Generalized Golomb Codes and Adaptive Coding of Wavelet-Transformed Image Subbands

A. Kiely[1] and M. Klimesh[1]

We describe a class of prefix-free codes for the nonnegative integers. We apply a family of codes in this class to the problem of runlength coding, specifically as part of an adaptive algorithm for compressing quantized subbands of wavelet-transformed images. On test images, our adaptive coding algorithm is shown to give compression effectiveness comparable to the best performance achievable by an alternate algorithm that has been previously implemented.

I. Generalized Golomb Codes

A. Introduction

Suppose we wish to use a variable-length binary code to encode integers that can take on any nonnegative value. This problem arises, for example, in runlength coding—encoding the lengths of runs of a dominant output symbol from a discrete source. Encoding cannot be accomplished by simply looking up codewords from a table because the table would be infinite, and Huffman's algorithm could not be used to construct the code anyway. Thus, one would like to use a simply described code that can be easily encoded and decoded. Although in practice there is generally a limit to the size of the integers that need to be encoded, a simple code for the nonnegative integers is often still the best choice. We now describe a class of variable-length binary codes for the nonnegative integers that contains several useful families of codes.

Each code in this class is completely specified by an index function f from the nonnegative integers onto the nonnegative integers. The index function f partitions the nonnegative integers into indexed sets S_0, S_1, $\cdots$, where S_i denotes the set of integers that map to index i:

$$S_i = \{z : f(z) = i\}$$

We require each set S_i to be finite.

The codeword for an integer z consists of two parts. The first part is the unary codeword for the index $i = f(z)$, i.e., i ones followed by a zero. The second part specifies the rank of z among the members of S_i. The rank r, given by

[1] Communications Systems and Research Section.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

$$r = |\{j : j \in S_i, j < z\}|$$

takes on the values $0, 1, \cdots, |S_i| - 1$. Let $b \triangleq \lfloor \log_2 |S_i| \rfloor$. If $r < 2^{b+1} - |S_i|$, the rank is simply encoded as the b-bit binary representation of r. Otherwise the $(b+1)$-bit binary representation of $r + 2^{b+1} - |S_i|$ is used. Note that if $|S_i|$ is a power of two, the second part is particularly simple: it is always the b-bit representation of r. If $|S_i| = 1$, the second part is empty.

The most well-known examples of codes of this form are the Golomb codes [1], which correspond to the case where the index function is $f(z) = \lfloor z/g \rfloor$ for some positive integer g so that each S_i consists of g members. The resulting code family is a set of optimal codes for geometric distributions [2]. We refer to the more general class of codes as *generalized Golomb codes.* Howard [3] generalizes this further in defining "unary prefix/suffix" (UP/S) codes, which allow any prefix-free code to be used to identify members of each S_i. Howard tabulates several codes for nonnegative integers, formulated as UP/S codes.[2]

The condition that the index function f is onto implies that no set S_i is empty and ensures that the resulting code is efficient at least in the sense of satisfying the Kraft inequality with equality. Thus, the code is prefix-free, and no codeword can be added without violating the prefix-free condition. One would expect the index functions of useful generalized Golomb codes to be nondecreasing, so that each set S_i consists of consecutive integers. This is a reasonable requirement if the probability distribution on the source integers z is nonincreasing. By using an index function f that increases slower than linearly, we hope to more effectively encode source distributions that decay more slowly than a geometric distribution.

In the remainder of this article, we'll give two examples of generalized Golomb code families and illustrate their application to runlength coding, especially in the context of encoding quantized subbands of wavelet-transformed images.

B. Two Simple Families of Generalized Golomb Codes

1. A Generalization of Exponential-Golomb Codes. Our first new family of codes, which we call *generalized exponential-Golomb codes*, has a single integer parameter $m > 0$. The code with parameter m is defined by the index function

$$f_m(z) = \left\lfloor \log_2 \left(1 + \frac{z}{m}\right) \right\rfloor$$

For this code, the first set S_0 has m elements, and each subsequent set has twice as many elements as the preceding one, so $|S_i| = m2^i$.

We're most interested in the case where $m = 2^s$ for some nonnegative integer s, so that the size of each S_i is a power of two. This simplifies encoding and decoding because all elements of S_i have codewords with the same length. The resulting set of codes are Teuhola's *exponential-Golomb codes* [4], which are equivalent to the Fiala–Greene "start-step-stop" codes [5], with the parameters (start, step, stop) equal to $(s, 1, \infty)$.

To encode z in this case, we can compute $w = 1 + \lfloor z/2^s \rfloor$ so that the index $f_{2^s}(z)$ equals the number of bits following the leading '1' bit in the binary representation of w. The codeword for z consists of the unary encoding of $f_{2^s}(z)$, followed by the $f_{2^s}(z)$ least-significant bits of the binary representation of w, and then the s least-significant bits of the binary representation of z. Thus, the codeword length is

[2] Howard's generalization is rather broad, as he points out that *any* prefix-free code can be formulated as a UP/S code. Note that in the example codes of [3, Table 2] only the Golomb code even makes use of sets S_i with sizes that aren't powers of two.

$$\ell_s(z) = 1 + 2f_{2^s}(z) + s = 1 + s + 2\left\lfloor \log_2(1 + z2^{-s}) \right\rfloor \tag{1}$$

If we reverse the order of the last $i + s$ bits (it's easy for the decoder to accommodate this), then encoding of z can be accomplished using the following C code:

```
w = ww = 1 + (z>>s);
while (w > 1) {
   output_bit(1);
   w >>= 1;
}
output_bit(0);
while(s > 0) {
   output_bit(z & 1);
   z >>= 1;
   s--;
}
while(ww > 1) {
   output_bit(ww & 1);
   ww >>= 1;
}
```

Fiala and Greene [5] also give pseudocode for encoding.

