header is packed horizontally byte-wise into the payload of ATM packets. Since the main header contains critical information for correct decoding, it needs to be protected adequately. With the assumption that no channel coding is involved, we take the approach to repeat the main header to increase its robustness against packet erasures [†]. Thus the horizontal section contains two copies of the main header. Since the main header takes a relatively small part of the codestream, the duplication does not decrease compression efficiency significantly, while it can dramatically increase the protection for the main header. This approach is also used in the MPEG-4 header extension code (HEC), where vital information for correct decoding is repeated after the HEC, and has proven successful [10] . We note here that the repeated data can be put in a JPEG2000 "COMMENT" marker segment so that the result is a valid codestream.

[†]we ignore that this can be considered as a crude form of channel coding

In the vertical section, the compressed image data are interleaved bit-wise into the remaining packets (vertically). That is, the data is filled into the first bit of each packet continuously in the vertical direction, then the second bit, and so on, until the end of either the bitstream or the packets is reached.

Consider the situation when the packet erasure rate is relatively low. With the proposed interleaving scheme, in the horizontal section, if any packet is erased, it is very likely that its duplicate survives. Thus the main header can be reconstructed by combining the two parts from that section together, and the reconstruction can be expected to succeed with high probability.

In the vertical section, compressed image data are fully interleaved across the packets. For every erased packet in this section, the 48-byte erasures are spread out into 384 bit erasures in many coding passes from different codeblocks within the codestream. So at relatively low packet erasure rate, every coding pass may only contain a few erased bits. When the main header is reconstructed perfectly, coding pass length information can be retrieved from the corresponding packet headers. So the decoder knows the position of each coding pass within the codestream. Together with the knowledge of which packets are erased, the decoder can deduce the positions of the erased bits inside each coding pass (if any).

As discussed before, when the *ERTERM* and *RESTART* modes are used, the decoder can detect an error inside a coding pass with high reliability. Therefore, if the number of erasures inside an affected coding pass is small, the decoder can try to recover the erasures by simply brute-force bit-flipping with 0 or 1 at these erased bits until the attempted coding pass is decoded with a consistent termination state. Consequently, with high probability, a coding pass with erasures can be completely recovered. This procedure can be carried out within each codeblock from the first coding pass containing erasures and then repeated sequentially for the rest of coding passes. Suppose some coding pass contains more erasures than the decoder is willing to attempt to recover (for complexity reasons). In this case, the decoder simply stops decoding and discards the corrupted coding pass and all subsequent coding passes within the same codeblock. The decoded coding passes from the same codeblock and the decoding of other codeblocks are unaffected.

Obviously, the chance of successful decoding of a coding pass depends on the number of erasures it contains, and the computational capacity of the decoder. In the worst case, for n erasures, the decoder must test 2^n cases. Fortunately, the data partitioning of JPEG2000 can be adjusted to control of the tradeoff between complexity and system performance. As mentioned before, the codeblock is the smallest independent coding unit. With a large codeblock size, correlation can be exploited in a larger area so high compression performance can be achieved. However, this will generate long coding passes, which span more packets vertically and therefore contain more erasures. This weakens the recovery capability of the decoder and increases the likelihood that these coding passes will be discarded. On the other hand, with smaller codeblock sizes, source coding performance degrades. However, the more coding passes are generally shorter and contain fewer erasures, and so are more recoverable by the decoder. Also, with smaller codeblocks, there are more codeblocks resulting in more independently decodable sub-codestreams. A key benefit of this is that any unsuccessful recovery of a coding pass is contained to a smaller region.

In summary, codeblock size can serve as a parameter to adjust the tradeoff between source efficiency and robustness. When the packet erasure rate is low, large codeblock sizes can be chosen to achieve good compression performance; as the erasure rate increases, the codeblock size can be reduced to generate more robust codestreams accordingly.

