From LOCO-I to the JPEG-LS Standard

Marcelo J. Weinberger, Gadiel Seroussi
Computer Systems Laboratory
HP Laboratories Palo Alto
HPL-1999-3
January, 1999

E-mail: [marcelo,seroussi]@hpl.hp.com

lossless image compression, standards

LOCO-I (LOw COmplexity LOssless COmpression for Images) is the algorithm at the core of the new ISO/ITU standard for lossless and near-lossless compression of continuous-tone images, JPEG-LS. The algorithm, presented at DCC'96, was conceived as a "low complexity projection" of the universal context modeling paradigm, matching its modeling unit to a simple coding unit based on Golomb codes. The JPEG-LS standard evolved after successive refinements of the core algorithm. This evolution, and the main algorithmic components of JPEG-LS are described in this paper. JPEG-LS attains compression ratios similar or superior to those obtained with state-of-the-art schemes based on arithmetic coding. Moreover, it is within a few percentage points of the best available compression ratios, at a complexity level estimated at an order of magnitude lower.

Internal Accession Date Only

1 Introduction

LOCO-I (*LOw COmplexity LOssless COmpression for Images*) is the algorithm at the core of the new ISO/ITU standard for lossless and near-lossless compression of continuous-tone images, JPEG-LS [1]. The algorithm was introduced in [2], and the standard evolved after successive refinements [3, 4, 5, 6, 7]. In this paper, we present a full description of the main algorithmic components of JPEG-LS.

Lossless data compression schemes often consist of two distinct and independent components: *modeling* and *coding*. The conceptual separation between these components [8] was made possible by the invention of the *arithmetic codes* [9], which can realize any probability assignment dictated by the model, to a preset precision. These two milestones in the development of lossless data compression allowed researchers to view the problem merely as one of probability assignment, concentrating on the design of imaginative models for specific applications (e.g., image compression) with the goal of improving on compression ratios. Optimization of the sequential probability assignment process for images, inspired on the ideas of *universal modeling*, is analyzed in [10], where a relatively high complexity scheme is presented as a way to demonstrate these ideas. Rather than pursuing this optimization, the main objective driving the design of LOCO-I was to systematically "project" the image modeling principles outlined in [10] and further developed in [11], into a low complexity plane, both from a modeling and coding perspective. A key challenge in this process is that the above separation between modeling and coding becomes less clean under the low complexity coding constraint. This is because the use of a generic arithmetic coder, which enables the most general models, is ruled out in many low complexity applications, especially for software implementations.

While [10] represented the best published compression results at the time (at the cost of high complexity), it could be argued that the improvement over the state of the art (e.g., variants of the Sunset algorithm [12, 13]), was scant. The research leading to the CALIC algorithm [14], conducted in parallel to the development of LOCO-I, seems to confirm a pattern of diminishing returns. CALIC avoids some of the optimizations performed in [10], but by tuning the model more carefully to the image compression application, some compression gains are obtained. Yet, the improvement is not dramatic, even for the most complex version of the algorithm [15]. More recently, the same observation applies to the TMW algorithm [16]. Actually, in many applications, a drastic complexity reduction can have more practical impact than a modest increase in compression. This

observation suggested that judicious modeling, which seemed to be reaching a point of diminishing returns in terms of compression ratios, should rather be applied to obtain competitive compression at significantly lower complexity levels. Thus, the challenge for LOCO-I was to bridge the (significant) compression gap between simplicity-driven schemes (e.g., the Huffman mode of the lossless JPEG standard [17]), and the more complex ones.

The FELICS algorithm [18] can be considered as a first step in bridging the above gap. While maintaining the complexity level of FELICS, LOCO-I attains significantly better compression ratios, similar or superior to those obtained with state-of-the art schemes based on arithmetic coding, but at a fraction of the complexity. In fact, as shown in [2, 19], LOCO-I performs within a few percentage points of the best available compression ratios (given by CALIC), at a complexity level estimated at about an order of magnitude lower.

The remainder of this paper is organized as follows. Section 2 presents a detailed description of the basic algorithm behind JPEG-LS, culminating with a summary of all the steps of the algorithm. While the modeling considerations discussed in [2] are omitted (the reader is referred to [19] for an in depth discussion), some basic elements of LOCO-I are revisited for the sake of completeness. A more detailed description is reserved to JPEG-LS components that are new or are not discussed in detail in [2]. A lossy mode of operation, termed "near-lossless," is discussed in Section 3. Section 4 lists various features in the standard, including the treatment of multi-component (color) images. Section 5 presents a variant of LOCO-I, based on arithmetic coding, adopted for a prospective extension of the baseline JPEG-LS standard. In Section 6, compression results are reported, including measured data throughput for a software implementation.

2 Detailed description of JPEG-LS

The modeling in LOCO-I/JPEG-LS follows the structure pioneered by the Sunset algorithm [12], including:

a. *Prediction* of a value $\hat{x}_{t+1}$ for the next sample x_{t+1} based on a finite subset (a *causal template*) of the past data $x^t = x_1 x_2 \cdots x_t$ (in the sequel, time indexes reflect a raster-scan order).

b. Determination of a *context* in which x_{t+1} occurs (the context is a function of a causal template).

c. Choice of a probabilistic model for the *prediction residual* $\epsilon_{t+1} \stackrel{\Delta}{=} x_{t+1} - \hat{x}_{t+1}$, conditioned on the context of x_{t+1}.

