Evaluation of JPEG-LS, the new lossless and near-lossless still image compression standard, for compression of high-resolution elevation data*

Shantanu D. Rane and Guillermo Sapiro
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA
guille@ece.umn.edu.

Abstract

The compression of elevation data is studied in this paper. The performance of JPEG-LS, the new international ISO/ITU standard for lossless and near-lossless still image compression, is investigated both for data from the USGS Digital Elevation Model (DEM) data and the navy-provided Digital Terrain Model (DTM) data. Using JPEG-LS has the advantage of working with a standard algorithm. Moreover, in contrast with algorithms like the popular JPEG-lossy standard, this algorithm permits the completely lossless compression of the data, as well as a controlled lossy mode where a sharp upper bound on the elevation error is selected by the user. The results here reported suggest that JPEG-LS can immediately be adopted for the compression of elevation data for a number of applications.

*We thank Dr. Gary Hewer and Dr. Larry Peterson from China Lake (NAWCWD) for providing high-resolution DTM data. (Questions regarding the distribution of this data should be addressed to Dr. Gary Hewer.) Dr. Larry Peterson also provided DTM reading software. Dr. Wen Masters and Dr. Carey Schwartz from ONR introduced us to the problem and motivated this investigation. Andres Solé produced Figure 1. GS thanks Dr. Gadiel Seroussi and Dr. Marcelo Weinberger from Hewlett-Packard Laboratories for teaching him a lot on image compression. The tests in this paper were done with the public domain version of JPEG-LS distributed by Hewlett-Packard Laboratories. This work was partially supported by a grant from the Office of Naval Research ONR-N00014-97-1-0509, the Office of Naval Research Young Investigator Award, the Presidential Early Career Awards for Scientists and Engineers (PECASE), a National Science Foundation CAREER Award, and by the National Science Foundation Learning and Intelligent Systems Program (LiS).
1 Introduction

The compression of elevation data is fundamental for a number of applications, including storage, transmission, and real time visualization in navigation exercises. The storage and transmission of high resolution elevation information can consume considerable amounts of resources, and with the increased interest in mapping the earth and having maps for real time navigation, the development of compression techniques to help in these tasks is becoming an absolute must.

In the paper we investigate the use of JPEG-LS, the new ISO/ITU standard for lossless and near-lossless compression of continuous-tone images, for the compression of high resolution elevation data. The advantages of using a standard algorithm are numerous, and they include the possibility of any user to compress and de-compress the data, and the wide availability of supporting software and hardware. This is what has motivated the development of popular standards such as JPEG (lossy still image compression), MPEG (lossy video compression), the new JPEG-LS (lossless and near-lossless still image compression), and the forthcoming JPEG2000. In particular, for elevation data, JPEG-LS has a number of advantages over other standards (e.g., JPEG). First of all, it is capable of lossless compression. This is not just important but mandatory for data-bases as those collected and maintained by the USGS. Secondly, it has a near-lossless mode that permits to control the maximal error in pixel value, thereby, the maximal error in elevation in the reconstructed image. This is fundamental for applications such as landing, where the terrain slope is of primary importance. This makes the standard JPEG-LS a perfect candidate for the compression of elevation maps. While other papers have reported results on the compression of elevation maps, e.g., [1] and references therein, to the best of our knowledge none of them have these three fundamental qualities all together (using a standard algorithm, and having both lossless and controlled-lossy modes).

Digital elevation data for a region normally consists of an array of elevations for ground positions at regularly spaced intervals, see Figure 1. In this report we investigate data from both the USGS Digital Elevation Model (DEM) and the Navy Digital Terrain Models (DTM). The exact details and resolutions for this data are given in the results section.

This paper is divided in two major parts. In the first part a brief description of JPEG-LS is provided, while the second part reports results on applying JPEG-LS to the elevation data.

2 A brief description of the JPEG-LS algorithm

JPEG-LS achieves state-of-the-art compression rates at very low computational complexity and memory requirements. These characteristics are what brought to the selection of JPEG-LS, which is based on the LOCO-I algorithm developed at Hewlett-Packard Laboratories, as the new ISO/ITU standard for lossless and near-lossless still image compression.

The basic block diagram of JPEG-LS is given in Figure 2. In this section we briefly describe the main components of JPEG-LS. A detailed description of this algorithm is contained in [2], from where the brief description below was taken.

2.1 Modeling and prediction

Modeling in lossless image compression can be formulated as an inductive inference problem [3]. In a raster scan, after having scanned past data, one infers the next pixel value by assigning a
conditional probability distribution to it. In state-of-the-art lossless image compression schemes, this probability assignment is generally broken into the following components:

a. A prediction step, in which a deterministic value $\hat{x}_{t+1}$ is guessed for the next sample $x_{t+1}$ based on a finite subset (a causal template) of the available past sequence $x^t$. 

b. The determination of a context in which $x_{t+1}$ occurs. The context is a function of a (possibly different) causal template.

c. A probabilistic model for the prediction residual (or error signal) $\epsilon_{t+1} \triangleq x_{t+1} - \hat{x}_{t+1}$, conditioned on the context of $x_{t+1}$.

The prediction and modeling units in JPEG-LS are based on the causal template depicted in Figure 3, where $x$ denotes the current sample, and $a$, $b$, $c$, and $d$, are neighboring samples in the relative positions shown in the figure. The dependence of $a$, $b$, $c$, $d$, and $x$, on the time index $t$ has been deleted from the notation for simplicity. Moreover, by abuse of notation, we will use $a$, $b$, $c$, $d$, and $x$ to denote both the values of the samples and their locations. The use of the template of Figure 3 implies a buffering requirement of one scan line.
Specifically, the fixed predictor in JPEG-LS guesses:

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

(1)

The predictor (1) picks $b$ in many cases where a vertical edge exists left of the current location, $a$ in many cases of an horizontal edge above the current location, or $a + b - c$ if no edge has been detected. The latter choice would be the value of $x$ if the current sample belonged to the “plane” defined by the three neighboring samples with “heights” $a$, $b$, and $c$. This expresses the expected smoothness of the image in the absence of edges.

