A MODIFIED JPEG-LS IMAGE CODER WITH INCREASED CHANNEL ERROR ROBUSTNESS

Ofir Shalev[†], Leon Schveiger[†], Ilan Sutskover[†] and Ran Bar-Sella[‡]

[†] SIPL – Signal and Image Processing Lab.
Department of Electrical Engineering - Technion
Technion City, Haifa, 32000, ISRAEL
E-mail: ilan@siglab.technion.ac.il

[‡] Net2Wireless Research
Malat Bldg. , Technion
Technion City , Haifa 32000, ISRAEL
E-mail: barsella@net2wireless.co.il

Abstract

JPEG-LS is a lossless and near-lossless compression algorithm for continuous-tone images. As many variable length source coders, it is highly vulnerable to channel errors. In this paper, some modifications to the JPEG-LS are suggested. The modified algorithm has good robustness to channel errors, while maintaining compression ratio that is close to that of the original JPEG-LS.

Introduction

JPEG-LS [1] (based on the LOCO-I algorithm [2,3]) is a low-complexity lossless image compression algorithm. In order to discuss its vulnerability to channel errors, we now briefly review part of its structure. Only the lossless mode of JPEG-LS is discussed.

The algorithm uses a modeling stage followed by a coding stage. The modeling part consists of 1) Causal prediction of the current pixel, based on adjacent pixels; raster-scan is used to determine the order of the pixels. 2) Determination of the *context* of the pixel. 3) Parameter estimation for the probabilistic model of the prediction residual.

The predictor of the pixel x is a sum of two parts: fixed and adaptive. The fixed part is a causal template that utilizes the pixel's local neighborhood, i.e. pixels[1] a, b and c in figure 1. The adaptive part is an additive integer term that depends on the context of the pixel (called *bias*).

	h		
m	g		
l	f		
c	b	d	e
a	x		

Figure 1: The vicinity of a pixel x

The context of the pixel x is determined by the quantized values of the three gradients surrounding it, $\{d-b, b-c, c-a\}$.

Encoding of the residual is done using Golomb codes. These are tuned by a context dependent parameter k. The code consists of two parts: a binary number represented by k bits, and a unary represented number. Since Golomb code applies to positive integers only, a mapping suggested by Rice is used to map the two sided residual according to the order

[†] This work was conducted at the Signal and Image Processing Lab, EE department, Technion.
[‡] This work was conducted while the author was a research assistant at SIPL.

[1] We abuse notation by denoting the values of the pixels in figure 1 by their name. Hence we may address pixel x as the pixel that is tagged by the name x, and we may speak about the difference in the pixel values as $b-x$.

$\{0,-1,1,-2,2,...\}$. Some very "smooth" areas (flat, if lossless mode is wanted) are encoded using run length encoding rather than pixel by pixel.

Modifications to JPEG-LS

As the compressed code is transmitted through a noisy channel, it is likely to get distorted. A reasonable requirement is that a small amount of flipped bits causes only small distortion in the decoded image. This requirement, however, is not fulfilled by the code of JPEG-LS. Even a flip of one bit may cause complete destruction of the decoded image.

It is obvious that a flip in any bit in the unary part of Golomb-Rice code is catastrophic since synchronization with the input bitstream is lost. Therefore, we assume henceforth that these bits are protected by some error correction code. Simulations show that the unary part forms about 35% of the code. Similarly, the run mode must be protected (or canceled), if synchronization is to be retained.

A change in a bit in the binary part causes JPEG-LS to fail. This, however, can be avoided by modification to the algorithm, and without protecting these bits.

First, it turns out that the context parameter k (the Golomb-Rice parameter) usually converges after 4% of the rows are encoded. Therefore, to mitigate unnecessary risks, no more updates of k are allowed after these rows are transmitted (in a protected mode). Freezing k this way seems to have very little effect on compression ratio.