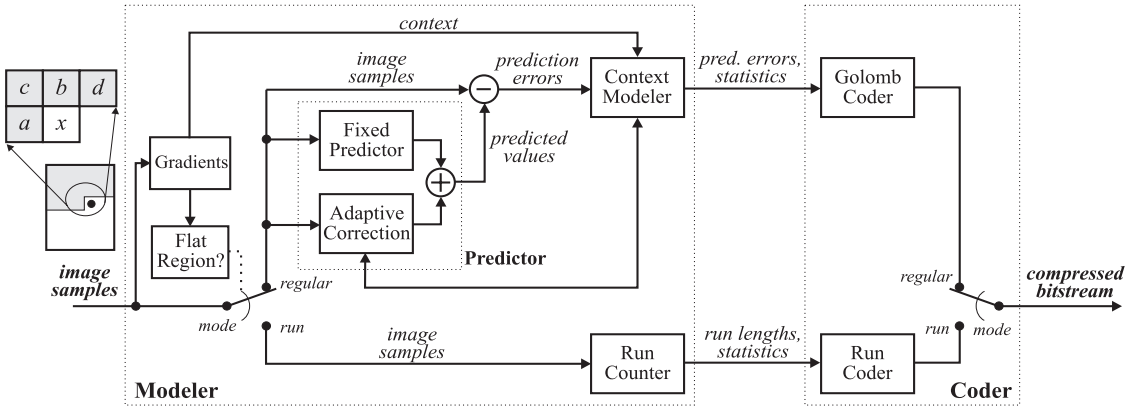

Figure 1: JPEG-LS: Block Diagram

The overall simplicity of LOCO-I/JPEG-LS can be mainly attributed to its success in matching the complexity of the modeling and coding units, combining simplicity with the compression potential of context models, thus "enjoying the best of both worlds." The main blocks of the algorithm are depicted in Figure 1. The shaded area in the image icon at the left of the figure represents the scanned past sequence x^t, on which prediction and context modeling are based, while the black dot represents the sample currently encoded. The switches labeled *mode* select operation in "regular" or "run" mode, as determined from x^t by a simple region "flatness" criterion (see Section 2.7). The causal template for JPEG-LS is also depicted in Figure 1, where x denotes the current sample, and a, b, c, and d, are neighboring samples (both the value and the location) in the relative positions shown in the figure,[1] where the dependence on the time index t is omitted. By using the template of Figure 1, JPEG-LS limits its image buffering requirement to one scan line.

2.1 Predictor

In LOCO-I/JPEG-LS, the predictor consists of a fixed and an adaptive component. The fixed component incorporates prior knowledge on the structure of images to be compressed (thus saving unnecessary "model learning" costs [20, 11]) through a rudimentary edge detecting capability. Specifically, it guesses:

$$\hat{x}_{\mathrm{MED}} \stackrel{\Delta}{=} \begin{cases} \min(a,\, b) & \text{if } c \geq \max(a,\, b) \\ \max(a,\, b) & \text{if } c \leq \min(a,\, b) \\ a + b - c & \text{otherwise.} \end{cases} \tag{1}$$

Combining interpretations in [2] and [21], the predictor (1) was termed "median edge detector" (MED) during the standardization process.

[1] The causal template in [2] includes an additional sample, e, West of a. This location was discarded in the course of the standardization process as its contribution did not justify the additional context storage required [5].

The adaptive component is limited to an *integer* additive term, analogous to an affine term in the adaptive predictor of [10]. It effects a context-dependent translation ("bias cancellation"), and can be interpreted as part of the estimation procedure for the probabilistic model for the prediction residuals (see Section 2.4).

2.2 Parameterization

Reducing the number of parameters is a key objective in a context modeling scheme, to avoid "context dilution." The total number of parameters in the model depends on the number of contexts and on the number of free parameters defining the coding distribution at each context. The latter is reduced to two in LOCO-I/JPEG-LS by assuming a *two-sided geometric distribution* (TSGD) model for the prediction residuals. According to this parametric class, the marginal probability mass function (PMF) of a prediction residual, conditioned on the context, is given by

$$P_{(\theta,\rho)}(\epsilon) = C(\theta,\rho)\theta^{|\epsilon+\rho|}, \qquad \epsilon = 0, \pm 1, \pm 2, ... \tag{2}$$

Here, $C(\theta,\rho)$ is a normalization factor, $\theta \in (0,1)$ controls the two-sided exponential decay rate, and $0 \le \rho < 1$ is a *shift parameter* which reflects a DC offset typically present in the prediction error signal of context-based schemes. This offset is due to integer-value constraints and possible bias in the prediction step. The shift parameter is also useful for better capturing the two adjacent modes often observed in empirical context-dependent histograms of prediction errors. This is achieved by letting ρ take non-integer values. Moreover, the limited range of ρ, which corresponds to PMF modes at 0 and -1, assumes that the adaptive term of the predictor is tuned to produce average residuals between 0 and -1 (see Section 2.4). The choice of this interval is matched to the codes presented in Section 2.5. Therefore, the fixed prediction offset is broken into an integer part (or "bias"), canceled by the adaptive part of the predictor, and a (negative) fractional part (or shift). The TSGD centered at zero corresponds to $\rho = 0$, and, when $\rho = \frac{1}{2}$, $P_{(\theta,\rho)}$ is a bi-modal distribution with equal peaks at -1 and 0. Notice that the parameters of the PMF are context-dependent, even though this fact is omitted in our notation. Coding of TSGDs is extensively studied in [22].

The image alphabet, assumed infinite in (2), has a finite size α in practice. For a given prediction $\hat{x}$, ϵ takes on values in the range $-\hat{x} \le \epsilon < \alpha - \hat{x}$. Since $\hat{x}$ is known to the decoder, the actual value of the prediction residual can be reduced, modulo α, to a value between $-\lfloor \alpha/2 \rfloor$ and $\lceil \alpha/2 \rceil - 1$. This is done in LOCO-I/JPEG-LS, thus remapping large prediction residuals to small ones. Merging the "tails" of peaked distributions with their central part does not significantly affect the two-sided geometric behavior. Since α is typically quite large, the analysis is still based on the infinite

alphabet model (2).

2.3 Context model

The context that conditions the encoding of the prediction residual in JPEG-LS is built out of local gradients [10, 2]. Specifically, the differences $g_1 = d - b$, $g_2 = b - c$, and $g_3 = c - a$ are quantized into up to 9 connected regions by a quantizer $\kappa(\cdot)$. To preserve symmetry, the regions are indexed $-4, \cdots, -1, 0, 1, \cdots, 4$, with $\kappa(g) = -\kappa(-g)$, for a total of 729 different quantized context triplets. For a prediction residual ϵ_{t+1}, if the first non-zero element of a triplet $C_t = [q_1, q_2, q_3]$, where $q_j = \kappa(g_j)$, j=1, 2, 3, is negative, the encoded value is $-\epsilon_{t+1}$, using context $-C_t$. This is anticipated by the decoder, which flips the sign if necessary to obtain the original error value. Merging contexts of "opposite signs" results in a total of 365 contexts. For an 8-bit per pixel alphabet, the default quantization regions are $\{0\}$, $\pm\{1, 2\}$, $\pm\{3, 4, 5, 6\}$, $\pm\{7, 8, \cdots, 20\}$, $\pm\{\, e \,|\, e \geq 21\,\}$. However, the boundaries are adjustable parameters, except that the central region must be $\{0\}$. In particular, a suitable choice can collapse quantization regions, resulting in a smaller effective number of contexts, with applications to the compression of small images. For medium-sized to large images, in turn, more contexts could be afforded without incurring an excessive model cost. However, the compression improvement is marginal and does not justify the increase in resources [5]. Through appropriate scaling, default boundary values are also provided for general alphabet sizes α [4, 1].