2.2 Context modeling

Reducing the number of parameters is a key objective in a context modeling scheme. The total number of parameters in the model depends on the number of free parameters defining the coding distribution at each context and on the number of contexts.

2.2.1 Coding distributions

It is an accepted observation, adopted by JPEG-LS, that the global statistics of residuals from a fixed predictor in continuous tone images are well modeled by Tow-Sided-Geometric-Distributions (TSGD) centered at zero. For context-conditioned predictors, this distribution has an offset, and this is addressed by JPEG-LS as well.
The prediction residual $\epsilon$ can take on any value in the range $-\alpha < \epsilon < \alpha$, where $\alpha$ is the size of the image alphabet. Actually, given the predicted value $\hat{x}$ (known to both the encoder and decoder from the causal template), $\epsilon$ can take on only $\alpha$ possible different values. This property is exploited in JPEG-LS by reducing, modulo $\alpha$, the actual value of the prediction residual to a value between $-\lfloor \alpha/2 \rfloor$ and $\lfloor \alpha/2 \rfloor - 1$, thus remapping large prediction residuals to small ones. Merging the “tails” of peaked distributions with their central part does not significantly affect the original two-sided geometric behavior. In the common case in which $\alpha = 2^\beta$ (i.e., $\beta$ bits per sample), the above remapping consists of just interpreting the $\beta$ least significant bits of $x - \hat{x}$ in $2^\beta$'s complement representation.

### 2.2.2 Context determination

The context that conditions the encoding of the current prediction residual in JPEG-LS is built out of the differences $g_1 = d - b$, $g_2 = b - c$, and $g_3 = c - a$. These differences represent the local gradient, thus capturing the level of activity (smoothness, edginess) surrounding a sample, which governs the statistical behavior of prediction errors. For further model size reduction, each difference $g_j$, $j = 1, 2, 3$, is quantized into a small (fixed) number of approximately equiprobable, connected regions by a quantizer $\kappa(\cdot)$ independent of $j$. This aims to maximize the mutual information between the current sample value and its context, an information-theoretic measure of the amount of information provided by the conditioning context on the sample value to be modeled. We refer to [4] and [5] for an in-depth theoretical discussion of these issues.

To preserve symmetry, the regions are indexed $-T, \ldots, -1, 0, 1, \ldots, T$, with $\kappa(g) = -\kappa(-g)$, for a total of $(2T + 1)^3$ different contexts. To further reduce the number of contexts, observe that, by symmetry, it is reasonable to assume that

$$\text{Prob}\{\epsilon_{t+1} = \Delta | C_t = [q_1, q_2, q_3]\} = \text{Prob}\{\epsilon_{t+1} = -\Delta | C_t = [-q_1, -q_2, -q_3]\}$$

where $C_t$ represents the quantized context triplet and $q_j = \kappa(g_j)$, $j = 1, 2, 3$. Hence, if the first non-zero element of $C_t$ is negative, the encoded value is $-\epsilon_{t+1}$, using context $-C_t$. This is anticipated by the decoder, which flips the error sign if necessary to obtain the original error value. By merging contexts of “opposite signs,” the total number of contexts becomes $((2T + 1)^3 + 1)/2$. For JPEG-LS, $T = 4$ was selected, resulting in 365 contexts.

To complete the definition of the contexts in JPEG-LS, it remains to specify the boundaries between quantization regions. For an 8-bit per pixel alphabet, the default quantization regions are $\{0\}$, $\pm\{1, 2\}$, $\pm\{3, 4, 5, 6\}$, $\pm\{7, 8, \cdots, 20\}$, $\pm\{e | e \geq 21\}$, as defined by the thresholds $T_1$, $T_2$, and $T_3$. However, the boundaries are adjustable parameters, except that the central region must be $\{0\}$. A suitable choice collapses quantization regions, resulting in a smaller effective number of contexts, with applications to the compression of small images. This will be important below for the partition of large elevation maps in small regions for semi-random access.

### 2.2.3 Bias cancellation

The systematic context-dependent biases (offsets) in prediction residuals deteriorate the performance of the Golomb-Rice coder used by JPEG-LS (see below), which relies heavily on Two-Sided-Geometric-Distributions (TSGD) of prediction residuals centered about zero. To alleviate the effect of systematic biases, JPEG-LS uses an error feedback aimed at “centering” the distributions of prediction residuals. This bias cancellation is based on keeping counters (per context) for the number
of total context occurrences and the accumulated prediction residual \((B)\). See [2] for details on this very low complexity approach.

2.3 Coding

2.3.1 Golomb codes and optimal prefix codes for the TSGD

To encode bias corrected prediction residuals distributed according to the TSGD, JPEG-LS uses a minimal complexity family of optimal prefix codes for TSGD’s, sequentially selecting the code among this family.

Golomb codes were first described in [6], as a means for encoding run lengths. Given a positive integer parameter \(m\), the Golomb code \(G_m\) encodes an integer \(y \geq 0\) in two parts: a \textit{unary} representation of \([y/m]\), and a \textit{modified binary} representation of \(y \mod m\) (using \([\log m]\) bits if \(y < 2^{\lfloor \log m \rfloor} - m\) and \(\lfloor \log m \rfloor\) bits otherwise). Golomb codes are optimal [7] for \textit{one-sided geometric} distributions of the nonnegative integers, i.e. distributions of the form \((1-\theta)\theta^y\), where \(0 < \theta < 1\). Thus, for every \(\theta\) there exists a value of \(m\) such that \(G_m\) yields the shortest average code length over all uniquely decipherable codes for the nonnegative integers.

