International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.4, (2016), pp.339-346
http://dx.doi.org/10.14257/ijsip.2016.9.4.30

An Improved Two-Dimensional Run-Length Encoding Scheme and Its Application

Peng Wu[1,2], Shunping Zhou[1,3], Bo wan[1,3], Fang Fang[1] and Sha Zhou[1]

[1]*National Engineering Research Center for GIS, China University of Geosciences, Wuhan Hubei, CHINA*
[2]*Research Center on Applied Physics and Information Technology, Yangtze University, Jingzhou Hubei, CHINA*
[3]*Zondy Cyber Group Co. Ltd., Optical Valley Software Park, Wuhan Hubei, CHINA*
E-mail: zhoushunping@mapgis.com

Abstract

In this paper, we propose an improved two-dimensional run-length encoding (I2DRLE) scheme for representing grayscale images. Conventional 2D run-length encoding scheme is simple and effective that has been widely used, while it is not suitable to represent non-block images. Our approach is a new data compression algorithm inspired by 2D run-length encoding and quadtree, which apply some predefined patterns to represent various data and can sharply reduce the number of blocks in image representation. Experimental results show that this method is an effective lossless grayscale image encoding scheme.

Keywords*: image compression, bit-plane decomposition, 2D run-length encoding, quadtree*

1. Introduction

Data compression has attracted increasing research interest in recent years motivated by a range of applications from seismic exploration, remote sensing and telemetry, multimedia communication to Geographical Information System (GIS) [6]. Data compression is a long-history technique of human activities, these activities of data compression were informal before Shannon [1], who has first created a formal intellectual discipline for data compression. Thereafter, a great number of methods [2-6] have been proposed to represent various information. There are two main purposes for people to put forward new methods: (1) to minimize computer storage space; (2) to speed up a specific process. Recent researches have made a lot of progress on these two aspects.

Among numerous representation methods, run-length encoding (RLE) is a simple and effective compression scheme [7-9] in which runs of data are stored as a single data value and count. This characteristic makes it become widely used [11,13,14,15,17] and even become a component of other schemes, such as the JPEG standard [10], proposed in the 1980's and in use for transmitting and storing images, uses discrete cosine transformation, quantization, run-length coding, and Huffman entropy coding [2]. The idea of run-length encoding has also been extensively exploited by researchers for other kinds of data compression. These approaches attempt to compress data as either 1D streams, 2D images or 3D volumes. For example, Žalik and Lukač [16] introduce a new efficient chain code compression algorithm based on move-to-front transform and adaptive run-length encoding for compressing the repetitions of symbols' combinations. Qian and Chen [8] present an adaptive 2D run-length encoding(2DRLE) method to minimize the total number of run-lengths automatically, which has been proved to be effective through experiment. Shen and Spann [11] discuss a volumetric representation of 3D regions

ISSN: 2005-4254 IJSIP

achieved by the generalization of run-length coding for 2D binary images, then they assert that the run-length description for 3D regions has a better performance on the memory expense than the traditional octree representation. Quadtree [4-12] is another well studied hierarchical representation method, which has a certain connection with 2DRLE. It is commonly used in quadtree that data are segmented symmetrically, which will separate adjacent point to different block even if they are fairly correlative, hence it needs to be further improved.

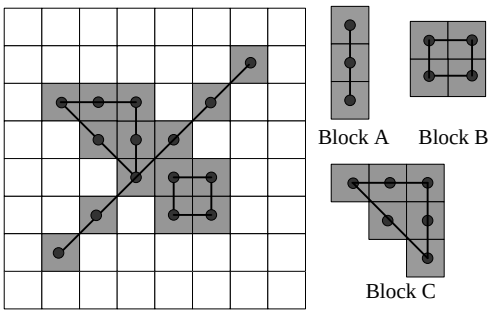
These two methods adopt a single type of block, *i.e.* square, which is not suitable to various images, especially to texture-rich images. In this paper, inspired by 2DRLE and quadtree, we employ several blocks, such as line, rectangle, triangle, then bit-plane decomposition are adopted, thus grayscale images are divided and the number of blocks has been significantly reduced, higher compression ratio is achieved as well.

The rest of the paper is organized as follows. The improved 2DRLE algorithm is described in Section 2, then the storage structure and data amount analysis is given in Section 3. Experiments and performance evaluation are presented and discussed in Section 4. Finally, conclusion are drawn in Section 5.