2.4 Bias cancellation

As mentioned in Section 2.1, the adaptive part of the predictor is context-based and it is used to "cancel" the integer part of the fixed prediction offset. As a result, the distributions in (2) present a context-dependent shift ρ, $0 \leq \rho < 1$. In principle, bias cancellation could be performed by keeping, for each context, a count N of context occurrences, and a cumulative sum D of prediction errors incurred so far in the context by the fixed predictor (1). Then, a *correction value* C' could be computed as the rounded average $C' = \lceil D/N \rceil$ and added to the fixed prediction $\hat{x}_{\text{MED}}$, to offset the prediction bias. This approach, however, has two main problems. First, it requires a general division, in opposition to the low complexity requirements of LOCO-I/JPEG-LS. Second, it is very sensitive to the influence of "outliers:" atypical large errors can affect future values of C' until it returns to its typical value, which is quite stable. In LOCO-I/JPEG-LS, the two problems are solved by approximating the quotient C' with a stored correction value C, which is initialized to zero and adjusted by at most one unit per occurrence of the context. An accumulation of

corrected prediction residuals, B, is also kept, with initial value zero, and is "forced" to remain in the interval $(-N, 0]$. The two variables, a set of which is kept per context, are updated according to the division-free procedure shown in Figure 2 (in C-language style). This procedure will tend to

```
B = B + ε;     /* accumulate prediction residual */
N = N + 1;     /* update occurrence counter */
/* update correction value and shift statistics */
if ( B ≤ −N ) {
    C = C − 1; B = B + N;
    if ( B ≤ −N ) B = −N + 1;
}
else if ( B > 0 ) {
    C = C + 1; B = B − N;
    if ( B > 0 ) B = 0;
}
```

Figure 2: Bias computation procedure

produce average prediction residuals in the interval $(-1, 0]$. To reduce storage requirements, C is not incremented (resp. decremented) over 127 (resp. under -128). Notice that $-B/N$ approximates the shift parameter ρ.

2.5 Coding

In a low complexity framework, the choice of a TSGD model is of paramount importance, since it leads to the surprisingly simple coding unit used in LOCO-I/JPEG-LS. The recent characterization of the family of optimal prefix codes for TSGDs [22] provides insight into the coding procedure, which is derived by reduction of the above family using techniques presented in [23]. As a result, *adaptive* symbol-by-symbol coding is possible at very low complexity, thus avoiding the use of the more complex arithmetic coders. The codes of [22] are based on *Golomb codes* [24], whose structure enables simple calculation of the code words without recourse to the storage of code tables, as would be the case with unstructured, generic Huffman codes.[2]

[2]The use of Golomb codes in conjunction with context modeling was pioneered in the FELICS algorithm [18].

2.5.1 Golomb codes and optimal prefix codes for the TSGD

Given a positive integer parameter m, the mth order Golomb code G_m encodes an integer $y \geq 0$ in two parts: a *unary* representation of $\lfloor y/m \rfloor$, and a *modified binary* representation of $y \bmod m$. Golomb codes are optimal [25] for *one-sided geometric distributions* (OSGDs) of the nonnegative integers, i.e. distributions of the form $(1-\theta)\theta^y$, where $0 < \theta < 1$. The special case $m = 2^k$ leads to particularly simple encoding/decoding procedures. We refer to codes G_{2^k} as *Golomb-power-of-2* (GPO2) codes, and to k also as a *Golomb parameter*, the distinction from 2^k being clear from the context.

In [22], the TSGD parameter space (θ, ρ) is partitioned, and a different optimal prefix code corresponds to each class in the partition ($\rho \leq \frac{1}{2}$ is assumed, since the case $\rho > \frac{1}{2}$ can be reduced to the former by means of the reflection/shift transformation $\epsilon \rightarrow -(\epsilon + 1)$). Classes are associated with one of the following three one-to-one mappings onto the nonnegative integers. One mapping is applied to the integer ϵ to be encoded, and is followed by a Golomb code; the other two are applied to $|\epsilon|$, followed by a Golomb code and a sign bit whenever $\epsilon \neq 0$. For each type of code, the classes are also indexed with a positive integer ℓ, given by a many-to-one function of θ and ρ. The parameter of the Golomb code used follows from the corresponding value of ℓ. The first mapping related to the optimal codes in [22] is given by

$$M(\epsilon) = 2|\epsilon| - u(\epsilon), \tag{3}$$

where $u(\epsilon) = 1$ if $\epsilon < 0$, or 0 otherwise. This mapping gives the index of an integer in the interleaved sequence $0, -1, 1, -2, 2, \ldots$ It was first used by Rice in [26] to encode TSGDs centered at zero, by applying a GPO2 code to $M(\epsilon)$. In case $\rho \leq 1/2$, $M(\epsilon)$ orders prediction residuals by probability. The corresponding mapping for $\rho > \frac{1}{2}$ is $M'(\epsilon) = M(-\epsilon-1)$. For $-\lfloor \alpha/2 \rfloor \leq \epsilon \leq \lceil \alpha/2 \rceil - 1$, the values $M(\epsilon)$ are in the range $0 \leq M(\epsilon) \leq \alpha - 1$. This is also the range for $M'(\epsilon)$ with even α. For odd α, $M'(\epsilon)$ can also take on the value α (but not $\alpha - 1$).

2.5.2 Sequential parameter estimation

The structured family of codes characterized in [22], provides a reasonable alternative for low complexity adaptive coding of TSGDs: Based on the past sequence of prediction residuals $\epsilon^t = \epsilon_1 \epsilon_2 \cdots \epsilon_t$ encoded at a given context, select a type of code and a value of the code index ℓ sequentially, and use this code to encode ϵ_{t+1}. (Notice that t here indexes occurrences *of a given context*, thus corresponding to the variable N introduced in Section 2.4.) The decoder makes the same determination, after decoding the same past sequence. Thus, the coding is done "on the fly," as with adaptive

arithmetic coders. Such strategy aims at an expected code length that approaches the one that would be obtained with the best *fixed* code in the family, i.e. the Huffman code for the (unknown) parameter values.