The special case of Golomb codes with \(m = 2^k\) leads to very simple encoding/decoding procedures: the code for \(y\) is constructed by appending the \(k\) least significant bits of \(y\) to the unary representation of the number formed by the remaining higher order bits of \(y\) (the simplicity of the case \(m = 2^k\) was already noted in [6]). The length of the encoding is \(k + 1 + \lfloor y/2^k \rfloor\). We refer to codes \(G_{2^k}\) as \textit{Golomb-power-of-2} (GPO2) codes.

In order to use these codes, the TSGD’s have to be mapped into one-sided geometric distributions. In [8, 9], the TSGD parameter space \((\theta, \rho)\) (\(\theta\) gives the decay and \(\rho\) the bias or offset), is partitioned, and a different optimal prefix code corresponds to each class in the partition. Classes are associated with one of the following three one-to-one mappings onto the nonnegative integers. One mapping is applied to the integer \(\epsilon\) 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 \(\ell\), given by a many-to-one function of \(\theta\) and \(\rho\). The parameter of the Golomb code used follows from the corresponding value of \(\ell\).

The first mapping related to codes in [8, 9] is given by

\[
M(\epsilon) = \begin{cases} 
2\epsilon & \epsilon \geq 0, \\
2|\epsilon| - 1 & \epsilon < 0.
\end{cases}
\]  

(2)

For prediction residuals \(\epsilon\) in the range \(-\lfloor \alpha/2 \rfloor \leq \epsilon \leq \lfloor \alpha/2 \rfloor - 1\) the values \(M(\epsilon)\) are in the range \(0 \leq M(\epsilon) \leq \alpha - 1\). The mapping \(M(\epsilon)\) orders prediction residuals, interleaving negative values and positive values in the sequence \(0, -1, 1, -2, 2, \ldots\). It was first used by Rice in [10] to encode TSGDs centered at zero, by applying a GPO2 code to \(M(\epsilon)\).

2.3.2 Sequential parameter estimation

In keeping with the low complexity constraints set for JPEG-LS, as in [10], JPEG-LS uses the sub-family of codes for which the Golomb parameter is a power of 2. Furthermore, only codes based on the mapping (2) are used. The codeword assigned to an integer \(\epsilon\) is denoted \(\Gamma_k(\epsilon) = G_{2^k}(M(\epsilon))\). Given the nature of the mapping (2), which privileges 0 over \(-1, 1\) over \(-2, \ldots\), an additional
code, $\Gamma'_0$ is used. It corresponds to $k = 0$ but with the mapping (2) modified so that, with the new mapping $M'(\cdot)$, negative and nonnegative values are interleaved in the sequence $-1, 0, -2, 1, \ldots$. This code accounts for values of $\rho$ larger than $\frac{1}{2}$, for which $-1$ is more frequent than $0$, and small $\theta$. For $k > 0$ (larger values of $\theta$) this re-mapping (which actually realizes the transformation $\epsilon \rightarrow - (\epsilon + 1)$) is irrelevant, as the values in the pairs $(0, -1), (1, -2), \ldots$, are given the same code length. Thus, the sequential code selection process in JPEG-LS consists of the selection of a Golomb parameter $k$, and in case $k = 0$, a mapping $M(\cdot)$ or $M'(\cdot)$. Notice that this sub-family of codes, which we denote $\mathcal{C}$, represents a specific compression-complexity trade-off.

To conclude the coding then, we need to select the specific code from this sub-family. The explicit computation method, as opposed to an exhaustive search for the best code, is used in JPEG-LS for determining the optimal code in the sub-family $\mathcal{C}$, based on the sufficient statistics $S_t$ and $N_t$ for the parameters $\theta$ and $\rho$, given by

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

and

$$N_t = \sum_{i=1}^{t} u(\epsilon_i).$$

Here, the indicator function $u(\epsilon)$ takes the value 1 whenever $\epsilon < 0$, and 0 otherwise. Clearly, $N_t$ is the total number of negative samples in $\epsilon^t$, and $S_t + N_t$ is the accumulated sum of absolute values. An advantage of the explicit strategy is that it depends only on $S_t$ and $N_t$, as opposed to the exhaustive search, which would require one counter for each code in $\mathcal{C}$ (per context). Moreover, using these sufficient statistics, rather than the variance or the sum $F_0$, results in a surprisingly simple calculation for $k$ and, in case $k = 0$, also for the mapping. The adaptive selection of the code is based on results proved in [9]. 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 $B$ (bias) and $N$ (total context appearances) related to the bias correction in Section 2.2.3. Then, the following procedure is implemented:

a. Compute $k$ as

$$k = \min\{k' \mid 2^{k'} N \geq A\}.$$  \hspace{1cm} (3)\]

b. If $k > 0$, choose code $\Gamma_k$. Otherwise, if $k = 0$ and $2B \leq -N$, choose code $\Gamma_0$. Otherwise, choose code $\Gamma'_0$.

In software, $k$ can be computed by the C programming language "one-liner"

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

Hardware implementations are equally trivial.

To avoid expansions for single samples, GPO2 codes are modified in JPEG-LS to limit the code length per sample. In addition, to enhance adaptation to non-stationarity of image statistics, JPEG-LS periodically resets all the counters.
2.4 Embedded alphabet extension

Golomb codes, being subsets of the class of Huffman coding (as opposed to arithmetic coding) have a problem of redundancy (i.e., excess code length over the entropy) for contexts representing smooth regions, which have peaked distributions as a prediction residual of 0 is very likely. This is due to its fundamental limitation of producing at least one code bit per encoding. JPEG-LS address the problem of redundancy by embedding an alphabet extension into the context conditioning. Specifically, the encoder enters a “run” mode when a context with \(a = b = c = d\) is detected, as this indicates a flat region. Since the central region of quantization for the gradients \(g_1, g_2, g_3\) is the singleton \(\{0\}\), the run condition is easily detected in the process of context quantization by checking for \([q_1, q_2, q_3] = [0, 0, 0]\). Once in run mode, a run of the symbol \(a\) is expected, and the run length (which may be zero) is encoded. Notice that once in run mode, the context is not checked and some of the samples forming the run may occur in contexts other than \([0, 0, 0]\). When the run is broken by a non-matching sample \(x\), the encoder goes into a “run interruption” state, where the difference \(\epsilon = x - b\) is encoded (here, \(b\) corresponds to the value of the sample above the non-matching sample). 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. Images in which the run mode is exercised very frequently are usually compressed/decompressed much faster than the average.