2. Improved 2DRLE Algorithm

In our proposed method several predefined blocks are adopted to represent the target data. We will put forward some typical predefined blocks and further improve the algorithm efficiency in two ways: (1) these predefined blocks are classified into three types, rectangle, triangle and line blocks. The former can be used to represent regular regions and the latter two are apt to non-block regions; (2) allow all types of blocks overlapped to form larger blocks.

With the purpose of explaining the merits of the improved method relative to traditional 2DRLE algorithm, we will show an example in Figure 1. According to sequential scanning order to find the starting point of block in the 8×8 binary image, it is obvious that the image can be split up into 3 blocks. As is shown in Figure 1 (a), if we use the traditional 2DRLE, there will be 9 blocks in total.



(a) Binary image (b) block instance

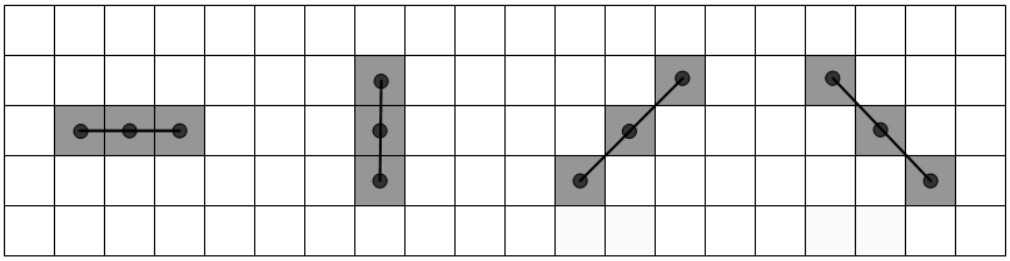
Figure 1. Binary Image By Using I2DRLE Segmentation

m	*Rectangle*	*Triangle*	*Line*
Grayscale	Number of Rectangle	Number of Triangle	Number of Line

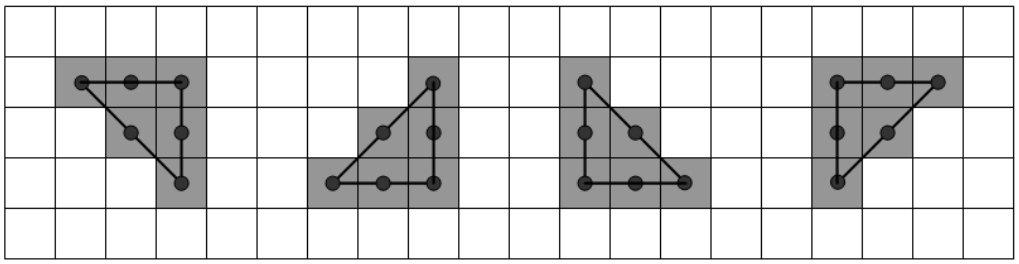
Figure 2. I2DRLE Header Structure

The principle of I2DRLE is to choose the largest one after scanning these three types of blocks as the final type of new block. Correlative parameters such as the coordinate of starting point, type of blocks and other parameters will be recorded

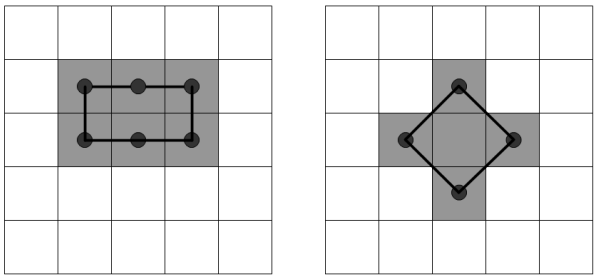
simultaneously. As a result, the binary image in Figure 1(a) can be divided into 3 blocks, which is shown in Figure 1 (b), *i.e.* block A, B and C. From this example, we can see that the number of block has decreased substantially. In order to adapt to various circumstance, all these three types of pattern can be rotated by 45 degree or 90 degree, Figure 3 shows all the variants of line, triangle and rectangle. With this rule some larger blocks will be detected and the total number will be reduced significantly.



(a) Four types of line



(b) Four types of triangle



(c) Two types of rectangle

Figure 3. Variants of Line, Triangle and Rectangle

In the above example, the number of blocks are 9 and 3 respectively. In order to save more storage space, I2DRLE algorithm does not need additional identification information, only need to add a header node and store the number of various blocks, where the space consumption is nearly the same as the original instances. Suppose that each blocks instance occupies L bits, the given binary image will occupy $9L$ bits by using 2DRLE and $3L$ bits by using I2DRLE. It shows that I2DRLE image representation method reduces storage space effectively.