2. Another Generalized Golomb Code Family. As a second example of a family of generalized Golomb codes, we form a fairly trivial extension of the $s = 0$ exponential-Golomb code described above. Each code in the family is identified by a single integer parameter $t \geq 0$ and is defined by the index function

$$h_t(z) = \begin{cases} z, & \text{if } z \leq t \\ t + \lfloor \log_2(1 + z - t) \rfloor, & \text{otherwise} \end{cases}$$

The first $t+1$ sets $S_0, S_1, \cdots, S_t$ each have a single element, and subsequent sets each have twice as many elements as the preceding one. The codeword for integer z has length

$$\ell = \begin{cases} 1 + z, & \text{if } z \leq t \\ 1 + t + 2\lfloor \log_2(1 + z - t) \rfloor, & \text{otherwise} \end{cases}$$

Compared to the codes of Section I.B.1, these codes should be more appropriate for probability distributions with more mass concentrated near the origin.

C. A Matching Probability Distribution Model

Given a discrete random source with probability distribution $\{p_j\}$, ideally we'd like to devote $\log_2(1/p_j)$ bits to encoding the jth element of the distribution. Put another way, we would expect an effective binary code for the probability distribution $\{p_j\}$ to have codeword lengths $\{\ell_j\}$ satisfying $p_j \approx 2^{-\ell_j}$.

We follow this line of reasoning to determine a probability distribution that is well matched to the generalized exponential-Golomb codes of Section I.B.1. To encode integer z, we unary encode the value of the index $f_m(z)$, which requires $1 + f_m(z)$ bits, followed by about $\log_2 |S_{f_m(z)}|$ additional bits. Thus, the length of the codeword for z is approximately

$$\ell \approx 1 + f_m(z) + \log_2 |S_{f_m(z)}|$$

and since $|S_{f_m(z)}| = m 2^{f_m(z)}$, this is approximately

$$\begin{aligned}\ell &\approx 1 + \log_2 m + 2 f_m(z) \\ &\approx 1 + \log_2 m + 2\left[\log_2\left(1 + \frac{z}{m}\right) - \frac{1}{2}\right] \\ &= -\log_2 m + 2\log_2(m+z)\end{aligned}$$

and we have

$$2^{-\ell} \approx m(m+z)^{-2}$$

Because of the approximations used, the leading factor of m isn't the right normalization for a probability distribution. The properly normalized probability distribution function of this form is

$$p_\alpha(z) = \frac{1}{\Psi'(\alpha)}(\alpha + z)^{-2}, \;\; z = 0, 1, 2, \cdots, \tag{2}$$

where $\alpha > 0$, Ψ' is the first derivative of the digamma function $\Psi(y) = \Gamma'(y)/\Gamma(y)$, and Γ is the Euler gamma function. When $\alpha = 1$, this gives a zeta distribution on $z + 1$.

A random variable with probability distribution $\{p_\alpha(z)\}$ has infinite mean and entropy

$$\mathcal{H}_\alpha = \log_2 \Psi'(\alpha) - \frac{2}{\Psi'(\alpha)\ln 2}\zeta^{(1,0)}(2, \alpha) \quad \text{(bits)}$$

where ζ is the generalized Riemann zeta function $\zeta(y, a) = \sum_{j=0}^{\infty}(j + a)^{-y}$ and $\zeta^{(1,0)}(y, a)$ denotes $(\partial/\partial y)\zeta(y, a)$.

In the next section, we'll see an application where this probability model provides a good fit to empirically observed data sets and exponential-Golomb codes are rather effective. We remark, however, that no exponential-Golomb code can be an optimum code for a probability distribution of the form of Eq. (2). This can be established by observing that the codewords for the members of S_i all have the same length, but for sufficiently large i the expected codeword length can be reduced by using a one-bit shorter codeword for the most-probable symbol in S_i and one-bit longer codewords for the two least-probable symbols in S_i.[3]

[3] This analysis suggests that the generalized exponential-Golomb codes resulting from using values of m roughly halfway between powers of two may achieve lower redundancy for this probability distribution than the exponential-Golomb codes. We leave this as a possible topic for future research.

II. Application of Exponential-Golomb Codes to Coding Runlengths in Wavelet-Transformed Image Data

Runlength coding is a common application for variable-length codes defined on the nonnegative integers. For a discrete source with a dominant output symbol that occurs with sufficiently high probability, it is more effective to encode the lengths of runs of this most-probable symbol separately from the remaining symbols than to apply a single Huffman code to the source [6].

As a specific instance of runlength coding, we examine the application of exponential-Golomb codes to coding the lengths of runs of zeros in quantized subbands of wavelet-transformed images.

A. Probability Model and Coding Effectiveness

We wish to encode the length of each run of zeros in a uniformly quantized subband of a wavelet-transformed image. A separate code would be used for the nonzero quantized samples, but for now we consider only the runlength coding problem.