For the specific form of encoding runs in JPEG-LS, which is also based on Golomb codes, see the extended report on JPEG-LS.

2.5 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, \(\delta\). The basic technique employed for achieving this goal is the traditional DPCM loop [11], where the prediction residual (after correction and possible sign reversion, but before modulo reduction) is quantized into quantization bins of size \(2\delta + 1\), with reproduction at the center of the interval. Quantization of a prediction residual \(\epsilon\) is performed by integer division, according to

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

Since \(\delta\) usually takes one of a few small integer values, the integer division in (4) can be performed efficiently both in hardware [12] and software.

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 \(a, b, c, d\) 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 \(\delta\) can be the result of quantization errors. Moreover, once in run mode, the encoder checks for runs within a tolerance of \(\delta\), while reproducing the value of the reconstructed sample at \(a\). Consequently, the 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 $2\delta$. Consequently, the default thresholds for gradient quantization are scaled accordingly.

The reduction of the quantized prediction residual is done modulo $\alpha'$, where

$$\alpha' = \left\lfloor \frac{\alpha + 4\delta}{2\delta + 1} \right\rfloor,$$

into the range $[-\lfloor \alpha'/2 \rfloor, \lfloor \alpha'/2 \rfloor - 1]$. 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 $[-\epsilon, \alpha'(2\delta+1)-1-\delta]$, 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, $\alpha'$ replaces $\alpha$ in the definition of the limited-length Golomb coding procedure, accordingly affecting the precise computation of all the counters [2].

For a discussion on Multi-component images, Palettes and Sample Mapping and the JPEG-LS Bitstream Structure, please refer to [2].

3 Compression of elevation data

This section describes the results on compression of high resolution digital elevation data. We report the different techniques used to compress elevation data based on spatial resolution, bits per pixel and range of pixel values. Using the number of bits per pixel of the original image as a criteria for classification we have three main types of images, as discussed below. For each of these classes we describe 3 compression approaches:

a. Compressing the whole image as is, both in lossless and near-lossless mode.

b. Partitioning the image into smaller blocks (128 x 128 pixels). This allows semi-random access to different regions of the image as well as local adaptation of the (few) JPEG-LS parameters.

c. Compressing a “slope” image. This is for applications such as helicopter landing, where the terrain slope is a critical factor.

In describing each of these approaches, we provide the results obtained and give methods to optimize the compression ratio. The advantages of a particular approach, and the tradeoff which it entails, will also be discussed.

3.1 Compressing the whole image as is

3.1.1 Images with 16 bits per pixel or less

The images used for testing in this section are in DEM (Digital Elevation Model) format and were obtained from the USGS site http://edcftp.cr.usgs.gov/pub/data/DEM/250. Another available format is the SDTS (Spatial Data Transfer Standard). Within DEM, there are images with scales of 1:24000, 1:100000 and 1:250000. The latter, having the highest resolution, were used for the tests described below. Specifications of the DEM format are available at
<table>
<thead>
<tr>
<th>Image</th>
<th>JPEG-LS CR (default T1,T2,T3)</th>
<th>Effective CR (default T1,T2,T3)</th>
<th>Optimized JPEG-LS CR</th>
<th>Optimized Effective CR</th>
<th>Optimum T1,T2,T3</th>
<th>Pixel value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>richmond-e</td>
<td>45.15</td>
<td>154.01</td>
<td>46.87</td>
<td>159.89</td>
<td>17,17,17</td>
<td>5-46</td>
</tr>
<tr>
<td>grand-rapids-e</td>
<td>13.76</td>
<td>46.99</td>
<td>13.82</td>
<td>47.13</td>
<td>4,4,4</td>
<td>192-381</td>
</tr>
<tr>
<td>dallas-e</td>
<td>11.69</td>
<td>39.86</td>
<td>11.79</td>
<td>40.22</td>
<td>3,3,3</td>
<td>73-262</td>
</tr>
<tr>
<td>puerto-rico-c</td>
<td>9.75</td>
<td>33.27</td>
<td>9.79</td>
<td>33.40</td>
<td>5,3,25</td>
<td>5-680</td>
</tr>
<tr>
<td>denver-w</td>
<td>4.12</td>
<td>14.07</td>
<td>4.36</td>
<td>14.89</td>
<td>7,13,33</td>
<td>1557-4350</td>
</tr>
</tbody>
</table>

Table 1: Compression results on 1:250000 DEM data

http://edcwww.cr.usgs.gov/glis/hyper/guide/usgs_dem.supplement#typea. When DEM data is read into a raster format, the elevations are merely numbers in an array of integers (specifically short integers). As described in 2.2.2, JPEG-LS has default quantization regions for context determination. These are obtained by assigning $T_1 = 3$, $T_2 = 7$ and $T_3 = 21$ as the default quantization level thresholds. Equalizing any two or all of the values collapses the quantization regions and reduces the effective number of contexts available. This might result in a significant improvement in compression ratio for small images such as 1:250000 DEMs (as well as for $128 \times 128$ blocks). Typically, these images are of size 9.81MB in their DEM data format. On converting them to a raster format (essentially a 2-D array of short integers) this size reduces to 2.88 MB. This file is then compressed using JPEG-LS. The “effective compression ratio” entered in Table 1, and henceforth, is calculated from the size of the original DEM file.