3. EXPERIMENTAL RESULTS

In the following experiments, Lenna and Goldhill (512×512) are used as our test images. Transmission is simulated through an ATM network with 48-byte packet payload, with total transmission rates 1.0005, 0.5024 and 0.2095 bpp, which correspond to 683, 343 and 143 channel packets respectively. We choose the codeblock size parameter from the set $\{8 \times 8,\ 16 \times 16,\ 32 \times 32\}$. Each JPEG2000 codestream is generated with a single layer, the desired codeblock size, *ERTERM+RESTART* modes turned on and PPM marker segment employed. The source compression rate is chosen such that after the main header duplication, the total length does not exceed the allowed transmission budget.

Source coding rates and PSNRs for the error free case (no erasures) are listed in Tables 1 and 2. For example, for a total rate of 1.0005 bpp, our scheme (using 32×32 codeblocks) allocates 0.95 bpp for source rate and achieves 39.80 dB PSNR in the error free case. The difference between 1.0005 bpp and 0.95 bpp accounts for the replicated main header. Also listed in the tables (for source efficiency comparison) are two cases that do not employ our scheme. Each uses 64×64 codeblocks. The first case uses *ERTERM+RESTART*, while the second case uses 20 quality layers.

settings	Total Rate		
	1.0005 bpp	0.5024 bpp	0.2095 bpp
32x32/ERTERM+RESTART/Repeat Header	0.95/39.80	0.47/36.65	0.19/32.50
16x16/ERTERM+RESTART/Repeat Header	0.91/39.07	0.45/35.91	0.18/31.88
8x8/ERTERM+RESTART/Repeat Header	0.85/37.87	0.42/34.66	0.18/30.65
64x64/ERTERM+RESTART	1.00/40.25	0.50/37.17	0.21/33.05
64x64/20-layer	1.00/40.33	0.50/37.26	0.21/33.13

Table 1. Error free source coding performance for lenna: Source Rate (bpp)/PSNR (dB).

settings	Total Rate		
	1.0005 bpp	0.5024 bpp	0.2095 bpp
32x32/ERTERM+RESTART/Repeat Header	0.95/35.95	0.47/32.68	0.19/29.52
16x16/ERTERM+RESTART/Repeat Header	0.91/35.19	0.45/32.09	0.18/29.13
8x8/ERTERM+RESTART/Repeat Header	0.85/33.77	0.42/31.06	0.17/28.38
64x64/ERTERM+RESTART	1.00/36.45	0.50/33.14	0.21/29.94
64x64/20-layer	1.00/36.53	0.50/33.19	0.21/29.99

Table 2. Error free source coding performance for goldhill: Source Rate (bpp)/PSNR (dB).

The resulting codestreams were transmitted through a memoryless packet erasure channel with erasure rates ranging from 1% to 10%. Kakadu v3.4 was used as the source coder, and each case was tested with 2000 trials. The results are direct average of PSNRs. In the experiments, we allow the decoder to be able to correct 10, 15 or 20 erasures in each coding pass respectively, corresponding to different levels of decoding effort. The results for different total transmission rates, codeblock sizes, and maximum recoverable erasures are shown in Figure 2. From the figures, codeblock size 32×32 can give the best performance at very low erasure rates (1% or 2%) with high decoding effort. However, the performance drops quickly as the erasure rate is increased. In the middle erasure rate range (3% to 8%), codeblock size 16×16 is the best choice. We note that there are significant gaps in performance as a funcion of different numbers of recoverable erasures (decoding effort). Throughout the entire testing range, a codeblock size of 8×8 gives most stable performance, most of the time within 2 dB of the best possible performance. Additionally, the 8×8 case begins to show its advantage at high erasure rates (9% to 10%). Furthermore, the performance difference between different numbers of recoverable erasures are negligible, which implies that high decoding effort is unnecessary in the 8×8 case.

Figure 3 gives the overall best performance among 3 codeblock size choices based on the results in Figure 2. In other words, Figure 3 represents the best possible performance by adjusting the codeblock size parameter given the maximum number of recoverable erasures at each erasure rate. At the same time, our results are compared with those from schemes OP [6] and PZW [5] in Figure 3 (a) and (c). In Figure 3(a) which has high source compression rate (1.0005 bpp), our scheme performs better than OP by 3 to 4 dB even with the least decoding effort at low erasure rates. This margin diminishes until the erasure rate reaches 5% and thereafter OP begins performing better if higher decoding effort is not used in our scheme. In Figure 3(c) with low source compression rate (0.2095 bpp), our scheme consistently gives better performance than OP and PZW at erasure rates 1% to 9% even with the least decoding effort.

