Compressing Digital Elevation Models based on Run Length Encoding Approach

Giovanni Guzman and Rolando Quintero

Centre for Computer Research, National Polytechnic Institute, Mexico City, Mexico
{jguzmanl, quintero}@cic.ipn.mx
http://geo.cic.ipn.mx, http://www.cic.ipn.mx

Abstract. Up-to-date, some related algorithms to compress digital elevation models (DEM) or high-resolution DEMs use wavelet and JPEG-LS encoding approaches to generate compressed DEM files with good compression factor. However, to access the original data (elevations), it is necessary to apply a decompression approach to retrieve the contour lines. In this paper, we propose an algorithm oriented to compress a digital elevation model, which is based on a sequence of binary images to encode them using RLE compression technique, according to a specific height (contour lines). The sequence is compressed by applying a binary compressor. The main goal of our algorithm is that the specific parameters of the DEM (altitudes and contours lines) can be obtained without using a decompression stage, because the information is directly read from the compressed DEM. Our method reduces the amount of needed space to store DEM geo-images.

1 Introduction

The term digital elevation model or DEM is frequently used to refer to any digital representation of a topographic surface, however, most often it is used to refer specifically to a raster or regular grid of spot heights. Digital terrain model or DTM may actually be a more generic term for any digital representation of a topographic surface, but it is not so widely used [1]. DEM has gained popularity in applications to determine terrain's attributes, such as elevation at any point, slope and aspect, finding features on the terrain. The most important features are: drainage basins and watersheds, drainage networks and channels, peaks and pits and other landforms and, for modeling hydrologic functions, energy flux and forest fire [2].

On the other hand, the main purpose of a compression algorithm is to reduce the amount of information needed to describe the original data. To retrieve the original data we apply a process commonly named decompression. If the original information is fully-retrieved without any modification or alteration, then we can talk about a loss-less compression algorithm. Also, it is possible to have compression algorithms with loss of information [9]. Many compression approaches first proceed to inspect the input image and try to detect different types of redundancy: statistical, psycho-visual or by correlation.

© M. Moreno-Ibarra, S. Levachkine (Eds.)
Semantics in Geoinformatics Applications
Research in Computing Science 25, 2006, pp. 73-84

A digital elevation model requires a huge amount of data, up to 100 or more megabytes of storage. Up-to-date, compression techniques are used to compress Digital Elevations Models with a high compression coefficient, in [4], [5] and [6] the authors propose to use wavelets and JPEG-LS encoding approaches. However, if the user requires to access to the file or to obtain certain information about the image, it is necessary to decompress all DEM information.

In this paper, we propose an algorithm oriented to compress a digital geo-image of a digital elevation model, which is based on a sequence of images with a specific height. The sequence is compressed by applying a binary compressor. The main goal of our algorithm is that the specific parameters of the DEM (altitudes and contours lines) can be obtained without using a decompression stage, because the information is directly read from the compressed DEM.

The rest of this paper is organized as follows: section 2 describes the proposed compression algorithm. Section 3 describes the manipulation of compressed DEM file to obtain both elevations and contours lines, without applying a decompression process. Section 4 presents the results obtained by applying our approach, and section 5 outlines the conclusions and future works.

2 Description of the compression algorithm

Nowadays, there are some types of DEMs such as: 7.5-minute, 15-minute, 2-arc-second, and 1-degree units [7]. For implementation purposes, we have proposed to choose the 1-degree DEM variant.

An important characteristic is that the frequency of the DEM is not high, because the pixels do not present changes in their structure; the value of each pixel is correlated with their neighbor values. Due to this, it is possible to apply this kind of compression. Moreover, this approach is not useful for common images.

Basically, a DEM file is integrated by three types of records, usually called A, B and C. The structure of these records is as follows [1]:

- *Record A* contains information, which defines the general characteristics of DEM, it includes descriptive header information related to the DEM's name, boundaries, units of measurement, minimum and maximum data values, number of type B records and projection parameters. There is only one type A record for each DEM file, and it appears as the first record in the data file.
- *Record B* contains elevation data and associated header information. All type B records of the DEM files are made up data from one-dimensional bands, called points. Therefore, the number of complete points covering the DEM area is the same as the number of type B records in the DEM.
- *Record C* contains statistics on the accuracy of the data in the file.