As observed in Table 1, the compression ratios depend on the pixel values range. The first image has a small range of pixel values and the compression is improved by reducing the number of contexts. The second and third images also have a reasonably small range and compression improves marginally when $T_1, T_2$, and $T_3$ are equalized. Images 4 and 5 however have a larger range and benefit from having a larger number of contexts.

Figure 4 shows compression ratios for a large number of DEM images. (The horizontal axis is just the image number.) JPEG-LS applied to images of very flat terrain produces extremely high compression ratios (1000 and more). These high values would produce a distorted average compression ratio (mean is much larger than the median), which will not be indicative of the true performance of JPEG-LS algorithm. Hence they have been omitted from the above data set. The average compression ratio for the set is 14.23. Average effective compression ratio is 48.54.

### 3.1.2 Images with 16-32 bits per pixel

These are very high resolution images in DTM (Digital Terrain Model) format with 3m post spacing. 3m resolution DTM data consists of points described by 3 coordinates: x-coordinate (easting), y-coordinate (northing) and z-coordinate (elevation). The values are stored in double float format (64 bits per pixel) in all DTM files. However, for this class of images, the actual number of bits required to completely represent the z-coordinate (elevation) is far smaller than 64. For example, all elevations in the example “cososgeo3.asc” could be represented using 24 bits.

JPEG-LS can directly compress only (at most) 16-bit images. So, each 24-bit elevation in cososgeo3.asc was split into a 16-bit value and a 8-bit value and each was compressed separately. The results appear in Table 2. A large saving is realized even prior to compression, when the double
Figure 4: Compression Ratios for a large set of DEM Images

<table>
<thead>
<tr>
<th>Method</th>
<th>CR for Upper 16 bits</th>
<th>CR for Lower 8 bits</th>
<th>Total CR</th>
<th>Effective CR</th>
<th>Maximum Error (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Lossless</td>
<td>4.63</td>
<td>1.06</td>
<td>2.18</td>
<td>7.07</td>
<td>0</td>
</tr>
<tr>
<td>Upper 16 Lossless</td>
<td>4.63</td>
<td>1.65</td>
<td>2.89</td>
<td>9.35</td>
<td>0.0003</td>
</tr>
<tr>
<td>Lower 8 NL = 3</td>
<td>4.63</td>
<td>1.88</td>
<td>3.11</td>
<td>10.07</td>
<td>0.0005</td>
</tr>
<tr>
<td>Upper 16 Lossless</td>
<td>4.63</td>
<td>1.65</td>
<td>3.11</td>
<td>10.07</td>
<td>0.0005</td>
</tr>
<tr>
<td>Lower 8 NL = 5</td>
<td>7.02</td>
<td>1.65</td>
<td>3.37</td>
<td>10.90</td>
<td>0.0515</td>
</tr>
<tr>
<td>Lower 8 NL = 1</td>
<td>7.02</td>
<td>1.65</td>
<td>3.37</td>
<td>10.90</td>
<td>0.0515</td>
</tr>
<tr>
<td>Lower 8 NL = 3</td>
<td>7.02</td>
<td>1.65</td>
<td>3.37</td>
<td>10.90</td>
<td>0.0515</td>
</tr>
<tr>
<td>Lower 8 Lossless</td>
<td>4.63</td>
<td>—</td>
<td>6.94</td>
<td>22.46</td>
<td>0.0255</td>
</tr>
<tr>
<td>Lower 8 neglect</td>
<td>4.63</td>
<td>—</td>
<td>6.94</td>
<td>22.46</td>
<td>0.0255</td>
</tr>
</tbody>
</table>

FOR 12-12 SPLIT

<table>
<thead>
<tr>
<th>Method</th>
<th>CR for Upper 12 bits</th>
<th>CR for Lower 12 bits</th>
<th>Total CR</th>
<th>Effective CR</th>
<th>Maximum Error (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Lossless</td>
<td>11.03</td>
<td>1.36</td>
<td>2.42</td>
<td>5.87</td>
<td>0</td>
</tr>
<tr>
<td>Upper 12 Lossless</td>
<td>11.03</td>
<td>1.77</td>
<td>3.05</td>
<td>7.40</td>
<td>0.0003</td>
</tr>
<tr>
<td>Lower 12 NL = 3</td>
<td>11.03</td>
<td>1.77</td>
<td>3.05</td>
<td>7.40</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

[CR = Compression Ratio, NL = Near-lossless parameter]
[Original data size = 5.37 MB, DTM file size = 19.35 MB]
[Best Compressed File-size (Lossless) = 0.76 MB]

Table 2: Compression results on cosogeo3.asc DTM data
<table>
<thead>
<tr>
<th>Part of Elevation Data</th>
<th>Entirely Lossless</th>
<th>NL=1 for MID1 Neglect LO</th>
<th>NL=3 for MID1 Neglect LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HI</td>
<td>27.46</td>
<td>27.46</td>
<td>27.46</td>
</tr>
<tr>
<td>MID2</td>
<td>3.29</td>
<td>3.29</td>
<td>3.29</td>
</tr>
<tr>
<td>MID1</td>
<td>1.10</td>
<td>1.38</td>
<td>1.71</td>
</tr>
<tr>
<td>LO</td>
<td>5.47</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MAX ERROR (meters)</td>
<td>0</td>
<td>0.0003</td>
<td>0.0008</td>
</tr>
<tr>
<td>Total Compression</td>
<td>2.79</td>
<td>3.76</td>
<td>4.33</td>
</tr>
<tr>
<td>Effective Compression</td>
<td>7.32</td>
<td>9.86</td>
<td>11.35</td>
</tr>
</tbody>
</table>

[CR = Compression Ratio, NL = Near-lossless parameter]  
[Original data size = 61.94 MB, DTM file size = 226 MB]  
[Best Compressed File-size (Lossless) = 9.43 MB]

Table 3: Compression results on swath3.asc DTM data

data is converted to the aforementioned 16-8 split.