Finally, our scheme is compared with two other codestream organization methods of JPEG2000 in Figure 4. The first is to generate a codestream with a single layer using *ERTERM*+RESTART and packing the codestream into channel packets horizontally. Decoding of each codeblock is attempted, with all data prior to the first erasure

in each codeblock retained. This method differs from our scheme mainly in that we spend bits reproducing the header, and special interleaving is employed such that some erasures can be recovered. The second scheme generates a quality-progressive codestream with 20 layers and the codestream is packed horizontally into the channel packets. Source decoding stops with the first erased packet. As mentioned previously, the error free performance of these schemes is listed in Tables 1 and 2. From Figure 4, we see significant gains of our scheme compared with the other two in all cases. This shows that the gain we have over OP and PZW is due to our proposed scheme, and not just the inherent error resilience of JPEG2000.

The applicability of the proposed scheme relies heavily on the efficient decoding of a JPEG2000 codestream. With Kakadu v3.4, the decoding of all codeblocks for Lenna and Goldhill (512×512) at 1.00 bpp both takes about 0.04 seconds on a Pentium III 1GHz PC. With 10 maximum recoverable erasures, assuming the worst case of every coding pass need recovery, the average decoding time is about 21 seconds.

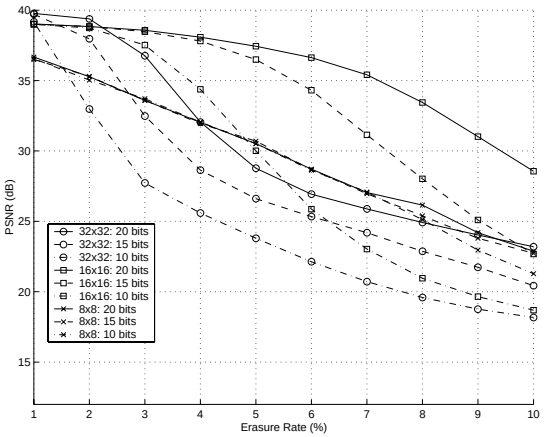
Although the present work uses no channel coding, it is possible to combine it with channel coding to improve system performance. Channel coding can reduce the number of erasures for the source decoding to handle, and some erasures unrecoverable by the channel decoding can be salvaged by the source decoding and thus leads to improvement over the original channel coding capability. In [11] , a similar idea is explored for LDPC codes with AWGN channels. Besides proposing a practical method for robust image transmission, this work reveals some potential contributions of error resilience functionalities other than those in the standard, in the case of packet erasure channels. This will help form a more complete evaluation of their importance.

4. CONCLUSION

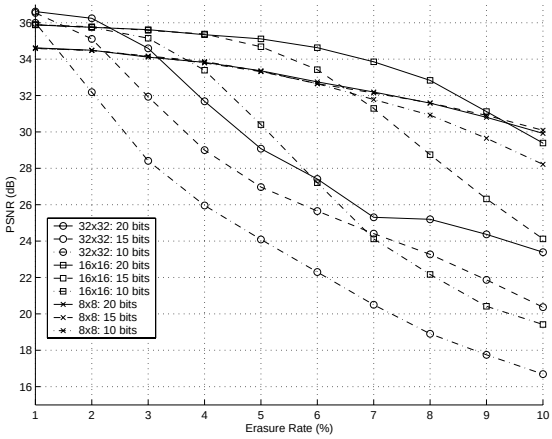
In this paper, a scheme is proposed based on JPEG2000 for robust image transmission over packet erasure channels. The robustness against packet erasures is achieved by exploiting only the properties of error resilience functionalities provided by JPEG2000 combined with a unique interleaving strategy. At the same time, it is also possible to combine this scheme with other channel coding techniques to give better performance. Furthermore, the importance of some error resilience functionalities of JPEG2000 is explored in a wider perspective.