In a Digital Elevation Model, the altitude describes elevations greater or equal to 0 in a specific area. In consequence, the minimum altitude is 0 meters (sea level) and the maximum is 8,850 meters (Everest mount). In most of the cases, the DEM is displayed by using a 3-D render approach or applying a transformation function to obtain a gray level image. In Fig. 1 we depict an example of DEM file.

To detail the proposed algorithm, it is necessary to give some definitions that are important to describe our method.

The altitude[1] or height associated to one element that belongs to Record B in the DEM, is defined by Eqn. 1.

$$h(x,y)=\alpha\,, \tag{1}$$

To associate all points with same height α_k, we obtain the set defined by Eqn. 2.

$$S_{\alpha_k}=\{(x,y)\neg h(x,y)=k\}\,, \tag{2}$$

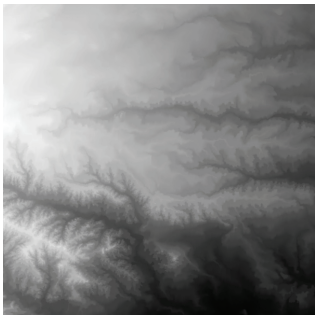


Fig. 1. Example of a DEM image

The number of elements in each S_{α_k} will be denoted by $card(S_{\alpha_k})$. Considering these definitions, the DEM altitudes are described by using Eqn. 3.

$$S=\{S_{\alpha_{\min}}\cup S_{\alpha_2}\cup\ldots\cup S_{\alpha_{\max}}\}, \tag{3}$$

where:

α_{MAX} denotes the maximum height in the original DEM.

Additionally, it is possible to define the join operation, according to Eqn. 4.

$$\bigcup_i^j S_\alpha=\{S_{\alpha_i}\cup S_{\alpha_{i+1}}\cup\ldots\cup S_{\alpha_{j-1}}\cup S_{\alpha_j};i\le m\le j\}\,, \tag{4}$$

To obtain the difference between two sets, we can use the minus operation defined in Eqn. 5.

$$S_{\alpha_a}-S_{\alpha_b}=\{(x,y)\neg(x,y)\in S_{\alpha_a}\ \text{and}\ (x,y)\notin S_{\alpha_b}\}, \tag{5}$$

With these definitions, we can formally state our proposed algorithm.

2.1 Compression approach

In this section, we provide a detailed explanation of the required steps to compress a 1-degree DEM with altitudes expressed as integer values.

Step 1. Records A and C are stored without modification in the output.

[1] We have adopted the convention that elevations in the DEM file are integer values.

Step 2. Find minimum and maximum altitudes $(\alpha_{\min}, \alpha_{\max})$ presented in the DEM and generate the header described in Table 1 in the output.

Table 1. Header description of the compressed DEM

Header of Compressed DEM	Type of variable to store information
Image width (columns)	integer (2 bytes)
Image height (rows)	integer (2 bytes)
α_{min}	integer (2 bytes)
α_{max}	integer (2 bytes)

The process starts with $k = \alpha_{\min+1}$ instead of $\alpha_{\min}$ due to all heights in the image are greater or equal to $\alpha_{\min}$, it is necessary to apply an iterative process from Steps 3 to 5.

Step 3. Determine the digital image associated to join operation, it is necessary to use the following two operations:

1. Compute the join operation composed of all altitudes lower than α_k, that is, inside the range $[0, k-1] \rightarrow i = 0, j = k-1$.
2. Generate a binary image applying the transformation function indicated by Eqn. 6.

$$g_{\alpha_k}(x,y) = \left\{ \begin{array}{l} 1 \quad \text{if } (x,y) \notin \bigcup_{i=\alpha_{\min}}^{k-1} S_{\alpha_i} \\ \\ 0 \quad \text{otherwise} \end{array} \right\}, \tag{6}$$

In Fig. 2 we depict the result of digital images applying a threshold algorithm in Fig. 1, by means of the function described in Eqn. 6. The threshold applied values were $\alpha_k = 450$ and $\alpha_k = 575$.