Table 2 also shows the results when the elevation data was split into two groups of 12 bits. The compression ratio is superior (as expected, because the upper 12 bits show very little variation for adjacent pixels, and hence can be compressed better than the upper 16 bits). However, the effective compression ratio is less, because we now have to use (16-bit) integers for both parts.

A different approach was used for the example “swath3.asc” which had a maximum of 25 bits per pixel. Here the split used was 8-8-8-1, the reasons being:

a. The LSB could be neglected entirely, if tolerable, or suited to the application.

b. The most significant byte varies little from pixel to pixel. So, the effective compression for the upper 16 bits was more if bits 18-25 were compressed separately from bits 10-17, as opposed to compressing bits 10-25 at once.

The larger size of this data set allowed the JPEG-LS adaptive predictor to train itself. This, along with point (b) above, is responsible for improving the compression ratio. (2.79 for swath3.asc as compared with 2.18 for cosogeo3.asc; both files have the same spatial resolution). Results appear in Table 3.

3.1.3 Images with 32 or more bits per pixel

These images are also in DTM format but with 10m post spacing. Thus they have lower resolution, but the pixels now take up more bits (Up to 48). Note that the numbers do not increase in magnitude, but only in precision (i.e. the number of digits after the decimal point is increased). These were split as 16-16-16 and 8-8-8-8-8-8. Due to the larger post spacing, the least significant bits in adjacent pixels are uncorrelated. The compression performance is good only for the upper 16 bits and there is little or no compression for the other packets. See Table 4. Thus totally lossless JPEG-LS, which preserves the elevation (accurate to $10^{-11}$ ft), does not provide appreciable compression for these images. A considerable saving is realized if, for applications which don’t require this accuracy, the lower 16 bits are neglected totally. Even this drastic omission in swath10.asc gives a maximum pixel error of 0.0000065536 m. Lossless compression results for the 16-16-16 split are slightly better than those for 8-8-8-8-8-8 split. This indicates that, if there are 16 bits of almost
FOR 8-8-8-8-8 SPLIT

<table>
<thead>
<tr>
<th>Part of Elevation Data</th>
<th>Entirely Lossless</th>
<th>NL=1 for LO</th>
<th>NL=3 for LO</th>
<th>Neglect LO and MID1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HI (8 bits)</td>
<td>29.32</td>
<td>29.32</td>
<td>29.32</td>
<td>29.32</td>
</tr>
<tr>
<td>MID4 (8 bits)</td>
<td>3.47</td>
<td>3.47</td>
<td>3.47</td>
<td>3.47</td>
</tr>
<tr>
<td>MID3 (8 bits)</td>
<td>1.07</td>
<td>1.07</td>
<td>1.07</td>
<td>1.07</td>
</tr>
<tr>
<td>MID2 (8 bits)</td>
<td>1.13</td>
<td>1.13</td>
<td>1.13</td>
<td>1.13</td>
</tr>
<tr>
<td>MID1 (8 bits)</td>
<td>1.05</td>
<td>1.05</td>
<td>1.05</td>
<td>—</td>
</tr>
<tr>
<td>LO (8 bits)</td>
<td>1.05</td>
<td>1.30</td>
<td>1.55</td>
<td>—</td>
</tr>
<tr>
<td>MAX ERROR (meters)</td>
<td>0</td>
<td>$10^{-11}$</td>
<td>$3 \times 10^{-11}$</td>
<td>$6.5536 \times 10^{-6}$</td>
</tr>
<tr>
<td>Total Compression</td>
<td>1.48</td>
<td>1.55</td>
<td>1.60</td>
<td>2.80</td>
</tr>
<tr>
<td>Effective Compression</td>
<td>3.73</td>
<td>3.91</td>
<td>4.03</td>
<td>7.05</td>
</tr>
</tbody>
</table>

FOR 16-16-16 SPLIT

<table>
<thead>
<tr>
<th>Part of Elevation Data</th>
<th>Entirely Lossless</th>
<th>NL=1 for LO</th>
<th>NL=3 for LO</th>
<th>Neglect LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>HI (16 bits)</td>
<td>6.01</td>
<td>6.01</td>
<td>6.01</td>
<td>6.01</td>
</tr>
<tr>
<td>MID (16 bits)</td>
<td>1.10</td>
<td>1.10</td>
<td>1.10</td>
<td>1.10</td>
</tr>
<tr>
<td>LO (16 bits)</td>
<td>1.08</td>
<td>1.20</td>
<td>1.30</td>
<td>—</td>
</tr>
<tr>
<td>MAX ERROR (meters)</td>
<td>0</td>
<td>$10^{-11}$</td>
<td>$3 \times 10^{-11}$</td>
<td>$6.5536 \times 10^{-6}$</td>
</tr>
<tr>
<td>Total Compression</td>
<td>1.49</td>
<td>1.57</td>
<td>1.62</td>
<td>2.77</td>
</tr>
<tr>
<td>Effective Compression</td>
<td>3.76</td>
<td>3.94</td>
<td>4.08</td>
<td>6.98</td>
</tr>
</tbody>
</table>

[CR = Compression Ratio, NL = Near-lossless parameter]
[Original data size = 9.54 MB, DTM file size = 28.7 MB]
[Best Compressed File-size (Lossless) = 2.54 MB]

Table 4: Compression results on swath10.asc DTM data [CR = Compression ratio][NL = Near Lossless Parameter]

Uncorrelated data, then we are better off compressing them together instead of splitting them into groups of 8 bits. But when the 16 lower bits are sacrificed, the 8-8-8-8-8-8 method wins, for the reasons discussed in section 3.1.2. (Note the very high compression ratio for the upper 8 bits). From Table 4, we surmise that the best possible split is 8-8-16-16. This yields a total lossless CR of 1.50 and an effective lossless CR of 3.77.