Yet, the complexity of both the code determination and the encoding procedure would be beyond the constraints set for JPEG-LS. For that reason, as in [26], LOCO-I/JPEG-LS only uses the sub-family of GPO2 codes. Furthermore, only codes based on the mappings $M(\cdot)$ and $M'(\cdot)$ are used. We denote $\Gamma_k(\epsilon) = G_{2^k}(M(\epsilon))$. The mapping $M'(\cdot)$ is relevant only for $k = 0$, since $\Gamma_k(\epsilon) = \Gamma_k(-\epsilon-1)$ for every $k > 0$. Thus, the sequential code selection task in LOCO-I/JPEG-LS consists of the selection of a Golomb parameter k, and in case $k = 0$, a mapping $M(\cdot)$ or $M'(\cdot)$. We further denote $\Gamma'_0(\epsilon) = G_1(M'(\epsilon))$.

In JPEG-LS, this code selection task is accomplished by approximating a sequential *explicit minimization* of the expected code length for a maximum likelihood estimate of the TSGD parameters θ and ρ given ϵ^t. The minimization, which performs essentially as well as the the best fixed code from the sub-family, is analyzed in [22], and approximated in [23]; see also [19, Theorem 1]. It is based on the sufficient statistics S_t and N_t for θ and ρ,

$$S_t = \sum_{i=1}^{t} (|\epsilon_i| - u(\epsilon_i)) \,, \qquad N_t = \sum_{i=1}^{t} u(\epsilon_i) \,.$$

N_t is the total number of negative samples in ϵ^t, and $S_t + N_t$ is the accumulated sum of absolute values.

In JPEG-LS, the explicit minimization is approximated as follows: For each context, the accumulated sum of magnitudes of prediction residuals, S_t+N_t, is maintained in a register A, in addition to the variables B and N defined in Section 2.4. The following procedure then selects a code for the prediction residual ϵ_{t+1}:

a. Compute k as
$$k = \min\{k' \,|\, 2^{k'} N \geq A\}. \tag{4}$$

b. If $k > 0$, choose code Γ_k. If $k = 0$ and $2B > -N$, choose code Γ_0. Otherwise, choose code Γ'_0.

The above procedure improves on the one described in [2] by its more accurate approximation of the optimal selection in the case $k = 0$. In software, k can be computed by the C programming language "one-liner"

```
for ( k=0; (N<<k)<A; k++ );
```

2.5.3 Limited-length Golomb codes

The encoding procedure of Section 2.5 can produce significant expansion for single samples: for example, with $\alpha = 256$ and $k = 0$, a prediction residual $\epsilon = -128$ would produce 256 output bits, a 32:1 expansion. Moreover, one can create artificial images with long sequences of different contexts, such that this situation occurs sample after sample, thus causing significant local expansion (this context pattern is necessary for this situation to arise, as large residuals would otherwise trigger adaptation of k to a larger value for a particular context). While such worst case sequence is not likely in practical situations, significant local expansion was observed for some images. This can be problematic in implementations with limited buffer space for compressed data. Therefore, adopting a practical trade-off, GPO2 codes are modified in JPEG-LS to limit the code length per sample to (typically) 4β, where $\beta \stackrel{\Delta}{=} \lceil \log \alpha \rceil$ (hereafter, logarithms are taken to the base 2), e.g. 32 bits for 8 bits/pixel images. The limitation is achieved using a simple technique [7] that avoids a negative impact on complexity. For a maximum code word length of $L_{\max}$ bits, the encoding of an integer y, $0 \leq y \leq \alpha$,[3] proceeds as in Section 2.5.1 whenever $q(y) = \lfloor 2^{-k} y \rfloor$ satisfies

$$q(y) < L_{\max} - \beta - 1 \stackrel{\Delta}{=} q_{\max}\,, \tag{5}$$

where we assume $L_{\max} > \beta + 1$. This is by far the most frequent case, adding only one comparison per encoding. By (4), $k \leq \beta - 1$, so that the total code length after appending k bits is within the required limit $L_{\max}$. Now, if $q(y) \geq q_{\max}$, $q_{\max}$ is encoded in unary, which acts as an "escape" code, followed by an explicit binary representation of $y - 1$, using β bits.

2.6 Resets

To enhance adaptation to non-stationarity of image statistics, and to limit the storage requirements per context, LOCO-I/JPEG-LS periodically resets the variables N, A, and B. Specifically, in each context these variables are *halved* (rounding down to nearest integer) each time N attains a predetermined threshold N_0. In this way, the immediate past is given a larger weight than the remote past. Values of N_0 between 32 and 256 work well for typical images; the default value in JPEG-LS is 64.

2.7 Embedded alphabet extension

Symbol-by-symbol (Huffman) coding (as opposed to arithmetic coding) is inefficient for very low entropy distributions, due to its fundamental limitation of producing at least one code bit per encoding. This problem is addressed in LOCO-I by embedding an *alphabet extension* in the model

[3] Notice that the range for y may include α in case α is odd and the mapping $M'(\cdot)$ is used.

("run" mode) for "flat regions." The encoder enters a "run" mode when a "flat region" context with $a = b = c = d$ is detected. A run of the sample a is expected, and the run length is encoded. When the run is broken by a non-matching sample x, the encoder goes into a "run interruption" state, where the difference $\epsilon = x - b$ (with the sample above x) is encoded. Runs can also be broken by ends of lines, in which case the encoder returns to normal context-based coding. Since all the decisions for switching in and out of the run mode are based on past samples, the decoder can reproduce the same decisions without any side information.

The encoding of run lengths in JPEG-LS is also based on Golomb codes, originally proposed in [24] for such applications. However, an improved adaptation strategy can be derived from viewing the encoded sequence as binary ('0' for a "hit," '1' for a "miss"). For a positive integer parameter m, let EG_m denote a variable-to-variable length code defined over the extended binary alphabet $\{1, 01, 001, \ldots, 0^{m-1}1, 0^m\}$, where 0^ℓ denotes a sequence of ℓ zeros. Under EG_m, the extended symbol 0^m (a successful run of m "hits") is encoded with a 0, while $0^\ell 1$, $0 \leq \ell < m$, is encoded with a 1 followed by the modified binary representation of ℓ. By considering a concatenation of extended input symbols, it is easy to see that EG_m is equivalent to G_m applied to the run length. However, EG_m is defined over a finite alphabet, with "hits" and "misses" modeled as independent and identically distributed (i.i.d.). We will refer to EG_m as an *elementary Golomb code* of order m. Variations of these codes were introduced in [27, 28, 29]. They are also studied in [30] in the context of embedded coding of wavelet coefficients, where insight is provided into their well-known efficiency for encoding i.i.d. binary sequences over a surprisingly wide range of values of the probability q of a '0.'