If the quantized subband data samples were independent and identically distributed (IID), then runlengths would be geometrically distributed. But, as we'll see below, the geometric model is empirically not a good fit to the distribution of runlengths. This is perhaps not surprising, as subband data samples are clearly not IID—in fact, many good wavelet-based image compression algorithms obtain their good performance in part by exploiting dependencies between nearby samples in the subbands [7–9].

For a given data set, let q_j denote the observed frequency of the jth symbol. The zeroth-order empirical entropy of the data set is

$$-\sum_j q_j \log_2 q_j \qquad \text{(bits)}$$

This gives an estimate of the rate at which one could encode the source that produced the data set, because an ideal entropy coder for an IID source whose actual probability distribution is $\{q_j\}$ would produce this number of bits per source sample on average. This is usually not the entropy of the source that produced the data, because the empirical observations of the symbol frequencies usually do not exactly equal the underlying probability distribution, and because this calculation ignores any dependencies that might be present in the source samples.

Given an empirical probability distribution and one or more parameterized probability distribution models, we'd like to select parameters for the models that give the best fit to the empirical distribution, and we'd like to measure how well each model fits the observed distribution. If we use an ideal entropy coder matched to a probability distribution $\{\hat{p}_j\}$ to encode a source whose actual probability distribution is $\{p_j\}$, then the rate obtained is

$$-\sum_j p_j \log_2 \hat{p}_j \qquad \text{(bits/symbol)}$$

Thus, in a data compression application, if we're using a parameterized distribution $\{\hat{p}_j\}$ to model an empirical distribution $\{p_j\}$, then we should select our model parameters to minimize the above quantity. This is equivalent to minimizing the quantity

$$-\sum_j p_j \log_2 \hat{p}_j - H(\{p_j\}) = \sum_j p_j \log_2 \left(\frac{p_j}{\hat{p}_j}\right) \qquad \text{(bits)}$$

which is the Kullback–Leibler (K-L) distance of $\{\hat{p}_j\}$ from $\{p_j\}$. Here $H(\{p_j\})$ is the entropy of the dis-tribution $\{p_j\}$. The K-L distance gives a measure of how well a distribution model fits an empirical distribution.

As a test case, we use the "forest_2kb4" image from a set of test images used in the development of the emerging Consultative Committee for Space Data Systems (CCSDS) image compression standard [10]. This image contains 2048×2048 pixels, at a bit depth of 10 bits/pixel, taken from a single band of Advanced Very High Resolution Radiometer (AVHRR) instrument data. We apply a single two-dimensional decomposition of the image using the (5,3) integer discrete wavelet transform using the implementation described in [11]. In the examples below, we use the horizontally high-pass, vertically low-pass subband data, arranged in raster-scan order. The data samples are quantized with a uniform quantizer before encoding.

Example 1. Quantizing the subband with a stepsize of 12 results in 83 percent of the subband samples being quantized to zero, giving about 180,000 runlengths to encode. The distribution of runlengths has zeroth-order empirical entropy of 2.69 bits per runlength. If we model the runlengths using a geometric distribution Prob[runlength $= z$] $= (1-q)q^z$, we find a minimum K-L distance of 1.20 bits at $q = 0.83$. The probability distribution of Eq. (2) gives a much better fit, obtaining minimum K-L distance of 0.01 bits at $\alpha = 1.159$. Figure 1(a) shows the empirical distribution and the two probability models. The best Golomb code for the empirical distribution has parameter $g = 6$ and gives average codeword length of 3.90 bits per runlength. The exponential-Golomb code with parameter $s = 0$ does better, giving an average cost of 2.75 bits per runlength.

Example 2. Using a quantizer stepsize of 4 for this subband results in 59.5 percent of the quantized samples being zero, or about 425,000 runlengths to encode. Here the zeroth-order empirical entropy of the runlength distribution is 2.05 bits per runlength. The K-L distance for the geometric distribution is minimized at $q = 0.60$, giving a value of 0.36 bits. By contrast, the probability distribution of Eq. (2) gives minimum K-L distance of 0.01 bits at $\alpha = 0.899$. Figure 1(b) compares the empirical distribution with the two models. Picking the optimum parameter for the Golomb code ($g = 1$) results in unary encoding of runlengths, giving an average codeword length of 2.47 bits per run. By contrast, the exponential-Golomb code with parameter $s = 0$ gives 2.18 bits per runlength, and the code of Section I.B.2 with $t = 1$ does slightly better, with an average rate of 2.14 bits per runlength.

B. Parameter Selection in Adaptive Coding of Subband Data

We've seen that the probability distribution of Eq. (2) is a good model for the distribution of runlengths in a quantized subband of a wavelet-transformed test image, and we've seen that the generalized Golomb codes can provide effective coding of these runlengths when the code parameter s is well chosen. In this section, we show how to determine the value of the parameter s to use when applying exponential-Golomb codes to encode a source having the probability distribution of Eq. (2) with a known value of the parameter α. We apply this result to the problem of adaptively determining the code parameter s on-the-fly.