(a) (b)

Fig. 2. Results of the threshold algorithm

3. If the new binary image is equal to previous altitude, it is necessary to skip the Step 4, and use the value -1 in the encoded DEM file and continue with Step 5.

Step 4. According to the binary image obtained in Step 3, we proceed to use the run length encoding (RLE) method [8] [10]. Basically, RLE is a straightforward way

of encoding data so that it takes less space. It relies on the string being encoded containing runs of the same character. In the RLE approach, it is important to indicate both, the frequency and the intensity encoded, but considering that the resulting image contains only two values (0 or 1), the value or intensity is not required in the final sequence. To determine the encoded image the following steps are applied.

1. By convention, the first frequency denotes the number of continuous zeros in the image.
2. By applying a top-down inspection[2], it is indispensable to count the number of pixels with same intensity and send this value to the encoded array, according to Eqn. 7.

$$value = \begin{Bmatrix} byte(value) \mid 80_{HEX} \text{ if } value < 128 \\ integer(value) \text{ if } value \geq 128 \end{Bmatrix}, \tag{7}$$

where:
byte (value) returns the byte representation of *value*, additionally we establish the most significative byte (MSB) to 1 using the Boolean OR operation.
integer (value) computes the integer representation of *value*[3].

By using this adjustment[4], it is possible to increase the compression factor because in some cases the total number of continuous pixels with same intensity should not be more than 128. To indicate this condition, the MSB of *value* is established to one; otherwise, an integer representation of *value* is stored in the new array.

3. Repeat the previous step until the end of the image is reached.
4. At the beginning of encoded array, we append the total number of encoded characters, which is the number of transitions between 0 and 1 in *g(x, y)*.

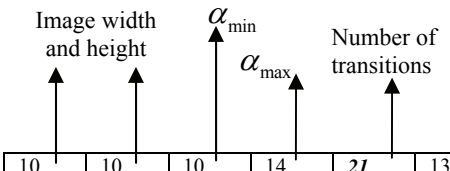
Step 5. Make a unitary increment to *k* variable, and repeat it from Step 1. The stop condition is used when $k > \alpha_{\max}$.

In Figs. 3 and 4, we show the result of the compression approach applied to a small set of DEM elevations. Moreover, the original elevation data are presented in Fig. 3, and Fig. 4 depicts the final result.

10	10	11	11	11	12	12	13	14	13
10	10	11	11	11	12	13	14	14	14
10	10	10	11	11	12	13	14	14	14
10	11	11	11	11	12	13	14	14	14
10	11	12	12	12	12	13	14	14	13
10	11	12	12	13	13	13	13	13	12
10	11	11	12	13	12	12	12	12	12
10	11	11	12	12	12	11	12	11	11
10	11	11	12	12	11	11	11	11	10
10	11	11	12	12	11	10	10	10	10

Fig. 3. Original elevations samples

[2] The inspection can be made in other directions, i.e. bottom-up (right-left and down-up).
[3] In this case, we consider that an integer requires 16 bits (2 bytes) of memory.
[4] For integers altitudes, the MSB always must be zero. Otherwise, it will not distinguish between bytes and integer elevations.



10	10	10	14	***21***	130	136	130	136	131
135	129	137	129	137	129	137	129	137	129
137	129	136	130	133	132	***23***	133	133	133
133	133	133	133	133	130	136	130	136	131
135	131	131	129	129	133	130	136	130	133
15	135	131	134	132	134	132	134	132	134
132	132	133	133	129	163	***11***	136	129	136
131	135	131	135	131	135	130	179		

Fig. 4. Compressed elevations

To represent information of the elevations, we require in the first case, 200 bytes of memory, and in the encoded version we only require 86 bytes.

2.2 Decompression method

The DEM compressed files contain three headers: Records A and C, DEM header, and the total of $\alpha_{max} - \alpha_{min} - 1$ RLE arrays. Retrieving original binary values from any RLE array do not provide the specific altitude at some point. If a point *p(x, y)* of source DEM has an altitude equals to *k*, this point has an intensity of 1 in total of $(k - \alpha_{min})\, g(x, y)$ images.