Next, since the predictor of JPEG-LS is local, and easily propagates errors in the decoded image, we alter it. Notice that the gradients $\{e-d, d-b, b-x\}$ also define a context. Since e, d and b are known, selecting a context also implies a range of values for x. In order to predict x, we use our context database to choose the context $\{e-d, d-b, b-x\}$ that occurred most in the past. This (empirical) *maximum likelihood* (ML) predictor does not propagate errors as easily as the original predictor does. Since a context is defined by quantized gradients, there is a set of pixel values x that correspond to the same context. Empirical results suggest that best compression ratio is achieved when x is chosen to be the value in this set that is closest to the value of pixel b. Table 1 presents the effect on compression ratio for different images when the ML predictor replaces the JPEG-LS predictor.

Image	JPEG-LS rate	Rate using ML predictor
Lena	4.25 bpp	4.32 bpp
Peppers	4.62 bpp	4.78 bpp
Aerial2	5.3 bpp	5.53 bpp

Table 1: Comparison of image rates for the ML predictor

Synchronization loss due to channel error usually occurs when a channel error causes a pixel to be decoded using a wrong context having a parameter k different than the one used by the encoder. To reduce the probability of a "context switch", we modify the quantization of the gradients. JPEG-LS uses a non-uniform quantization that has smaller qunatization regions for smaller gradient absolute values. This allows maximization of the mutual information between the pixel and its gradients values [3]. We suggest to modify this criterion and the inferred quantization regions in favor of minimum probability of "context switch" criterion.

Therefore, our algorithm uses a uniform quantization of the gradients. The reduction in compression ratio caused by this modification usually does not exceed 7%. Actually, using a single context seems to degrade performance very little, and has the advantage that loss of synchronization is impossible.

How are the quantization regions set? Because the error is bounded, there exists a bounded set of gradient values that might be shifted to an adjacent quantization region in case of an error. These values lie around both edges of the region. The quantization regions are therefore scaled to contain enough values that cannot be shifted to an adjacent region even in case of an error. Assume a context with a parameter k, then channel errors may effect the decoded gradient by a value in the range $\left\{-2^k,...,2^k-1\right\}$, whereas an error of one bit can cause an error value in $\left\{-2^{k-1},-2^{k-2}...,2^{k-1}\right\}$. Note that changing the least significant bit usually causes the worst effect. This bit holds the sign of the residual value. Hence, we protect this bit as well. Note that a single bit change causes now only value changes in $\left\{-2^{k-2},-2^{k-3},...,2^{k-2}\right\}$. The protected bits form now about 45% of the code.

The last suggested modification relates to the structure of the context. Using the ML predictor together with the JPEG-LS contexts, a propagation of a channel error is feasible in three directions, and a "context switch" is possible in the shaded locations depicted in figure 2. As the number of "context switches" increases, synchronization loss is more probable. Therefore, the context is changed to be the quantized gradients $\{h-g, g-f, f-b\}$ instead of $\{d-b, b-c, c-a\}$. This modification causes the error to propagate usually in the same column only, and the probability of synchronization loss is dramatically reduced, as also is the visual effect on the entire image. It should be noted that this modification has the largest impact on the compression ratio (about 10% decrease), since this context does not surround the pixel x, and does not capture well its neighborhood's activity.

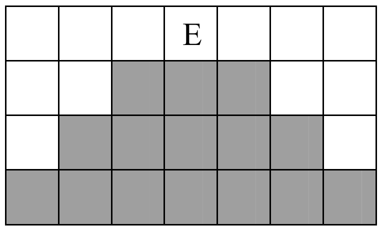


Figure 2: Propagation of the error. The "E" symbolizes the location of the error.

Demonstration of results

The performance of the modified algorithm is compared to that of the original JPEG-LS in table 2. The rate measured, for both algorithms, does not take into account the syntax structure required by the standard [1] as well as the protection bits required for the protection of the unary part of the code of the modified algorithm.

Image	Rate of the modified algorithm	Increase in rate compared to JPEG-LS
Lena	4.96 bpp	14.4 %
Peppers	5.3 bpp	13 %
Aeriel2	5.88 bpp	10 %

Table 2: Rate results of the modified algorithm, compared to the original JPEG-LS

Figure 3 presents the effect of a single error, in the binary part of the code, on the JPEG-LS algorithm. Figure 4 presents the response of the modified algorithm to the same error.