When q is unknown *a priori*, EG_m is superior to G_m in that m can be adapted *within a run*, based on the current estimate of q. The optimal adaptation turns out to be extremely simple if the the family of codes is reduced, again, to the case where $m = 2^g$, while the redundancy remains very small for the ranges of interest. This approach, termed *block-MELCODE*, originates in [28, 3] and was adopted in JPEG-LS. Count-based adaptation strategies for the parameter g are proposed in [28] and [30]. JPEG-LS uses a pre-defined table to approximate these strategies. A run segment of length m (i.e., 0^m) triggers an index increment, while a "miss" (i.e., $0^\ell 1$, $0 \leq \ell < m$) triggers an index decrement. The index is used to enter the table, which determines how many consecutive run segments (resp. "misses") trigger an increment (resp. decrement) of g (see [1]).

Main Algorithm

Step 0. Initialization:

a. Compute $L_{\max} = 2(\beta_{\max} + \max\{8, \beta_{\max}\})$, where $\beta_{\max} = \max\{2, \lceil \log \alpha \rceil\}$.

b. Initialize the 365 sets of context counters A, B, C, and N as follows: $B = C = 0$, $N = 1$, $A = \max\{2, \lfloor (\alpha + 32)/64 \rfloor\}$. Similar initializations are needed for the corresponding variables in the two run interruption contexts.

c. Initialize to 0 the index I_{RUN} to the run mode adaptation table.

d. Set current sample x to the first sample in the image.

Step 1. Compute the "local gradients" $g_1 = d - b$, $g_2 = b - c$, and $g_3 = c - b$.

Step 2. If $g_1{=}g_2{=}g_3{=}0$, go to **Run Mode Processing**. Otherwise, continue in "regular" mode.

Step 3. Quantize the local gradients g_i, $i = 1, 2, 3$ as described in Section 2.3.

Step 4. Denote the quantized gradients by q_i, $i = 1, 2, 3$. If the first non-zero component of $[q_1, q_2, q_3]$ is negative, reverse all the signs in the triplet and associate a negative sign to it. Otherwise, associate a positive sign. Map the triplet, on a one-to-one basis, into an index in the range $[1, 364]$ (context number 0 is reserved to the run mode). Use the index to address the context counters.

Step 5. Compute the fixed prediction $\hat{x}_{\text{MED}}$ according to (1).

Step 6. Correct $\hat{x}_{\text{MED}}$ by adding (resp. subtracting) the value of C for the context in case the sign associated to the context is positive (resp. negative). Clamp the corrected value to the range $[0, \alpha - 1]$ to obtain the corrected prediction $\hat{x}$.

Step 7. Compute the prediction residual $\bar{\epsilon} = x - \hat{x}$ and, if a negative sign is associated to the context, set $\bar{\epsilon} \leftarrow -\bar{\epsilon}$. Reduce $\bar{\epsilon}$ modulo α to a value ϵ in the range $[-\lfloor \alpha/2 \rfloor, \lceil \alpha/2 \rceil - 1]$.

Step 8. Compute the Golomb parameter k according to (4).

Step 9. Map ϵ to $M(\epsilon)$ or, if $k = 0$ and $2B \leq -N$, to $M'(\epsilon)$.

Step 10. Golomb-encode the mapped prediction residual using the parameter k and, if necessary, perform the code word length limitation procedure with maximum length $L_{\max}$.

Step 11. Update the context counters by adding ϵ to B and $|\epsilon|$ to A, halving A, B, and N in case $N = N_0$ (reset threshold), and incrementing N.

Step 12. Update the values of B and C following the "if-else" statement in Figure 2.

Step 13. Go to Step 1 to process the next sample.

Run Mode Processing

Step 1. Read new samples until either $x \neq a$ or the end of the line is encountered.

Step 2. Let $m = 2^g$ denote the current parameter of the elementary Golomb code. For each run segment of length m, append a '1' to the output bit stream and increment the index I_{RUN}. If so indicated by the table, double m.

Step 3. If the run was interrupted by the end of a line, append '1' to the output bit stream and go to Step 1 of the main algorithm (Fig. 3). Otherwise, append '0' to the output bit stream followed by the binary representation of the residual run length using g bits, decrement I_{RUN}, and if so indicated by the table, half m.

Step 4. Encode the run interruption sample and go to Step 1 of the **Main Algorithm**.

Figure 3: JPEG-LS: encoding of a single component.

For a run interruption sample x the coded value is $\epsilon = (x-b) \bmod \alpha$. Thus, both the fixed predictor (1) and the bias cancellation procedure are skipped. Coding is otherwise similar to the regular sample case. However, conditioning is based on two special contexts, determined according to whether $a = b$ or $a \neq b$. In the former case, we always have $\epsilon \neq 0$ (since, by definition of run interruption, $x \neq a$). Therefore, the mappings $M(\cdot)$ and $M'(\cdot)$ are modified to take advantage of this exclusion. Moreover, since the B counter is not used in these contexts (no bias cancellation is performed), the decision between $M(\cdot)$ and $M'(\cdot)$ is based on the number N_t of negative values occurring in each context. The same reset procedure as in Section 2.6 is used for the corresponding counters. Also, the length limitation for the Golomb code takes into account the $g + 1$ bits of the last coded run segment, thus limiting every code word length to $L_{\max}-g-1$ bits.

2.8 Summary of encoding procedures

Figure 3 summarizes the lossless encoding procedures described in Section 2 for a single component of an image. The decoding process uses the same basic procedures and follows almost the same steps in reverse order (see [1] for details). Causal template values falling outside image boundaries are suitably defined in [1].

3 Near-lossless compression

JPEG-LS offers a lossy mode of operation, termed "near-lossless," in which every sample value in a reconstructed image component is guaranteed to differ from the corresponding value in the original image by up to a preset (small) amount, δ. The basic technique employed for achieving this goal is the traditional DPCM loop, where the prediction residual (after correction and possible sign reversion, but before modulo reduction) is quantized into bins of size $2\delta+1$, with reproduction at the center of the interval. Quantization of a prediction residual ϵ is performed by integer division, according to

$$Q(\epsilon) = \mathrm{sign}(\epsilon) \left\lfloor \frac{|\epsilon| + \delta}{2\delta + 1} \right\rfloor .$$

The following aspects of the coding procedure are affected by the quantization step. Context modeling and prediction are based on reconstructed values, so that the decoder can mimic the operation of the encoder. In the sequel, the notation a, b, c, and d, will be used to refer also to the reconstructed values of the samples at these positions. The condition for entering the run mode is relaxed to require that the gradients g_i, $i = 1, 2, 3$, satisfy $|g_i| \leq \delta$. This relaxed condition reflects the fact that reconstructed sample differences up to δ can be the result of quantization errors. Moreover, once in run mode, the encoder checks for runs within a tolerance of δ, while reproducing

the value of the reconstructed sample at a. Consequently, the two run interruption contexts are determined according to whether $|a - b| \leq \delta$ or not. The relaxed condition for the run mode also determines the central region for quantized gradients, which is $|g_i| \leq \delta$, $i = 1, 2, 3$. Thus, the size of the central region is increased by 2δ, and the default thresholds for gradient quantization are scaled accordingly.