REFERENCES

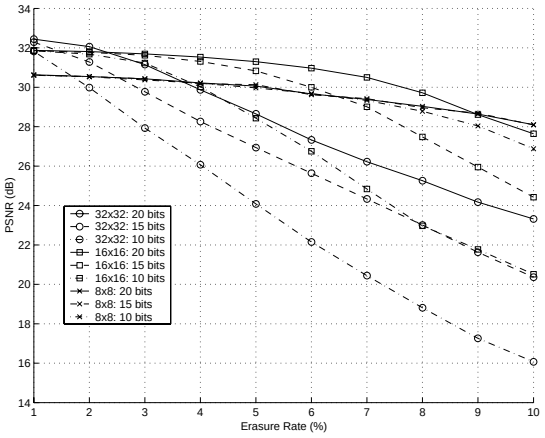
1. A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuts Syst. Video Technol.* **6**, pp. 243–249, June 1996.
2. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Practice and Standards*, Kluwer Academic Publishers, Massachusetts, 2002.
3. C. D. Creusere, "A new method of robust image compression based on the embedded zerotree wavelet algorithm," *IEEE Trans. Image Processing* **6**, pp. 1436–1442, Oct. 1997.
4. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing* **41**, pp. 3445–3462, Dec. 1993.
5. J. K. Rogers and P. C. Cosman, "Wavelet zerotree image compression with packetization," *IEEE Signal Processing Letters* **5**, pp. 105–107, May 1998.
6. X. Wu, S. Cheng, and Z. Xiong, "On packetization of embedded multimedia bitstreams," *IEEE Trans. Multimedia* **3**, pp. 132–140, Mar. 2001.
7. A. E. Mohr, E. A. Riskin, and R. E. Ladner, "Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction," *IEEE J. Selected Areas in Communications* **18**, pp. 819–828, June 2000.
8. V. Stankovic, R. Hamzaoui, and Z. Xiong, "Packet loss protection of embedded data with fast local search," in *Intl. Conf. on Image Processing*, **2**, pp. 165–168, 2002.
9. J. Thie and D. Taubman, "Optimal protection assignment for scalable compressed images," in *Intl. Conf. on Image Processing*, **3**, pp. 713–716, 2002.
10. I.Moccagatta, S.Soudagar, J. Liang, and H. Chen, "Error-resilience coding in JPEG-2000 and MPEG-4," *IEEE Trans. on Selected Areas on Communications* **18**, pp. 899–914, June 2000.
11. L.Pu, Z.Wu, A. Bilgin, M.Marcellin, and B. Vasic, "Iterative joint source-channel decoding for JPEG2000," in *Asilomar Conference on Signals, Systems, and Computers*, 2003. (to appear).



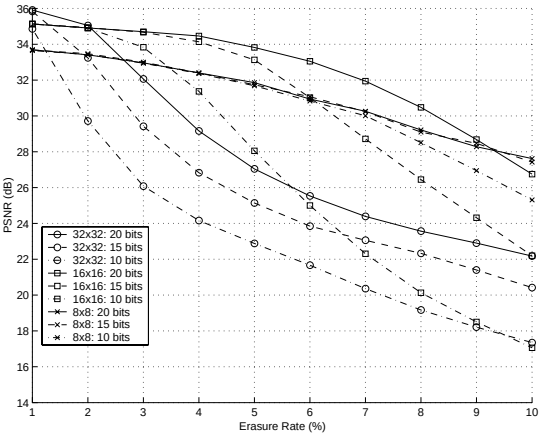
(a) lenna 1.0005 bpp.