3.2 Partition of image followed by compression

Partition of the image into tiles of of 64 × 64 or 128 × 128 pixels permits a semi-random access to the data. In addition, the near-lossless parameter can be independently selected per tile, and while some tiles can be compressed in lossless mode, others can be compressed in the near-lossless mode and with different error tolerance.

We now investigate the effects of tiling in the compression of elevation data. On one hand, tiling might improve the compression ratio when the JPEG-LS parameters can be adapted to each particular tile. On the other hand, the compression ratio for the total image will be negatively affected if the adaptive predictor has significantly less data to learn and adapt.
<table>
<thead>
<tr>
<th>Method</th>
<th>(T1,T2,T3)</th>
<th>Avg. CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a whole</td>
<td>3.7,21</td>
<td>3.58</td>
</tr>
<tr>
<td>81 segments</td>
<td>3.7,21</td>
<td>2.84</td>
</tr>
<tr>
<td>81 segments</td>
<td>7.7,7</td>
<td>3.43</td>
</tr>
</tbody>
</table>

[Each block is 128 x 128 pixels]

Table 5: Lossless Compression results on mariposa-w

<table>
<thead>
<tr>
<th></th>
<th>2.65</th>
<th>2.10</th>
<th>2.05</th>
<th>2.11</th>
<th>2.14</th>
<th>2.54</th>
<th></th>
<th>2.64</th>
<th>2.10</th>
<th>2.08</th>
<th>2.14</th>
<th>2.11</th>
<th>2.48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.36</td>
<td>2.00</td>
<td>1.93</td>
<td>1.97</td>
<td>1.94</td>
<td>2.43</td>
<td></td>
<td>2.27</td>
<td>1.95</td>
<td>1.90</td>
<td>1.94</td>
<td>1.93</td>
<td>2.45</td>
</tr>
<tr>
<td></td>
<td>2.25</td>
<td>2.07</td>
<td>2.08</td>
<td>2.09</td>
<td>1.97</td>
<td>2.31</td>
<td></td>
<td>2.18</td>
<td>2.02</td>
<td>2.08</td>
<td>2.07</td>
<td>1.97</td>
<td>2.30</td>
</tr>
<tr>
<td></td>
<td>2.02</td>
<td>2.15</td>
<td>2.15</td>
<td>2.12</td>
<td>2.07</td>
<td>2.43</td>
<td></td>
<td>2.02</td>
<td>2.13</td>
<td>2.15</td>
<td>2.01</td>
<td>2.03</td>
<td>2.46</td>
</tr>
<tr>
<td></td>
<td>2.02</td>
<td>2.12</td>
<td>2.13</td>
<td>1.90</td>
<td>2.01</td>
<td>2.38</td>
<td></td>
<td>1.98</td>
<td>2.15</td>
<td>2.13</td>
<td>1.83</td>
<td>2.01</td>
<td>2.40</td>
</tr>
<tr>
<td></td>
<td>3.21</td>
<td>2.55</td>
<td>2.53</td>
<td>2.49</td>
<td>2.51</td>
<td>3.32</td>
<td>(T1,T2,T3)=(3.7,21)</td>
<td>3.08</td>
<td>2.49</td>
<td>2.53</td>
<td>2.54</td>
<td>2.50</td>
<td>3.33</td>
</tr>
</tbody>
</table>

Table 6: Compression on segments of cosgeo3.asc DTM

3.2.1 Images with less than 16 bits per pixel

Table 5 shows the performance of JPEG-LS for mariposa-w, when the image is partitioned into blocks of 128 x 128 pixels, for two different triads of T1,T2 and T3. As expected, the compression ratio suffers due to the tiling operation. However, after tiling, it is possible to equalize T1,T2,T3 as shown, to obtain a significantly higher average compression ratio. It is thus clear that reducing the number of contexts results in better compression of an individual tile.

3.2.2 Images with 16-32 bits per pixel

Table 6 (left) shows compression ratios for 128 x 128 pixel tiles of the cosgeo3.asc (DTM) image. The compression ratio for cosgeo3.asc was 2.18 and the average ratio for the tiled version is 2.096 for the default triad (T1,T2,T3) = (3,7,21). The reduction is due to the tiling operation as discussed above.

Table 6 (right) contains the compression ratios for (T1,T2,T3) = (4,4,4). As expected, compression ratio increases for some segments and decreases for others. This suggests that different segments have different optimum triads, and the segment of interest can be compressed more if its optimum triad is found. Equalizing T1, T2, and T3 is profitable while compressing the upper 16 (or 12) bits. These do not vary appreciably from pixel to pixel and a smaller number of contexts suffices. The lower 8 (or 12) bits vary significantly across pixels and hence require a different optimum triad, wherein T1, T2 and T3 are not necessarily equal.

A better average overall compression ratio than that obtained in the above two tables will thus result if (4,4,4) is used to compress the upper 16 bits and (3,7,21) is used for the lower 8 bits.