When we perform the join operation in the range $[\alpha_{min+1}, \alpha_{max}]$ a pixel *p(x, y)* has a 0 intensity if Eqn. 6 is not accomplished $([k+1, \alpha_{max}])$, then the gray-level in *g(x, y)* will be 1 $([\alpha_{min+1}, k])$. In conclusion the elevation of one point *p(x, y)* is determined by using Eqn. 8.

$$h(x, y) = \alpha_{min} + \sum_{i=\alpha_{min+1}}^{\alpha_{max}} g_{\alpha_i}(x, y), \tag{8}$$

In addition, the required steps of decompression stage are the following:

Step 1. Both Records A and C are retrieved directly from compressed file.

Step 2. Process the DEM compressed header, which contains the *width* and *height* of DEM image, as well as the minimum and maximum elevations. Generate an integer *width x height* matrix (referred as DEM image) and establish all the matrix values to minimum elevation.

Apply an iterative process from Steps 3 to 5 for each RLE encoded array.

Step. 3. Obtain the number of transitions in current encoded array (*n*), and remember that the value of *n* has been written before each RLE encoded array. If the value of *n* is equal to -1, then we apply the next steps with the first preceded valid RLE array.

Step 4. Both the absolute pixel position (*position* variable) and the current intensity encoded *(intensity)* are initialized to zero.

Step 5. Read the next byte (b_1) from DEM compressed file. If the most significative byte of this data is -1, then read next byte (b_2) and apply Eqn. 9 to obtain the value (n_0) of continuous encoded[5] intensities. In the compression algorithm, the first n_0 refers to zero intensity. Additionally,

$$n_0 = \begin{Bmatrix} b_2 + \left(b_1 \times \left(2 \hat{} 8\right)\right) \text{ if } MSB(b_1) = 1 \\ b_1 \,\&\, 7F_{HEX} \text{ otherwise} \end{Bmatrix}, \tag{9}$$

If the current intensity is equal to 0, we only increment with the value of n_0, the absolute position *(position=position + n_0)*. In other case, we need to make an n_0-unitary increment both the value of the DEM image at the position *p(x, y)*, and to the absolute position, as described in Eqn. 10.

$$\begin{aligned} x &= \left(position - width \times \lfloor position / width \rfloor \right) \\ y &= \lfloor position / width \rfloor \\ p(x,y) &= p(x,y) + n_0 \\ position &= position + n_0 \end{aligned}, \tag{10}$$

Change the value of intensity if the current value is 0, set it to 1, and to 1 else, repeat Step 5 until process all RLE values.

3 Compressed DEM management

In section 1, we cited the most important characteristics of this compression algorithm. We assume that it is not indispensable to apply a decompression method to obtain information about the elevation from the DEM (read data). Additionally, we can generate the contour layer, which has several map applications related to Geoprocessing area. These two operations are described in the next section.

3.1 Accessing data elevations

In some cases a DEM file requires up to 100 MB storage space. Moreover, some person can require an application that works with 10, 20, or more DEMs. It is possible that any user does not have storage limitations, but in conditions that the use of the lower memory, or the use of the hard disk space is crucial, the DEM compressed file could be useful. To retrieve the elevation at some point *p(x, y)* we cited Eqn. 8, however it requires to process all RLE encoded array. To reduce processing time, we can apply the Eqn. 11.

$$p(x,y) = \alpha_h \neg g_{\alpha_h}(x,y) \neq 0, \left(g_{\alpha_{h+1}}(x,y) = 0 \vee \alpha_h = \alpha_{\max} \right), \tag{11}$$

[5] We suppose that high-part of the integer is stored first.

This equation basically consists of a lineal-search in all RLE encoded arrays, in the cases when the desired elevation is near $\alpha_{min+1,}$ the elevation is rapidly obtained, otherwise all arrays (closer to α_{max}) will be processed. In consequence, we have $\Theta(n/2)$ algorithm, where *n* denotes the total number of bytes to process. To improve this process the *bin-search* can be used. When *bin-search* starts, the total RLE arrays will be equal to *n* (see Eqn. 12).

$$A = \{a_0, a_1, ..., a_n\}, \tag{12}$$

Formally, the binary-search function (BS) is described in Eqn. 13.