The reduction of the quantized prediction residual is done modulo $\alpha' = \lfloor (\alpha + 4\delta)/(2\delta + 1) \rfloor$, into the range $[-\lfloor \alpha'/2 \rfloor, \lceil \alpha'/2 \rceil - 1]$. The reduced value is (losslessly) encoded and recovered at the decoder, which first multiplies it by $2\delta{+}1$, then adds it to the (corrected) prediction (or subtracts it, if the sign associated to the context is negative), and reduces it modulo $\alpha'(2\delta{+}1)$ into the range $[-\epsilon,\ \alpha' \cdot (2\delta{+}1) - 1 - \delta]$, finally clamping it into the range $[0, \alpha - 1]$. It can be seen that, after modular reduction, the recovered value cannot be larger than $\alpha - 1 + \delta$. Thus, before clamping, the decoder actually produces a value in the range $[-\delta, \alpha - 1 + \delta]$, which is precisely the range of possible sample values with an error tolerance of $\pm\delta$.

As for encoding, α' replaces α in the definition of the limited-length Golomb coding procedure. Since A accumulates quantized error magnitudes, $k < \lceil \log \alpha' \rceil$. On the other hand, B accumulates the encoded value, *multiplied by* $2\delta{+}1$. The alternative mapping $M'(\cdot)$ is not used.

Although the initial goal of this mode was to guarantee a bounded error for applications with legal implications (e.g., medical images), for small values of δ its visual and SNR performance is often superior to that of traditional transform coding techniques.

4 Other features of the standard

Bit stream. The compressed data format for JPEG-LS closely follows the one specified for JPEG [17]. The same high level syntax applies, with the bit stream organized into frames, scans, and restart intervals within a scan, markers specifying the various structural parts, and marker segments specifying the various parameters. New marker assignments are compatible with [17]. One difference, however, is the existence of default values for many of the JPEG-LS coding parameters (e.g., gradient quantization thresholds, reset threshold), with marker segments used to override these values. In addition, the method for easy detection of marker segments differs from the one in [17] (see [1, 19]).

Color images. For encoding color images, JPEG-LS supports combinations of single-component and multi-component scans. Section 2 describes the encoding process for a single-component scan. For multi-component scans, a single set of context counters (namely, A, B, C,

and N for regular mode contexts) is used across all components in the scan. Prediction and context determination are performed as in the single component case, and are component independent. Thus, the use of possible correlation between color planes (e.g., in an RGB representation) is limited to sharing statistics, collected from all planes. A more comprehensive exploitation of this correlation is beyond the specified scope of JPEG-LS.

In JPEG-LS, the data in a multi-component scan can be interleaved either by lines (*line-interleaved* mode) or by samples (*sample-interleaved* mode). In line-interleaved mode, the run mode adaptation is component-dependent. In sample-interleaved mode, the runs are common to all the components in the scan, with run mode selected only when the corresponding condition is satisfied for *all* the components. Likewise, a run is interrupted whenever so dictated by *any* of the components. Thus, a single run length, common to all components, is encoded. This approach is convenient for images in which runs tend to be synchronized between components (e.g., synthetic images), but should be avoided in cases where run statistics differ significantly across components, since a component may systematically cause run interruptions for another component with otherwise long runs (e.g. the run-prone K component in CMYK representation). Since one sample from each component is processed in turn, all components in a scan must have the same dimensions.

Palletized images. The JPEG-LS syntax also provides tools for encoding palletized images in index space (i.e., as an array of indices to a palette table), rather than in the original color space. To this end, the decoding process may be followed by a so-called *sample-mapping procedure*, which maps each decoded sample value (e.g., an 8-bit index) to a reconstructed sample value (e.g., an RGB triplet) by means of mapping tables. Appropriate syntax is defined to allow embedding of these tables in the JPEG-LS bit stream.

Many of the assumptions for the JPEG-LS model, targeted at continuous-tone images, do not hold when compressing an array of indices. However, an appropriate reordering of the palette table can sometimes alleviate this deficiency (e.g., the low complexity luminance-order heuristic [31]). JPEG-LS does not specify a particular palette ordering. Notice also that JPEG-LS was not designed, and might not give optimal performance, for images that have been palletized through dithering.

Sample-mapping is also useful for images with "sparse histograms." Such images contain only a subset of the possible sample values, and the fixed predictor (1) would tend to concentrate the

value of the prediction residuals into a reduced set. However, prediction correction tends to spread these values over the entire range, and even if that were not the case, the probability assignment of a TSGD model would not take advantage of the reduced alphabet. Assuming prior knowledge of the histogram sparseness, this problem is addressed by mapping the sparse samples to a contiguous set.

5 LOCO-A: an arithmetic coding extension

In this section, we present an arithmetic coding extension of LOCO-I, termed LOCO-A [32], which has been adopted for a prospective extension of the baseline JPEG-LS standard (JPEG-LS Part 2). The goal of this extension is to address the basic limitations that the baseline presents when dealing with very compressible images (e.g., computer graphics, near-lossless mode with $\delta \geq 3$), due to the symbol-by-symbol coding approach, or with images that are far from being continuous-tone or have sparse histograms. In addition, LOCO-A closes, in general, most of the (small) compression gap between JPEG-LS and the best published results (see Section 6), while still being considerably simpler than these higher complexity alternatives.