3.3 Compression of a “slope” image

Preserving the terrain slope, not its absolute elevation, it is of importance for operations such as helicopter landing. As a variation, JPEG-LS was tested on a “slope” image. This image was constructed out of the original cosgeo3.asc using the following relations:
**FOR 16-8 SPLIT**

<table>
<thead>
<tr>
<th>Method</th>
<th>CR for Upper 16 bits</th>
<th>CR for Lower 8 bits</th>
<th>Total CR</th>
<th>Effective CR</th>
<th>Maximum Error (meters/meter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Lossless</td>
<td>1.48</td>
<td>0.98</td>
<td>1.26</td>
<td>4.09</td>
<td>0</td>
</tr>
<tr>
<td>Upper 16 Lossless</td>
<td>1.48</td>
<td>1.47</td>
<td>1.47</td>
<td>4.76</td>
<td>$10^{-7}$</td>
</tr>
<tr>
<td>Lower 8 NL = 3</td>
<td>1.48</td>
<td>1.66</td>
<td>1.54</td>
<td>4.96</td>
<td>$1.667 \times 10^{-7}$</td>
</tr>
<tr>
<td>Upper 16 Lossless</td>
<td>1.48</td>
<td>—</td>
<td>2.22</td>
<td>7.17</td>
<td>$8.5 \times 10^{-6}$</td>
</tr>
<tr>
<td>Lower 8 Lossless</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower 8 neglect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**0.1234 FORMAT RETAINING SLOPE TO 4 DECIMAL PLACES**

<table>
<thead>
<tr>
<th>Method</th>
<th>Total CR</th>
<th>Effective CR</th>
<th>Maximum Error (meters/meter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totally Lossless</td>
<td>1.88</td>
<td>9.11</td>
<td>0</td>
</tr>
<tr>
<td>NL = 3</td>
<td>2.75</td>
<td>13.34</td>
<td>$10^{-4}$</td>
</tr>
</tbody>
</table>

**Table 7**: Compression results on cosogeo3.asc Slope data

\[
I_x(i, j) = \frac{(I(i + 1, j) - I(i - 1, j))}{2}
\]

\[
I_y(i, j) = \frac{(I(i, j + 1) - I(i, j - 1))}{2}
\]

\[
Gradient(I) = \sqrt{I_x^2 + I_y^2}
\]

$I_x$ and $I_y$ are the derivatives in the horizontal and vertical directions. To avoid boundary problems, we neglect the first and last rows and columns (from an $N \times M$ image we compute an $(N-2) \times (M-2)$ size image). The square root operation will result in very fine precision, but we persist with the same number of digits after the decimal points, as that in the original image. (In this case, 4). The compression ratio then is superior to that obtained by compressing the image as is. Table 7 shows the results.

Once again, the motivation for this approach comes from landing applications wherein slope of the terrain is more important than the absolute elevation. We have the following options:

a. Transmit a compressed image as in section 3.1.2 and perform slope calculations after reception.

b. Computing the slope image and compressing it prior to transmission, as above

### 3.4 Near lossless mode

Section 2.5 describes the near lossless mode in JPEG-LS. If we can tolerate a fixed error in some or all the pixels of an image, then the near lossless mode can be use to advantage, especially for
images with high elevation resolution. (Adjacent pixel values are correlated and their difference is small.) For example, the images cosgeo3.asc and swath3.asc in the DTM format have elevation resolution of 0.0001 m. Tables 2 and 3 show how a higher compression ratio is obtainable if near-lossless JPEG-LS is used to compress the least significant byte portion. For eg. when the lossy parameter \( \delta = 1 \), the pixels of cosgeo3.asc will be offset by at most 0.0001 m. For landing-related applications, this translates into a maximum slope error of \( 6.667 \times 10^{-5} \) meters/meter for 3m DTM data. Even with these insignificant errors, compression ratios are considerably improved.

Note that controlled-lossy results are also obtained when entire low significant bytes of data are neglected for large dynamic range images. This is also reported in the tables, where compression per byte (or word) is given. Also in this case the elevation error is considerably small and the compression ratio is significantly improved.

## 4 Conclusions

We have studied the application of JPEG-LS for the compression of elevation data. Using JPEG-LS has three main advantages: 1- It is a low complexity, high compression ratio, standard; 2- It permits lossless compression, which is fundamental for applications such as storage; 3- It provides a controlled lossy mode that permits the user to dictate a maximal error in the elevation (slope), thereby improving the compression ratio while guaranteeing performance.

From the investigation, we have also concluded the following design decisions:

a. The compression is marginally better if the upper 16 bits are first split into 2 bytes each and then separately compressed, as opposed to direct compression of the upper 16 bits.

b. When lower and middle bits of images are minimally correlated or uncorrelated, it is better to compress them in groups of 16 bits rather than groups of 8 bits. This is particularly helpful for 10m resolution DTM data.

c. It is beneficial if the least significant bits are in as small a group as possible (eg. 25 bit data can be split as 8-8-8-1). If permissible, this segment can be totally neglected resulting in higher effective compression. This applies especially to 3m resolution DTM data.

d. For images where a region of concentration can be defined, it is possible to partition the image into tiles of 128 × 128 pixels. The region of interest can be compressed with lossless JPEG-LS and the reduced pixel value range makes the compression ratio responsive to the quantization region thresholds T1,T2 and T3. For one optimal choice of the triad which collapses the quantization regions and reduces the number of contexts, better compression is obtained. The remaining segments can either be neglected or be compressed using near-lossless JPEG-LS, so that, in return for a controlled loss, improved compression is obtained.

e. The compression capability of JPEG-LS improves with the resolution of the the DEM data. Higher resolution provides a better correlation between adjacent pixel values and therefore a better performance for the JPEG-LS predictor.

f. The larger the image, the better the predictor in JPEG-LS is trained. Hence the compression is marginally better for a larger image than for a smaller image with the same spatial resolution.

g. Segmentation allows better compression of an individual segment but the overall compression ratio suffers due to point (f) above. (JPEG-LS would now compress small blocks of 128 × 128 pixels instead of one large image.)
To conclude, having in mind the immediate availability of JPEG-LS, the results here reported strongly support its adoption for the compression of elevation data for a number of applications, e.g., storage. This doesn’t mean that JPEG-LS provides a complete solution to the problem, and indeed the development of compression algorithms tailored to elevation data is still an open problem.

References