$$BS\left(A, k_1, k_2, m, g_{\alpha_m}(x,y)\right) = \begin{cases} \alpha_{\min} + k_1 + 1, \\ \text{if } g_{\alpha_{k_1}}(x,y) = 0, g_{\alpha_m}(x,y) = 1, k_2 = m+1 \\ \\ \alpha_{\max} \ \text{if} g_{\alpha_{k_2}}(x,y) = 1, k_2 = n \\ \\ BS\left(\{a_{k_1} \ldots a_{m'}\}, k_1, m'-1, m', g_{\alpha_{m'}}(x,y)\right), \\ \text{if } g_{\alpha_{m'}}(x,y) = 0 \\ \\ BS\left(\{a_{m'+1} \ldots a_{k_2}\}, m'+1, k_2, m', g_{\alpha_{m'}}(x,y)\right), \\ \text{if } g_{\alpha_{m'}}(x,y) = 1 \end{cases}, \tag{13}$$

where:

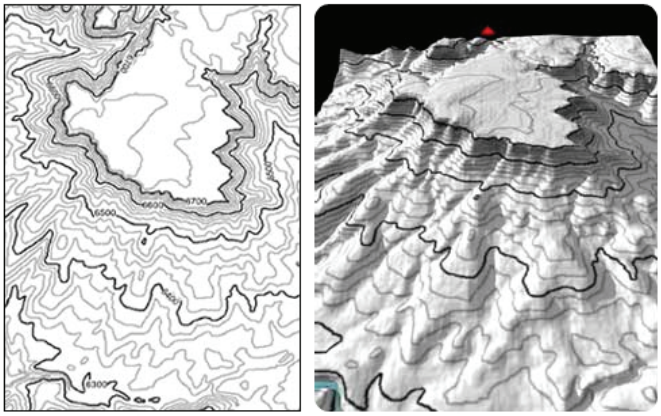
A denotes the RLE arrays to search.
k_1 and k_2 are the number of RLE array $k_1, k_2 \in \{0, 1, \ldots, n\}$.
m is the middle value of previous binary-search.
$g_{\alpha_m}(x,y)$ is the altitude at middle position in previous search.
m' is the new middle value, computed as $m' = \lfloor (k_2 - k_1)/2 \rfloor + k_1$.

In the first execution of the search function $k_1 = 0, k_2 = n, m = 0, g_{\alpha_m}(x,y) = 0$. A special condition is reached when the *k*-RLE array is empty because $g_{\alpha_k}(x,y) = g_{\alpha_{k-1}}(x,y)$, to solve this ambiguity the method searches for the first previous RLE array, which is not empty.

3.2 Generating contours without compression

Contours are lines drawn on a map, which connect points of equal elevation. Contour lines are useful because they allow us to show the shape of the land surface (topography) on a map [3]. The maps depicted in Fig. 5 illustrate an example of

contours. Fig. 5 (a) shows the contour lines for some region and in Fig. 5 (b), a 3-D perspective of same area overlapped with contours is depicted.



(a) (b)

Fig. 5. Contour lines example: (a) Only contours data (b) Contour lines overlapped with 3-D model

The methodology to obtain contour lines form in a Digital Elevation Model is not complicated; some commercial tools to handle cartographic data make this task. However, when we work with DEMs, these systems require all elevation date, but our proposed algorithm generates the contour lines directly form compressed DEM file. In this section we point out this process.

The single required parameter is the altitude interval to sample DEM (Δ), all additional information is available form compressed data.

Step 1. Generate an *N x M* matrix and establish all values to 0, this matrix will represent the contour lines image *f(x, y)*. The size of the matrix is obtained from DEM compressed header.

Step 2. Take the first RLE encoded array (i.e. encoded data of $g_{\alpha_{\min+1}}$). Obtain original binary image by using Eqn. 8. If the array is empty, then apply the same criteria as we defined it in previous sections. With original binary image, it is necessary to obtain its negated version.

Step 3. Compute the 8-connected contour of binary image. Strictly, an object pixel *p(x, y)* is part of contour, if the number of 8-neighbors with intensity equal to zero (background pixels) is at least equal to 1.

Step 4. Remove redundant contour pixels. All pixels that accomplish one of the masks defined in Fig. 6 are removed.