Using an exponential-Golomb code with parameter s to encode an IID source having the probability distribution function given in Eq. (2) gives, after some simplification, expected codeword length

$$L_s(\alpha) \triangleq E\big[\ell_s(z)\big] = \sum_{z=0}^{\infty} p_\alpha(z)\ell_s(z) = 1 + s + \frac{2}{\Psi'(\alpha)} \sum_{j=1}^{\infty} \Psi'\big(2^s(2^j-1)+\alpha\big)$$

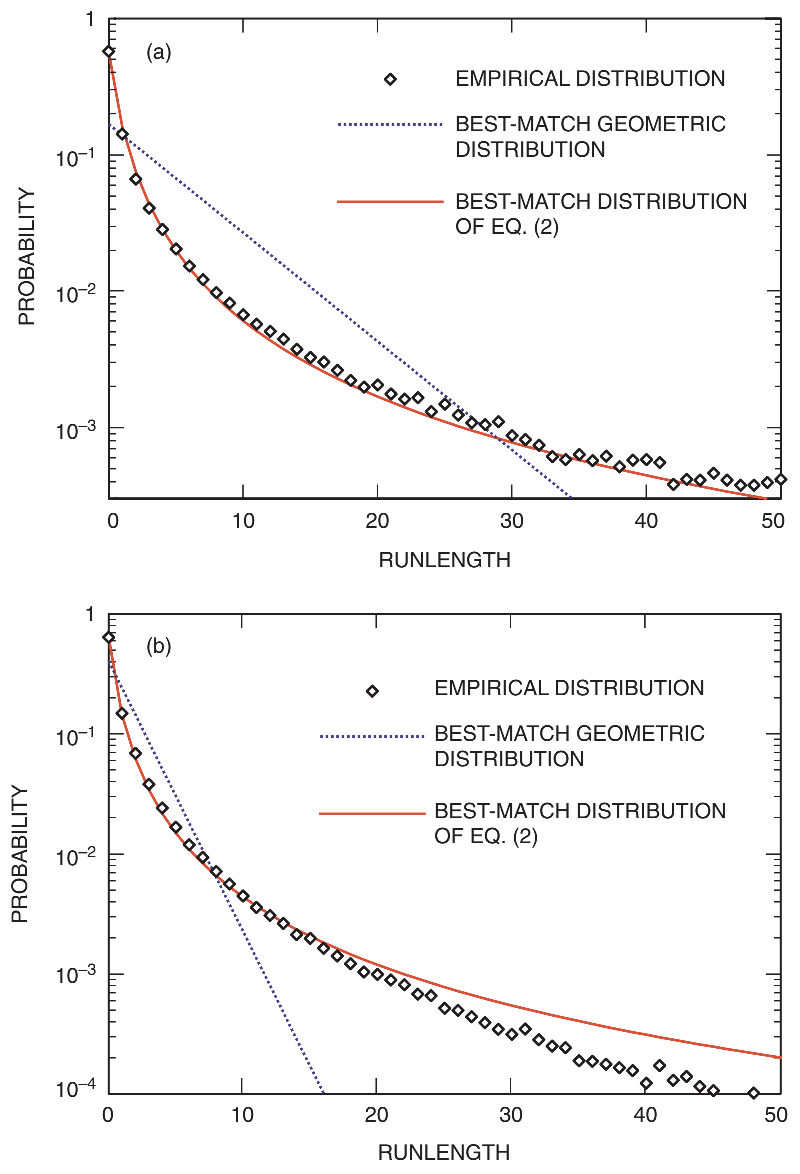

Fig. 1. Empirical distribution of lengths of runs of zeros for an image subband quantized using a uniform scalar quantizer, and probability distribution models with parameters selected to minimize K-L distance of the model from the empirical distribution, when (a) the quantizer stepsize is 12 and (b) the quantizer stepsize is 4.

[refer to Eq. (1) for the codeword length $\ell_s(z)$]. We'd like to pick the value of the parameter s that minimizes the expected codeword length. This is equivalent to minimizing the redundancy $L_s(\alpha) - \mathcal{H}_\alpha$, shown in Fig. 2 for several values of s.

The optimum value of the parameter s for a given α can be determined from a table of the values of α where we switch from one value of s to the next. Toward this end, we have numerically determined the first several values of α_s^*, where α_s^* is defined to be the value that gives $L_s(\alpha_s^*) = L_{s+1}(\alpha_s^*)$. The results are in Table 1.

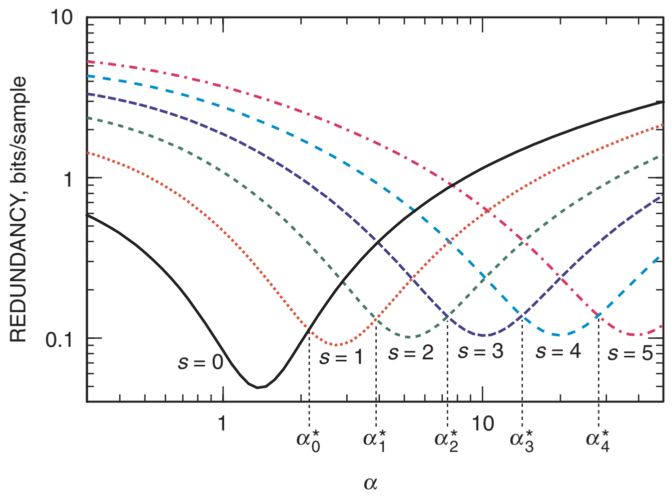

Fig. 2. Redundancy of exponential Golomb codes applied to an IID source with probability distribution given by Eq. (2), shown for different values of code parameter *s*.