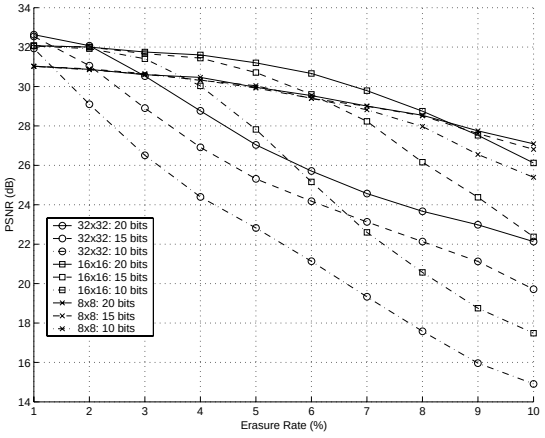
(b) lenna 0.5024 bpp.



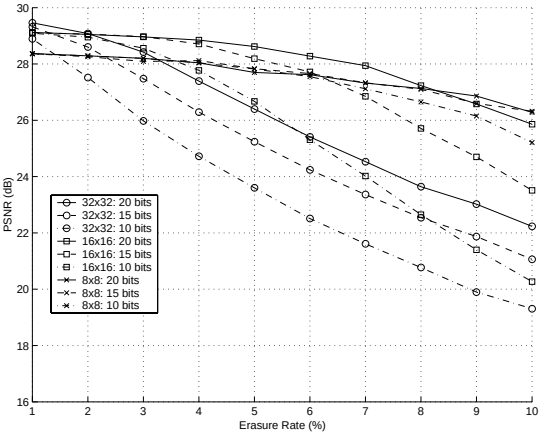
(c) lenna 0.2095 bpp.



(d) goldhill 1.0005 bpp.

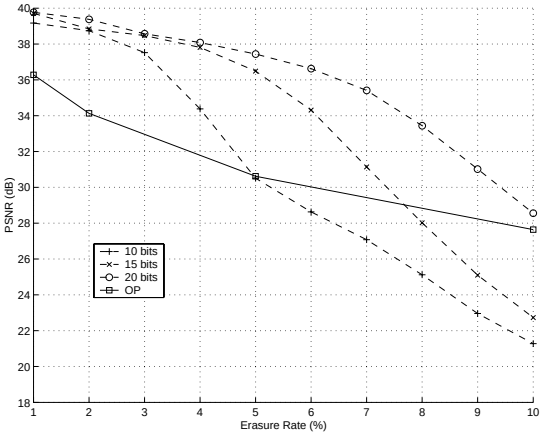


(e) goldhill 0.5024 bpp.

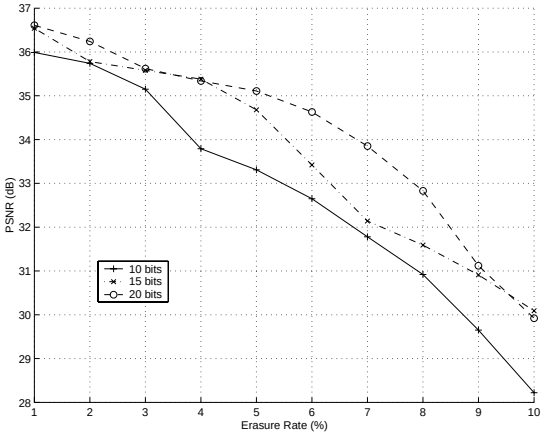


(f) goldhill 0.2095 bpp.

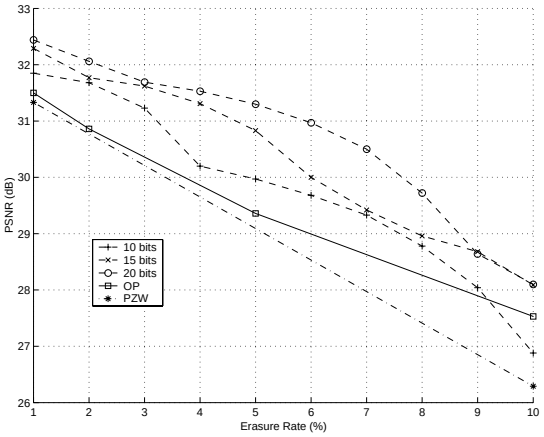
Figure 2. Lenna and goldhill (512x512) with different codeblock sizes and number of recoverable erasures.