LOCO-A is a natural extension of the JPEG-LS baseline, requiring the same buffering capability. The context model and most of the prediction are identical to those in LOCO-I. The basic difference follows from an alternative interpretation of the modeling approach in LOCO-I.[4] Under this interpretation, the use of only $\lceil \log \alpha \rceil + 1$ different prefix codes to encode context-dependent distributions of prediction residuals, is viewed as a (dynamic) way of clustering conditioning contexts. The clusters result from the use of a small family of codes, as opposed to a scheme based on arithmetic coding, which would use different arithmetic codes for different distributions. Thus, this aspect of LOCO-I can also be viewed as a realization of the basic paradigm proposed and analyzed in [11] and also used in CALIC [14], in which a multiplicity of predicting contexts is clustered into a few conditioning states. This interpretation suggests that *conditioning states* can be obtained by clustering contexts based on the value of the Golomb parameter k (thus grouping contexts with similar conditional distributions). The resulting state-conditioned distributions can be arithmetic encoded, therefore relaxing the TSGD assumption, which is used only as a means to form the states. The relaxation of the TSGD assumption is possible due to the small number of states, $\lceil \log \alpha \rceil + 1$, which enables the modeling of more parameters per state. In LOCO-A, this idea is generalized to create higher resolution clusters based on the average magnitude A/N of prediction residuals (as

[4] Xiaolin Wu, private communication.

k is itself a function of A/N). Since, by definition, each cluster would include contexts with very similar conditional distributions, this measure of activity level can be seen as a refinement of the "error energy" used in CALIC [14], and a further application of the paradigm of [11]. Activity levels are also used in the ALCM algorithm [33].

Modeling in LOCO-A proceeds as in LOCO-I, collecting the same statistics at each context (with the "run" condition defining a separate encoding state). Clustering is accomplished by modifying (4) as follows:

$$k = \min\{k' \mid 2^{k'/2} N \geq A\}.$$

For 8 bits/pixel images, 12 encoding states are defined: $k = 0$, $k = 1$, $\cdots$, $k = 9$, $k > 9$, and the run state.

Bias cancellation is performed as in LOCO-I, except that the TSGDs are centered in the interval $(-1/2, 1/2]$, instead of $(-1, 0]$ (as the coding method that justified the negative fractional shift in LOCO-I is no longer used). In addition, regardless of the computed correction value, the corrected prediction is incremented or decremented in the direction of $\hat{x}_{\text{MED}}$ until it is either a value that has already occurred in the image, or $\hat{x}_{\text{MED}}$. This modification alleviates the unwanted effects on images with sparse histograms, while having virtually no effect on "regular" images. No bias cancellation is done in the run state. A "sign flip" borrowed from the CALIC algorithm [14] is performed: if the bias count B is positive, then the sign of the error is flipped. In this way, when distributions that are similar in shape but have opposite biases are merged, the statistics are added "in phase." Finally, prediction errors are arithmetic-coded conditioned on one of the 12 encoding states. Binary arithmetic coding is performed, following the Golomb-based binarization strategy of [33]. For a state with index k, we choose the corresponding binarization tree as the Golomb tree for the parameter $2^{\lceil k/2 \rceil}$ (the run state also uses $k = 0$).

6 Results

Extensive comparisons of the compression performance of LOCO-I/JPEG-LS with that of other relevant lossless image compression schemes are presented in [2] and [19], where it is shown that LOCO-I/JPEG-LS significantly outperforms other schemes of comparable complexity (e.g., PNG, FELICS, JPEG-Huffman), and it attains compression ratios similar or superior to those of higher complexity schemes based on arithmetic coding (e.g., Sunset CB9 [13], JPEG-Arithmetic). LOCO-I/JPEG-LS is, on the average, within a few percentage points of the best available compression ratios (given by CALIC [14]), at a complexity level estimated at about an order of magnitude lower.

Image	LOCO-I	JPEG-LS	Near-lossless ±1	Near-lossless ±3	CALIC	LOCO-A
bike	3.59	3.63	2.39	1.57	3.50	3.54
cafe	4.80	4.83	3.41	2.40	4.69	4.75
woman	4.17	4.20	2.75	1.82	4.05	4.11
tools	5.07	5.08	3.68	2.65	4.95	5.01
bike3	4.37	4.38	3.10	2.17	4.23	4.33
cats	2.59	2.61	1.84	1.28	2.51	2.54
water	1.79	1.81	1.07	0.61	1.74	1.75
finger	5.63	5.66	4.03	2.89	5.47	5.50
us	2.67	2.63	1.64	1.14	2.34	2.45
chart	1.33	1.32	0.86	0.56	1.28	1.18
chart_s	2.74	2.77	1.85	1.25	2.66	2.65
compound1	1.30	1.27	0.90	0.67	1.24	1.21
compound2	1.35	1.33	0.94	0.70	1.24	1.25
aerial2	4.01	4.11	2.87	2.03	3.83	3.58
faxballs	0.97	0.90	0.68	0.47	0.75	0.64
gold	3.92	3.91	2.46	1.52	3.83	3.85
hotel	3.78	3.80	2.36	1.45	3.71	3.72
Average	3.18	3.19	2.17	1.48	3.06	3.06

Table 1: Compression results on JPEG-LS benchmark set (in bits/pixel averaged over color planes)

The results presented in this section highlight aspects not covered in [2], namely, the (slight) change in performance between LOCO-I and JPEG-LS, compression in near-lossless mode, the performance of the LOCO-A extension, and how the latter compares to CALIC [14]. The results are presented in Table 1. The rows in the table correspond to the subset of 8 bits/pixel images from the benchmark set provided in the Call for Contributions leading to JPEG-LS. This is a rich set with a wide variety of images, including natural images, compound documents, aerial photographs, scanned and computer generated images. Images were compressed in component-by-component mode with compression ratios given in bits/pixel averaged over color planes. Except for the columns labeled "near-lossless," results are for lossless compression.

The discrepancy between the LOCO-I and JPEG-LS columns is due to the main differences between the algorithms, namely: use of a fifth context pixel in LOCO-I, limitation of Golomb code word lengths in JPEG-LS, different coding methods in run mode, and overhead in JPEG-LS due to data format (e.g., marker segments, bit stuffing, etc.). The comparison between LOCO-A and CALIC shows that the latter scheme maintains a slight advantage (1-2%) for "smooth" images, while LOCO-A shows a significant advantage for the classes of images it targets: sparse histograms ("aerial2") and computer-generated ("faxballs"). Moreover, LOCO-A performs as well as CALIC on compound documents without using a separate binary mode [14]. The average compression ratios on the benchmark set for both schemes set end up being equal.

The performance of JPEG-LS on the images of Table 1 is very similar when run in line-interleaved mode, with a maximum compression ratio deterioration of 1% on "gold" and "hotel," and a maximum improvement of 1% on "compound1." In sample-interleaved mode, however, the deterioration is generally more significant (3 to 5% in many cases), but with a 3 to 4% improvement on compound documents.