Table 1. Values of the parameter in the distribution of Eq. (2) where the optimum exponential Golomb code parameter changes from s to $s + 1$, and associated average codeword lengths.

s	α_s^*	$L_s(\alpha_s^*)$, bits
0	2.15	3.796
1	3.90	4.802
2	7.36	5.804
3	14.24	6.805
4	28.00	7.805
5	55.51	8.805
6	110.53	9.805
7	220.56	10.805
8	440.62	11.805
9	880.75	12.805
10	1761.00	13.805
11	3521.49	14.805

To make practical use of the results in Table 1 in an adaptive coding situation, we would like to relate these transitions between different values of s to corresponding values of some quantity that we can easily estimate on-the-fly. We observe in Table 1 that, for each s,

$$L_s(\alpha_s^*) \approx s + 3.8$$

and thus when s is optimally chosen, the average codeword length is (approximately) in the range $[s + 2.8, s + 3.8]$. This suggests a simple adaptive strategy for choosing our code parameter s. We maintain an estimate $\hat{L}$ of the average codeword length and compute the difference $\hat{d} \triangleq \hat{L} - s$. When

$\hat{d} < 2.8$, we *decrease* s by 1 (and so effectively increase $\hat{d}$ by 1), and when $\hat{d} > 3.8$, we *increase* s by 1 (and so effectively decrease $\hat{d}$ by 1). We apply this strategy as part of a subband coding algorithm in Section III.A.

For sufficiently small values of α, it can be shown that the codes of Section I.B.2 with parameter $t = 1$ or $t = 2$ can give a marginal improvement over the optimum exponential-Golomb code. The improvement is small enough that we do not attempt to incorporate these codes as part of an adaptive coding strategy.

III. Subband Coding Algorithms

We now describe three different options for coding quantized subband data. In Section IV, we compare the performances of these algorithms on three sample images. We assume that the quantizer stepsize is fixed before coding begins, and we do not address the issue of selecting this stepsize. References [12,13] present a solution to the problem of selecting quantizer stepsize to meet a target bit rate.

In each coding algorithm, the quantized subband samples are taken in raster-scan order; thus, the input data stream is effectively one-dimensional. The coding techniques described here have low complexity and do not rely on two-dimensional context models such as the one used in JPEG2000 [7]; consequently, the algorithms should not be expected to offer competitive performance.

A. Separate Encoding of Runlengths and Nonzero Samples Using Generalized Golomb Codes

The first algorithm encodes the quantized subband samples using two separate codes, alternating between an exponential-Golomb code to encode the length of each run of zeros and a Golomb code to encode the nonzero samples. For each code, a simple adaptive parameter estimation technique is used.

The exponential-Golomb code used to encode the lengths of runs of zeros has parameter s selected adaptively based on statistics from previously encoded runs. We maintain a nominal running count of the number of bits used to encode runlengths, B, and a count of the nominal number of runlengths encoded, R. We estimate the average codeword length as B/R and use this estimate to determine the parameter s of the exponential-Golomb code using the adaptive parameter selection method described in Section II.B.[4]

Following the encoding of a runlength, we increment R by 1 and increment B by the number of bits in the codeword just encoded. When R reaches some maximum value R_0, we rescale by setting $R = \lfloor R/2 \rfloor$ and $B = \lfloor B/2 \rfloor$. Adjusting the value of the rescaling interval R_0 changes the degree to which coding adapts to local statistics.

To encode the nonzero samples in the quantized subband, we use a simple adaptive technique based on Golomb codes that is nearly identical to the method used for the coding of prediction residuals in lossless image compression in [14, Section 2.3], but modified to account for the fact that the sample being encoded is nonzero. For a nonzero quantized subband sample x, we first compute

$$M(x) = \begin{cases} 2|x| - 1, & \text{if } x < 0 \\ 2|x| - 2, & \text{otherwise} \end{cases}$$

Note that M is an invertible function that maps the nonzero integers onto the nonnegative integers, interleaving the positive and negative values of x. Empirically it is observed that $M(x)$ approximately follows a geometric distribution. Thus, we encode x by applying a Golomb code to $M(x)$, selecting the code parameter based on an estimate of the expected value of

[4] In practice, one would implement this parameter selection technique without using a division operation.

$$\left\lfloor \frac{M(x)+1}{2} \right\rfloor = \begin{cases} |x|, & \text{if } x < 0 \\ |x|-1, & \text{otherwise} \end{cases} \approx |x| - \frac{1}{2} \tag{3}$$

The approximation follows under the assumption that the distribution on x is symmetric. We maintain a nominal running count of the number of nonzero samples encoded, N, and a nominal sum, A, of the quantity in the right-hand side of Eq. (3). The Golomb code parameter used to encode $M(x)$ is $g = 2^k$, where k is computed as [14]

$$k = \min\{j : 2^j N > A\}$$

After encoding a sample x, N is incremented by one and A is incremented by $|x| - 1/2$. (Alternatively, one could track twice this value so that only integer arithmetic is needed.) Whenever N reaches some maximum value N_0, we rescale by setting $N = \lfloor N/2 \rfloor$ and $A = \lfloor A/2 \rfloor$.