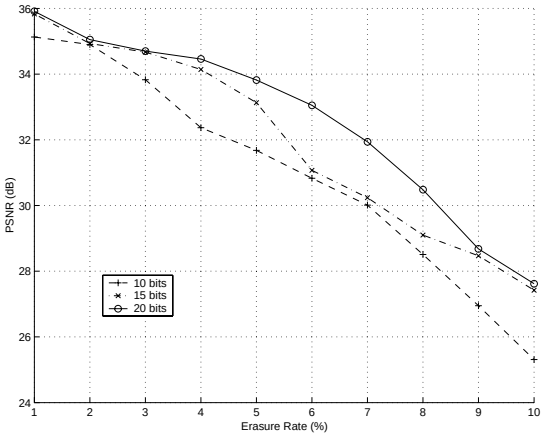
(a) lenna 1.0005 bpp.



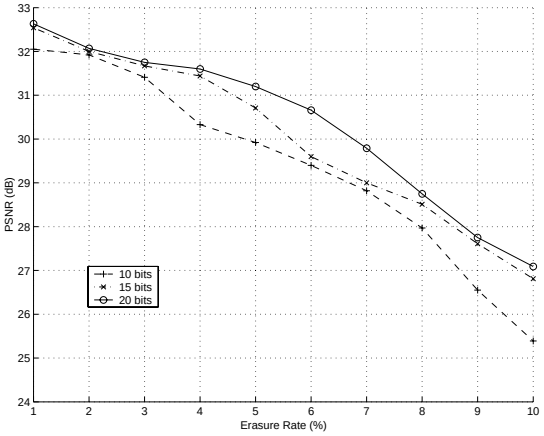
(b) lenna 0.5024 bpp.



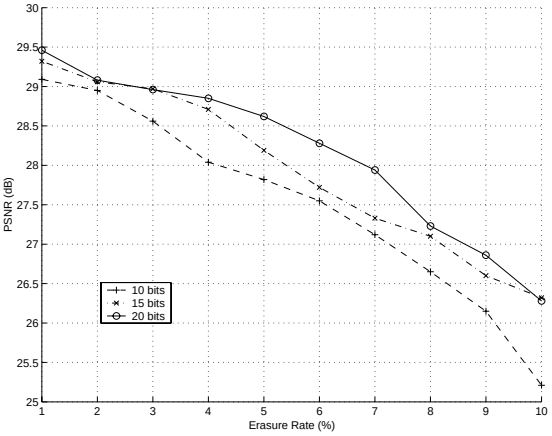
(c) lenna 0.2095 bpp.



(d) goldhill 1.0005 bpp.

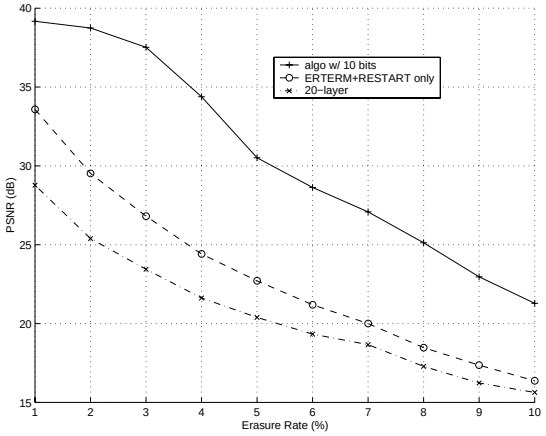


(e) goldhill 0.5024 bpp.

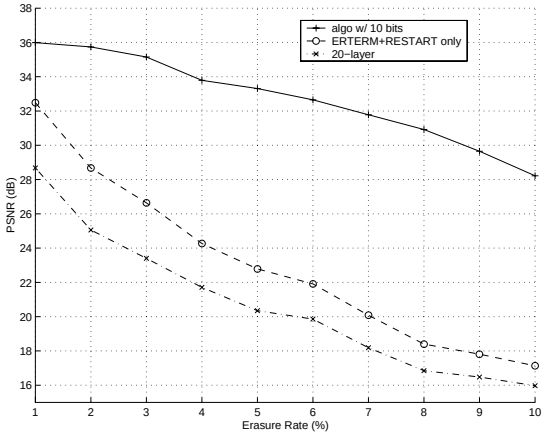


(f) goldhill 0.2095 bpp.