X	1	X
1	1	X
X	X	0

X	1	X
X	1	1
0	X	X

0	X	X
X	1	1
X	1	X

X	X	0
1	1	X
X	1	X

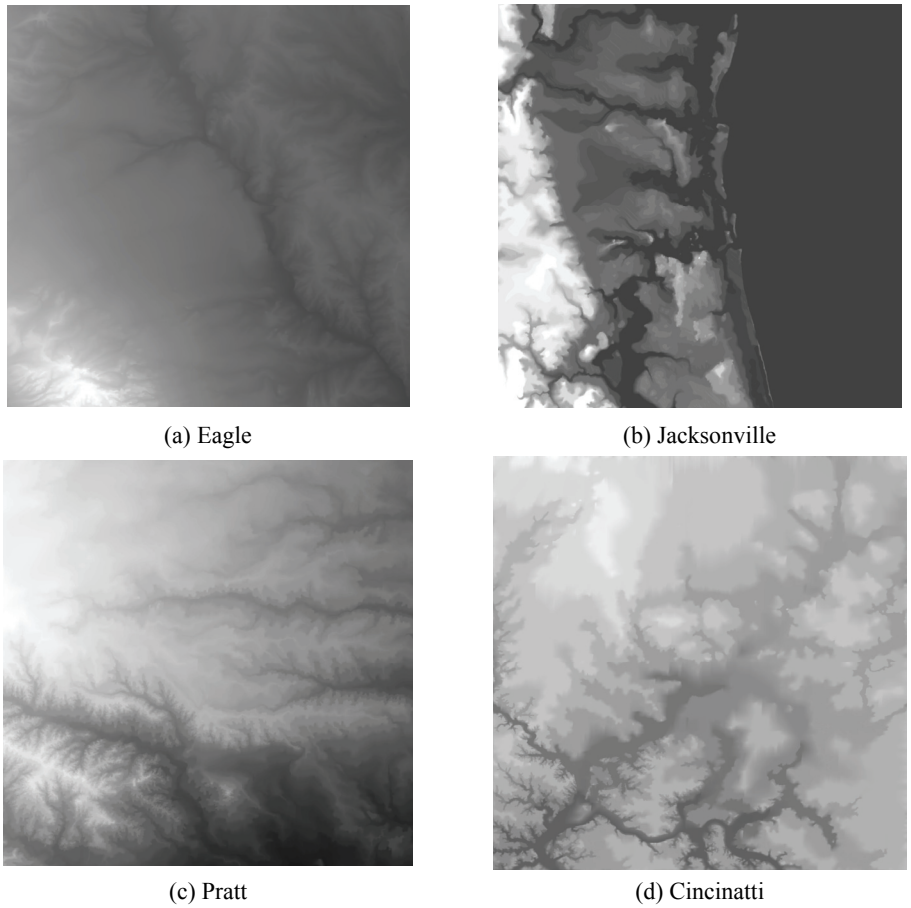
Fig. 6. Set of redundant pixels in detection masks

Step 5. Establish value of one at coordinates *(x, y)* in contour lines image, if the pixel *p(x, y)* is a member of non-redundant contour set obtained in the Step 4.

Step 6. Process each remained encoded array by applying from the Step 4.

4 Tests and Results

In this section, we show in a detailed way, the obtained results with our proposed approach, and compare it with commercial WinZip program (it uses LZW coding method); in this case the standard-compression option has been applied. The DEMs used for the tests are displayed in Fig. 7, and all DEM sources have the same spatial resolution (1,200 x 1,200 pixels) and their sizes are 10MB approximately. In Table 2 the numerical information about the obtained results appears.