B. Joint Encoding of Runlengths and Magnitudes

An algorithm recently proposed for the emerging CCSDS image compression standard encodes quantized subband data by applying variable-length binary codes to jointly encode the length of a run of consecutive zeros along with information about the magnitude of the sample that follows the run [10,12,13]. The codeword is followed by refinement bits that pinpoint the exact value of the nonzero sample. The variable-length codes used could be derived via Huffman's algorithm applied to suitable test data sets. In a space-based application, a library of suitable codes to accommodate the bit rates and data sets of interest for different subbands would be stored onboard [12,13].

To limit the number of codewords in the variable-length code, runlengths that are jointly encoded with magnitude information must be shorter than some maximum length $R_{\max}$. A codeword representing a run of $R_{\max}$ zeros is included in the code to handle longer runs. For each nonzero sample x, one computes a "category," denoted $\mathrm{Cat}(x)$, according to

$$\mathrm{Cat}(x) = 1 + \lfloor \log_2 |x| \rfloor$$

Thus, $\mathrm{Cat}(x)$ is equal to the number of bits in the binary representation of $|x|$. The variable-length code is defined on the symbol alphabet consisting of all possible $(\text{runlength}, \mathrm{Cat}(x))$ pairs plus the symbol representing a maximum-length run. A codeword for a $(\text{runlength}, \mathrm{Cat}(x))$ pair is followed by $\mathrm{Cat}(x)$ additional bits that specify the value of x within its category.

The performance of this algorithm depends on the particular variable-length codes used. To bound the best performance achievable by this technique, we can calculate the rate achieved by the optimum variable-length code for the symbols to be encoded in a given quantized subband. We do this in Section IV, determining the optimum variable-length codes by applying Huffman's algorithm directly to the empirical symbol distributions.

C. Direct Coding of Samples

The runlength-based coding techniques of Sections III.A and III.B are not well suited to data in which runs of zeros are short and infrequent. Thus, we have also measured the performance of applying a variable-length code directly to the quantized subband samples, i.e., without any runlength coding. This is done using Golomb codes with the adaptive parameter estimation technique used for the coding of prediction residuals in lossless image compression in [14, Section 2.3].

It was shown in [6] that for an IID source, when a zero occurs with probability less than 2/5, runlength coding of zeros is less effective than simply Huffman coding the source directly. Although we've argued in Section II.A that quantized subband data are not IID, nevertheless runlength coding becomes less effective than the direct coding approach when the probability that a sample is zero becomes small.

IV. Results

We now give some results comparing the performance of the three coding techniques described in Section III. For test images, we use the "forest_2kb4" and "ice_2kb1" AVHRR images used in the CCSDS image compression standard development. Each of these images is 2048×2048 pixels with a bit depth of 10 bits/pixel, taken from a single band from the AVHRR instrument. As a third image we use a 1024×1024 pixel image of the surface of Mars, at a bit depth of 12 bits/pixel. The image is a mosaic of several smaller images from the Mars Pathfinder mission.

We apply a single-stage two-dimensional subband decomposition to the image using the (5,3) integer discrete wavelet filter as implemented in [11]. As input data for each coding algorithm, we use the horizontally high-pass, vertically low-pass subband data, quantized and arranged in raster-scan order. For the AVHRR images, this gives 2^{20} data samples, and for the Mars image this gives 2^{18} samples.

Figures 3 through 5 show rate-distortion performance on the test data. The graphs show only the contribution to mean square error (MSE) and rate from the subband coded. Consequently, the results in the figures are not directly comparable to rate-distortion results from a complete image compression algorithm. Such an algorithm would also have to select appropriate quantizer stepsizes for the subbands of the transformed image.

The curves plotted show the performance of our adaptive coding technique (described in Section III.A), the bound on performance achievable by the technique that jointly encodes runlengths and magnitude information (Section III.B), the performance achieved by the direct adaptive Golomb coding technique (Section III.C), and the zeroth-order empirical entropy (Section II.A). The shaded region of each graph shows where the frequency zeros are roughly 2/5 or less, and thus one might expect runlength coding to be less effective.

For our adaptive coding algorithm of Section III.A, we initialize $B = 10$, $R = 2$, $N = 2$, and $A = 12$, and use a rescaling interval of $R_0 = 12$ for runlengths and $N_0 = 16$ for nonzero samples. For the method of Section III.B, we use the suggested maximum runlength of $R_{\max} = 64$ here. For the direct coding approach of Section III.C, we use a rescaling interval of 32 samples.

We observe in Figs. 3 through 5 that our adaptive coding technique achieves performance that is comparable to the bound on the best performance that could be achieved using the technique of Section III.B. At some quantizer stepsizes, our adaptive coding technique gives a bit rate that is lower than the zeroth-order empirical entropy. This is possible because by encoding runlengths we're exploiting some dependencies in the data and also because the code parameter estimation techniques adapt to local statistics. The coder of Section III.B exploits dependencies in a slightly different way, by jointly coding runlengths and sample magnitudes.

The figures show that, at sufficiently high bit rates (i.e., sufficiently small quantizer stepsizes), direct adaptive Golomb coding of subband samples outperforms our adaptive runlength coding technique. In practice, it would be straightforward to adaptively switch between the two techniques based on the frequency of zeros in the data.

Our adaptive coding technique was motivated by a desire for a simple encoder that eliminated the need for the multiple code tables required by the encoder of Section III.B. We conclude with a few remarks on comparisons between the two techniques.