A C-language implementation of LOCO-I/JPEG-LS benchmarks, on a 300MHz Pentium® II machine, at data rates from about 1.5 MBytes/s for natural images to about 6 MBytes/s for compound documents and computer graphics images. The latter speed-up is due in great part to the frequent use of the run mode. LOCO-I/JPEG-LS decompression is about 10% slower than compression, making it a fairly symmetric system. Executables for JPEG-LS in various platforms are available at the web site <http://www.hpl.hp.com/loco/>. The organizations holding patents covering aspects of JPEG-LS have agreed to allow payment-free licensing of these patents for use in the standard.

Acknowledgments. Many thanks to H. Kajiwara, G. Langdon, D. Lee, N. Memon, F. Ono, E. Ordentlich, M. Rabbani, D. Speck, I. Ueno, X. Wu, and T. Yoshida for useful discussions.

References

[1] ISO/IEC JTC1/SC29 WG1 (JPEG/JBIG), "Information technology - Lossless and near-lossless compression of continuous-tone still images," 1998. Final Draft International Standard FDIS14495-1 (JPEG-LS). Also, ITU Recommendation T.87.

[2] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," in *Proc. DCC'96*, (Snowbird, Utah, USA), pp. 140–149, Mar. 1996.

[3] I. Ueno and F. Ono, "Proposed modification of LOCO-I for its improvement of the performance." ISO/IEC JTC1/SC29/WG1 document N297, Feb. 1996.

[4] M. J. Weinberger, G. Seroussi, and G. Sapiro, "Fine-tuning the baseline." ISO/IEC JTC1/SC29/WG1 document N341, June 1996.

[5] M. J. Weinberger, G. Seroussi, and G. Sapiro, "Effects of resets and number of contexts on the baseline." ISO/IEC JTC1/SC29/WG1 document N386, June 1996.

[6] M. J. Weinberger, G. Seroussi, and G. Sapiro, "Palettes and sample mapping in JPEG-LS." ISO/IEC JTC1/SC29/WG1 document N412, Nov. 1996.

[7] M. J. Weinberger, G. Seroussi, G. Sapiro, and E. Ordentlich, "JPEG-LS with limited-length code words." ISO/IEC JTC1/SC29/WG1 document N538, July 1997.

[8] J. Rissanen and G. G. Langdon, Jr., "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, Jan. 1981.

[9] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Jl. Res. Develop.*, vol. 20 (3), pp. 198–203, May 1976.

[10] M. J. Weinberger, J. Rissanen, and R. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Processing*, vol. 5, pp. 575–586, Apr. 1996.

[11] M. J. Weinberger and G. Seroussi, "Sequential prediction and ranking in universal context modeling and data compression," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1697–1706, Sept. 1997. Preliminary version presented at the 1994 IEEE Intern'l Symp. on Inform. Theory, Trondheim, Norway, July 1994.

[12] S. Todd, G. G. Langdon, Jr., and J. Rissanen, "Parameter reduction and context selection for compression of the gray-scale images," *IBM Jl. Res. Develop.*, vol. 29 (2), pp. 188–193, Mar. 1985.

[13] G. G. Langdon, Jr. and C. A. Haidinyak, "Experiments with lossless and virtually lossless image compression algorithms," in *Proc. SPIE*, vol. 2418, pp. 21–27, Feb. 1995.

[14] X. Wu and N. D. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45 (4), pp. 437–444, Apr. 1997.

[15] X. Wu, "Efficient lossless compression of continuous-tone images via context selection and quantization," *IEEE Trans. Image Processing*, vol. IP-6, pp. 656–664, May 1997.

[16] B. Meyer and P. Tischer, "TMW – A new method for lossless image compression," in *Proc. of the 1997 International Picture Coding Symposium (PCS97)*, (Berlin, Germany), Sept. 1997.

[17] ISO/IEC 10918-1, ITU T.81, "Digital compression and coding of continuous tone still images - Requirements and guidelines," Sept. 1993.

[18] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," in *Proc. DCC'93*, (Snowbird, Utah, USA), pp. 351–360, Mar. 1993.

[19] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," 1998. Submitted to *IEEE Trans. Image Proc.* Available as Hewlett-Packard Laboratories Technical Report.

[20] J. Rissanen, *Stochastic Complexity in Statistical Inquiry*. New Jersey, London: World Scientific, 1989.

[21] S. A. Martucci, "Reversible compression of HDTV images using median adaptive prediction and arithmetic coding," in *Proc. IEEE Intern'l Symp. on Circuits and Syst.*, pp. 1310–1313, IEEE Press, 1990.

[22] N. Merhav, G. Seroussi, and M. J. Weinberger, "Lossless compression for sources with two-sided geometric distributions," 1998. Submitted to *IEEE Trans. Inform. Theory*. Available as Technical Report No. HPL-94-111, Apr. 1998, Hewlett-Packard Laboratories.

[23] G. Seroussi and M. J. Weinberger, "On adaptive strategies for an extended family of Golomb-type codes," in *Proc. DCC'97*, (Snowbird, Utah, USA), pp. 131–140, Mar. 1997.

[24] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 399–401, 1966.

[25] R. Gallager and D. V. Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 228–230, Mar. 1975.

[26] R. F. Rice, "Some practical universal noiseless coding techniques - parts I-III," Tech. Rep. JPL-79-22, JPL-83-17, and JPL-91-3, Jet Propulsion Laboratory, Pasadena, CA, Mar. 1979, Mar. 1983, Nov. 1991.

[27] J. Teuhola, "A compression method for clustered bit-vectors," *Information Processing Letters*, vol. 7, pp. 308–311, Oct. 1978.

[28] R. Ohnishi, Y. Ueno, and F. Ono, "The efficient coding scheme for binary sources," *IECE of Japan*, vol. 60-A, pp. 1114–1121, Dec. 1977. (In Japanese).

[29] G. G. Langdon, Jr., "An adaptive run-length coding algorithm," *IBM Technical Disclosure Bulletin*, vol. 26, pp. 3783–3785, Dec. 1983.

[30] E. Ordentlich, M. J. Weinberger, and G. Seroussi, "A low complexity modeling approach for embedded coding of wavelet coefficients," in *Proc. DCC'98*, (Snowbird, Utah, USA), pp. 408–417, Mar. 1998.

[31] A. Zaccarin and B. Liu, "A novel approach for coding color quantized images," *IEEE Trans. Image Processing*, vol. IP-2, pp. 442–453, Oct. 1993.

[32] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-A: an arithmetic coding extension of LOCO-I." ISO/IEC JTC1/SC29/WG1 document N342, June 1996.

[33] D. Speck, "Activity level classification model (ALCM)." A proposal submitted in response to the Call for Contributions for ISO/IEC JTC 1.29.12, 1995.