(a) Eagle (b) Jacksonville

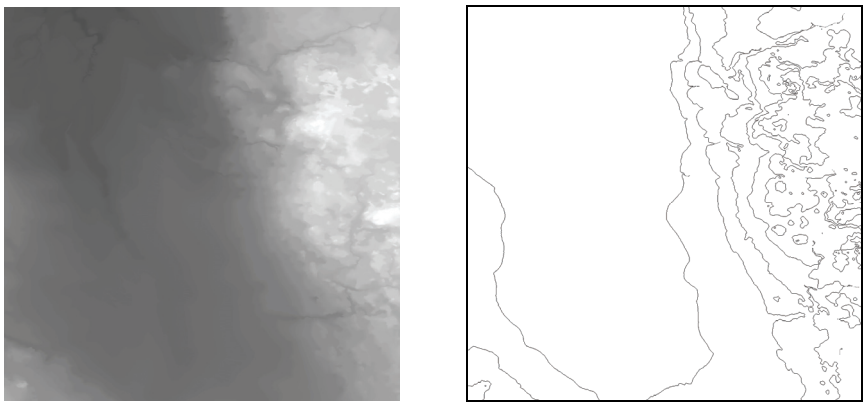
(c) Pratt (d) Cincinatti

Fig. 7. DEMs samples used in compression tests

Table 2. DEM compression results[6]

Name	WinZip	Compression	Our Technique	Compression
Eagle	917,504	90.67%	1,645,939	83.27%
Jacksonville	262,144	97.33%	236,126	90.07 %
Pratt	868,352	91.17%	1,306,272	86.73 %
Cincinatti	819,200	91.67%	1,642,829	83.31 %

Regarding the contour line computation, in Fig. 8 (b) we show the obtained result with the DEM source that appears in Fig. 8 (a). The altitude interval is 25 meters.



(a) (b)

Fig. 8. Contour lines result: (a) DEM original geo-image (b) Contour lines obtained from DEM compressed file

5 Conclusions

In the present work, we develop a loss-less compression algorithm using RLE encoding approach. Although the compression factor is not the same, we can obtain this factor applying other techniques, it is important to mention that encoded elevations can be directly read from DEM compressed file, and the computation of contours lines is easy and fast. The most important part of the test is the compression factor, which has been 80% high. To make some enhancements, this algorithm should be adapted as new format to store and describe digital elevations models. Future works are oriented to study others, encoding formulations that allow us to increase the compression coefficient and attempt to implement developed compression algorithms to compare them with our method.

An important characteristic is that the frequency of the DEM is not high, because the pixels do not present changes in their structure; the value of each pixel is

[6] All data are expressed in bytes.

correlated with their neighbor values. Due to this, it is possible to apply this kind of compression. Moreover, this approach is not useful for common images.

On the other hand, we have considered the most important characteristics of digital elevation models in our method, because it is important to point out that DEMs are complex geo-images, which involve several properties of a certain environment. In addition these properties reflect the semantics of the digital elevation models.

Acknowledgments

The authors of this paper wish to thank the CIC, CGPI, CONACYT and IPN for their support. Additionally, the authors wish to thank the reviewers for their pertinent comments.

References

1. Maune D.F., Digital Elevation Model Technologies and Applications: The DEM Users Manual, Asprs Pubns, USA, (2001).
2. Application of digital elevation models to delineate drainage areas and compute hydrologic characteristics for sites in the James River Basin, North Dakota, U.S. Geological Survey (USGS), USA, (1990).
3. Gousie M. B, Randolph F., Converting Elevation Contours to a Grid, Proceedings of the Eighth International Symposium on Spatial Data Handling, (1998).
4. Randolph F., Amir S., Lossy Compression of Elevation Data, USA, (1995).
5. Shantanu D. R., Sapiro G., Evaluation of JPEG-LS, the New Lossless and Controlled-Lossy Still Image Compression Standard, for Compression of High-Resolution Elevation Data, *IEEE Transactions On Geoscience And Remote Sensing*, Vol. 39, No. 10, (2001), pp. 2298-2306.
6. Creusere C. D., Compression Of Digital Elevation Maps Using Nonlinear Wavelets, IEEE, (2001), pp. 824-827.
7. Standards for Digital Elevation Models – Part 1, U.S. Geological Survey (USGS), USA, (2002).
8. Hinds, S.C, et al, A document skew detection method using run-length encoding and the Hough transform, IEEE, Proceedings, 10th International Conference on Pattern Recognition, Vol. 1, (1992), pp. 464-468.
9. Gonzalez R, Digital Image Processing Second Edition, Prentice Hall, USA, (2001).
10. Beenker, G F M, Immink, K A S, Generalized method for encoding and decoding run-length-limited binary sequences, *IEEE Trans. Info. Theory*, Vol. IT-29, No. 5, (1983), pp. 751-753.