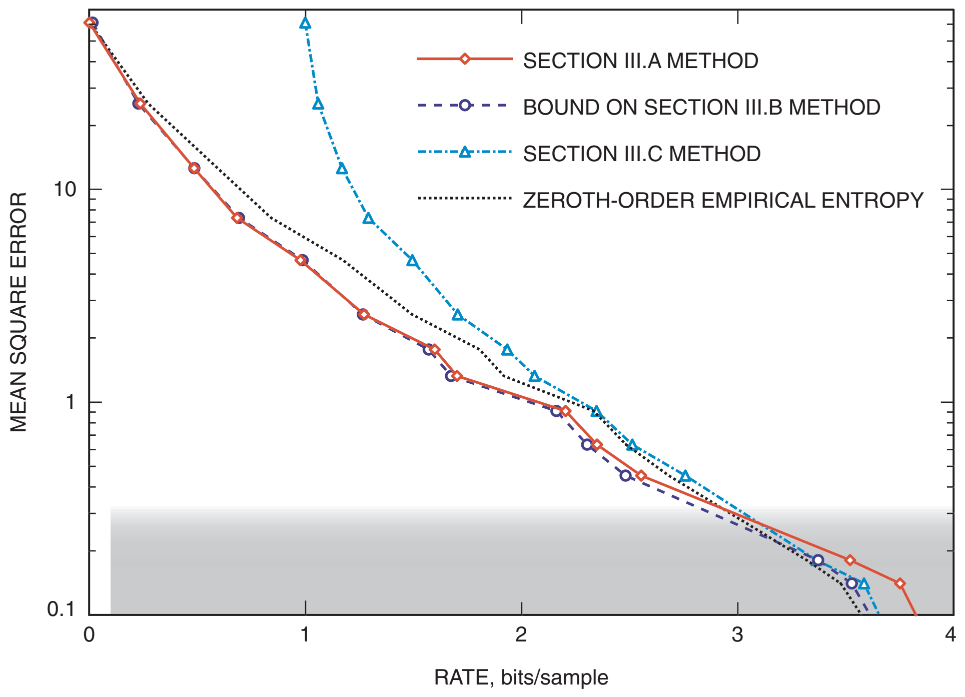

Fig. 3. Rate-distortion performance for coding a subband of the "forest_2kb4" CCSDS test image. The shaded region shows the range of MSE values where the fraction of zeros is roughly 2/5 or less, and thus one would expect runlength coding to be outperformed by direct coding.

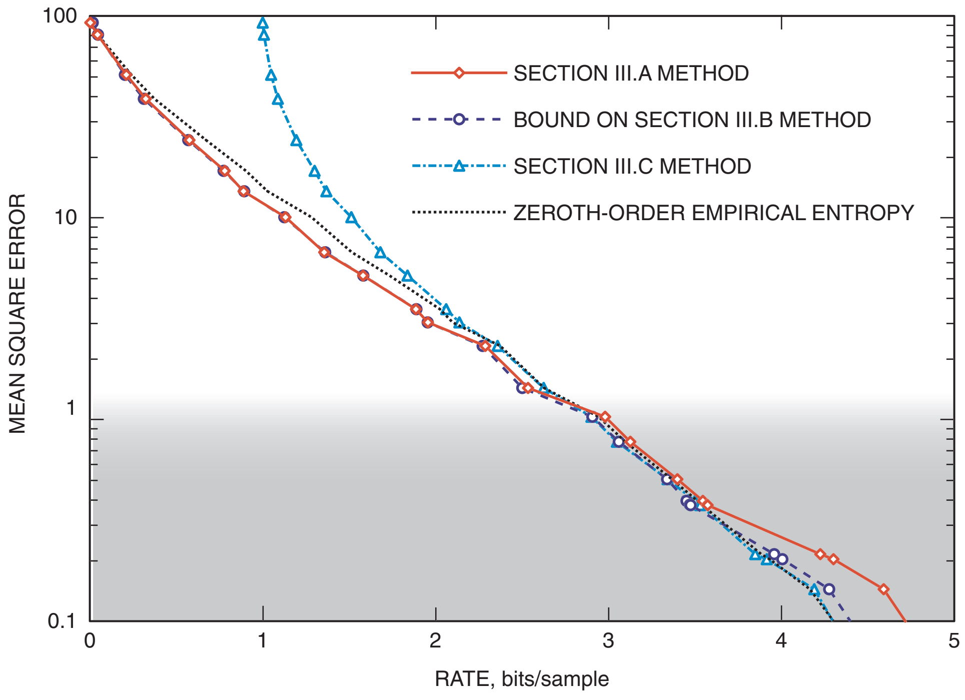

Fig. 4. Rate-distortion performance for coding a subband of the "ice_2kb1" CCSDS test image. The shaded region shows the range of MSE values where the fraction of zeros is roughly 2/5 or less, and thus one would expect runlength coding to be outperformed by direct coding.