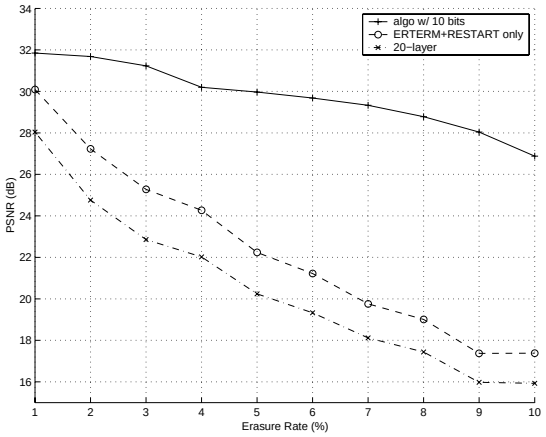
Figure 3. Lenna and goldhill (512x512) best achievable performance with 10,15, and 20 recoverable erasures.



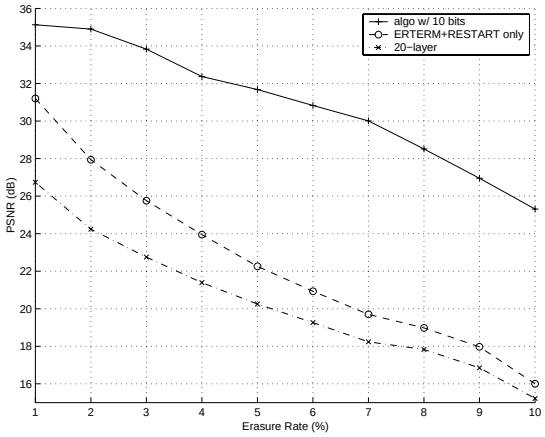
(a) lenna 1.0005 bpp.



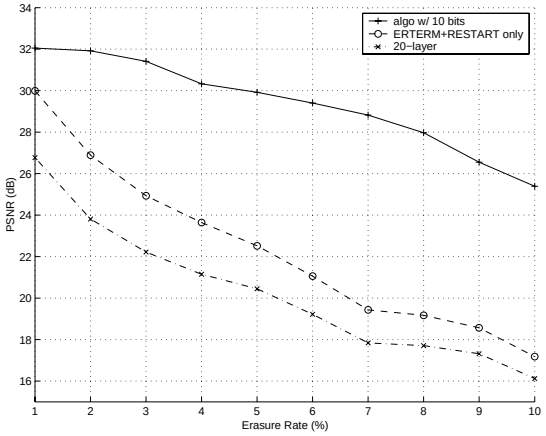
(b) lenna 0.5024 bpp.



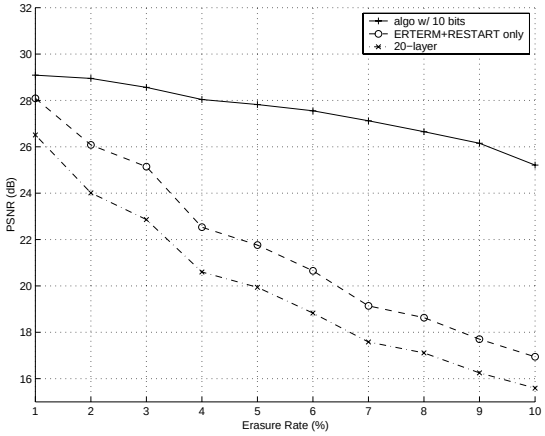
(c) lenna 0.2095 bpp.



(d) goldhill 1.0005 bpp.



(e) goldhill 0.5024 bpp.



(f) goldhill 0.2095 bpp.

Figure 4. Lenna and goldhill (512x512) comparison among 3 schemes.