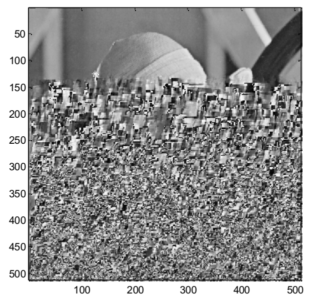


Figure 3: Response of JPEG-LS

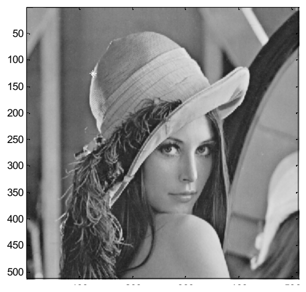
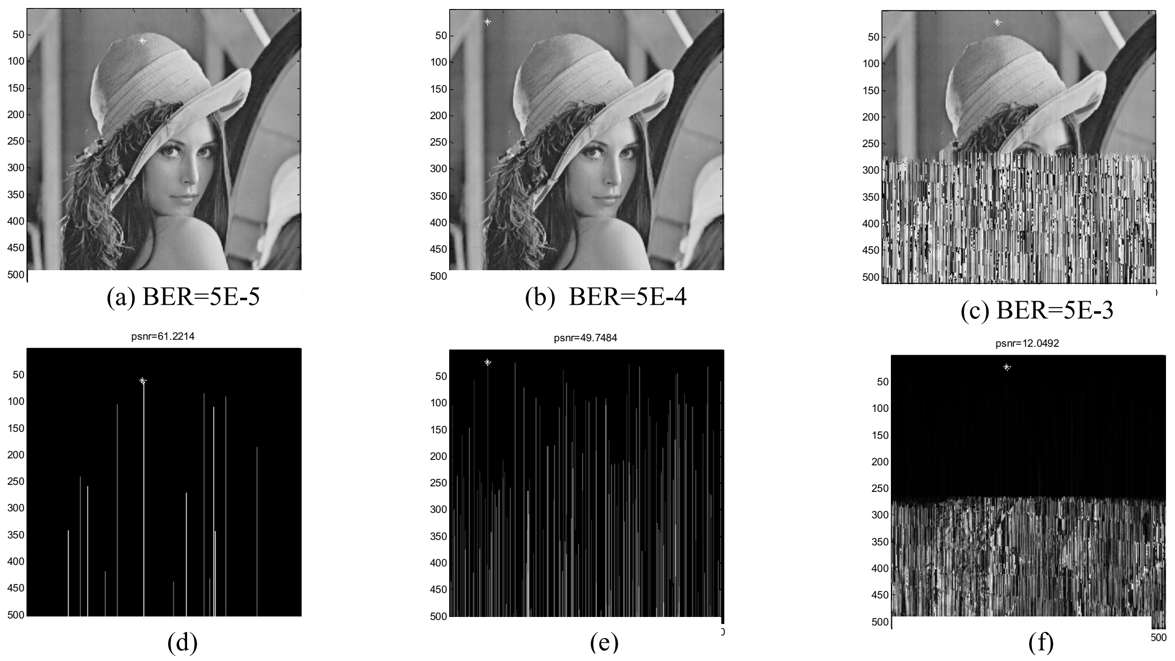


Figure 4: Response of the modified algorithm

Figures 5a-5c show the effect of the bit error rate, in the binary part only, on the reconstructed image. Figures 5d-5f show the reconstruction errors images that correspond to figures 5a-5c respectively.



(a) BER=5E-5 (b) BER=5E-4 (c) BER=5E-3

(d) (e) (f)

Figure 5: Effect of the bit error rate on the reconstructed image,
(a)-(c) The reconstructed image at different BERs: 5e-5 ÷ 5e-3
(d)-(f) The reconstruction errors

References

[1] ISO/IEC JTC1/SC29 WG1 ITU-T SG8 (JPEG/JBIG), CD 14495 “Lossless and near-lossless coding of continuous tone still images (JPEG-LS)”, 1998

[2] M. Weinberger *et al*, “LOCO-I: A Low Complexity, Context-Based Lossless Image Compression Algorithm”, In *1996 Data Compression Conference*, pp. 140-149, Snowbird, Utah, USA, March 1996.

[3] M. Weinberger *et al*, “The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS”, Hewlett-Packard internal report, HPL-98-193, November 1998.