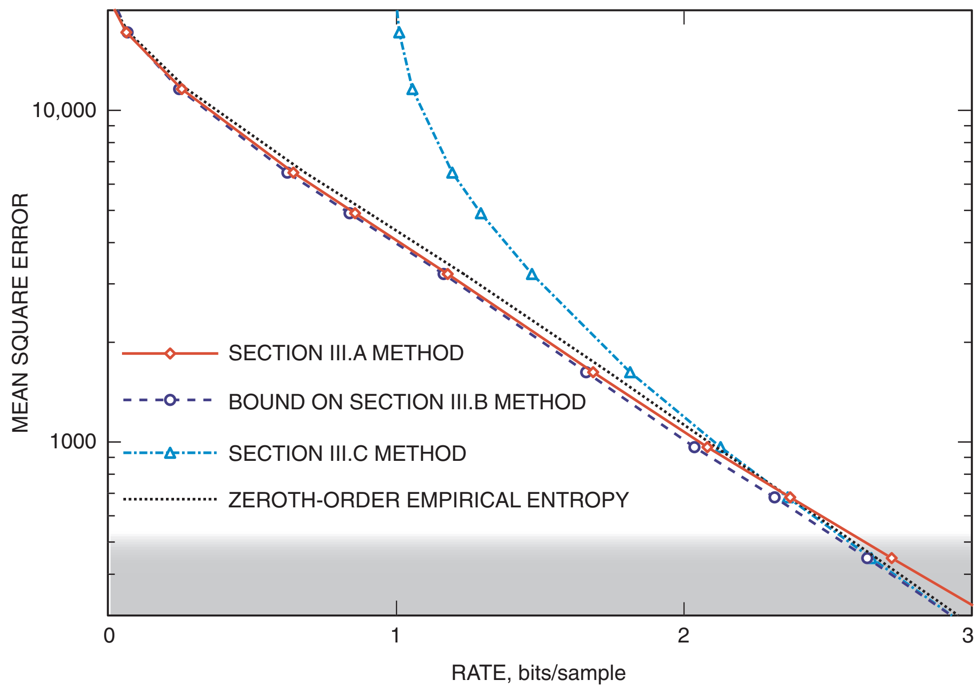

Fig. 5. Rate-distortion performance for coding a subband of the Mars surface image. The shaded region shows the range of MSE values where the fraction of zeros is roughly 2/5 or less, and thus one would expect runlength coding to be outperformed by direct coding.

Coding runlengths and sample values via codes defined on the nonnegative integers allows arbitrarily large input values. Accommodating a change in image size or the bit depth of the image pixels thus doesn't require any change in the coding technique.

In a space-based application, protection from data loss can be provided by partitioning a data set into separate segments that can be decoded independently. In this scenario, the state of our encoder can be specified in the appropriate packet headers by encoding the four quantities A, N, B, and R, used to select the parameters of the two codes. Thus, adaptivity in our algorithm does not need to come at the price of reduced error containment.

Finally, we note that simple modifications to the technique have the potential to offer improved performance. These includes refinements in the methods of selecting the two code parameters, either by incorporating context modeling or by exploiting dependencies between the runlengths and the nonzero samples.

References

[1] S. W. Golomb, "Run-Length Encodings," *IEEE Transactions on Information Theory*, vol. IT-12, no. 3, pp. 399–401, July 1966.

[2] R. G. Gallager and D. C. Van Voorhis, "Optimal Source Codes for Geometrically Distributed Integer Alphabets," *IEEE Transactions on Information Theory*, vol. IT-21, no. 2, pp. 228–230, March 1975.

[3] P. G. Howard, “Interleaving Entropy Codes,” *Proc. Compression and Complexity of Sequences 1997*, Salerno, Italy, pp. 45–55, 1998.

[4] J. Teuhola, “A Compression Method for Clustered Bit-Vectors,” *Information Processing Letters,* vol. 7, pp. 308–311, October 1978.

[5] E. R. Fiala and D. H. Greene, “Data Compression with Finite Windows,” *Communications of the ACM*, vol. 32, pp. 490–505, April 1989.

[6] K.-M. Cheung and A. Kiely, “An Efficient Variable Length Coding Scheme for an IID Source,” *Proc. 1995 IEEE Data Compression Conference*, Snowbird, Utah, pp. 182–191, March 28–30, 1995.

[7] M. D. Adams, *The JPEG-2000 Still Image Compression Standard*, ISO/IEC JTC 1/SC 29/WG 1 N 2412, September 2001.

[8] D. Taubman, “High Performance Scalable Image Compression with EBCOT,” *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[9] E. S. Hong, R. E. Ladner, and E. A. Risken, “Group Testing for Wavelet Packet Image Compression,” *Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, pp. 73–82, March 2001.

[10] P. S. Yeh, G. A. Moury, and P. Armbruster, “CCSDS Data Compression Recommendation: Development and Status,” *Proc. SPIE*, vol. 4790, Seattle, Washington, July 7–12, 2002.

[11] M. D. Adams and F. Kossentini, “Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis,” *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1010–1024, June 2000.

[12] C. Parisot, M. Antonini, M. Barlaud, C. Lambert-Nebout, C. Latry, and G. Moury, “On-Board Strip-Based Wavelet Image Coding for Future Space Remote Sensing Missions,” *Proc. 2000 IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2000)*, vol. 6, Honolulu, Hawaii, pp. 2651–2653, July 24–28, 2000.

[13] C. Parisot, M. Antonini, and M. Barlaud, “EBWIC: A Low Complexity and Efficient Rate Constrained Wavelet Image Coder,” *Proc. 2000 International Conference on Image Processing (ICIP2000)*, vol. 1, Vancouver, British Columbia, Canada, pp. 653–656, September 10–13, 2000.

[14] M. J. Weinberger, G. Seroussi, and G. Sapiro, “LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm,” *Proc. 1996 IEEE Data Compression Conference*, Snowbird, Utah, pp. 140–149, March 31–April 3, 1996.