For a $2^n \times 2^n$ grayscale image in I2DRLE algorithm, suppose that the gray scale is m, it produces m binary images, which is divided by using three types of predefined blocks. In order to save space, we just record black pixels and finally get the I2DRLE table for grayscale images.

Input: a $2^n \times 2^n$ grayscale image G with grayscale parameter m.

Output: I2DRLE table Q for grayscale image G.

Step 1: Decompose gray image G_k into m binary images $BP_i(1 \le i \le m)$, set number of bit-planes i to 1, point to the bit-plane where exists the lowest bits of each pixel.

Step 2: Carry on exclusive-OR operations according to pixel value between the former m-1 binary images BP_i and their following binary image BP_{i+1}, the last bit plane BP_m remain constant;

Step 3: Set various blocks counter, rect_num, tri_num and line_num to 0, record the number of blocks instances respectively, then define three arrays Q_1, Q_2 and Q_3, where the structure of data element is (x,y,height,width,type) which represent the starting point's coordinates, height and width of blocks instances.

Steps 4: Establish an unmarked starting point (x,y) from the first entrance of the image BP_i, trace the blocks and find out the biggest blocks in terms of the areas of the rectangles, triangles and lines, save parameters to the corresponding array Q_1, Q_2 and Q_3 according to the type of the biggest blocks.

Step 5: In the binary image BP_i, mark the area of selected rectangle blocks with 2, namely gray pixel, indicate that the pixels in the area cannot be used as the starting point, but can be reused and join subsequent blocks instances.

Step 6: Scan the image array in raster order to find the next unmarked black pixel as a new starting point.

Step 7: Repeat Step 4 to Step 6 until there are no unmarked new starting points.

Step 8: Merge Q_1 and Q_2 sequentially and set table header according to the counter variable, in this way a bit plane decomposition result has been saved. Increase the number of bit plane by one, *i.e.*, $i=i+1$. If $i \le m$, then go to Step 2, otherwise proceed to the next step.

Step 9: Merge the current bit plane decomposition results into Q_k, increase grayscale component number k by one. If $k \le 3$, return to Step 2, dispose the next gray image, or go to the next step.

Step 10: Scan the blocks instances one by one, employ K-Code transform of dimension reduction [7] and convert the coordinates to K-Code, that is, sp←K(x,y), then record the K-Code by using difference method.

Step 11: Output encoding result $Q=\{Q_1, Q_2, Q_3\}$.

3. Comparison Between the Two Methods

3.1 Storage Structure of I2DRLE

For a grayscale image, the output of I2DRLE is a queue set $Q = \{ Q_1, Q_2, Q_3\}$, each $Q_i(1 \le i \le 3)$ represents the number of different patterns. As is shown in Figure 4, storage structure of blocks instances is made up of two types of elements. One is the starting point, the other is the parameters of blocks instances. In general, the binary encoding lengths of x and y are both n. According to the definition of K-Code, the maximal lengths of both length and width are $n/2$. Since the storage structure of *sp* is represented by K-Code, storing *sp* need n bits. Therefore, storing a blocks instance needs 2n bits altogether.

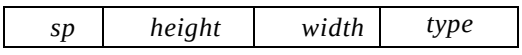


Figure 4. Storage Structure of Blocks Instances

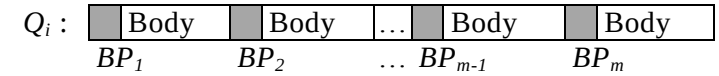


Figure 5. Storage Structure of a Grayscale Component

Figure 5 shows the storage structure of a grayscale component, which consists of the data of *m* bit planes, each bit plane is divided into two parts, table header and table body. The shaded area in Figure 5 is table header, the white part is table body. The table header will record the number of and the table body will record the data of blocks instances.

3.2 Data Amount Analyses of I2DRLE

As for images with different complexity, the data amount represented by I2DRLE are different. The following will make a comparative analysis between I2DRLE and classic linear quadtree.

First of all, linear quadtree adopts symmetrical division, and 2DRLE requires the block to be square, while I2DRLE adopts asymmetrical and overlapping division, for the same test image, I2DRLE will get less blocks than linear quadtree. Furthermore, each linear quadtree node needs 3*n*-1 bits, while each node represented by I2DRLE needs only 2*n* bits. Consequently for the same image, the storage requirement of linear quadtree representation will be 1.5 times or so that of the I2DRLE representation.

4. Experimental Results

In this section, some representative grayscale images of size $2^8 \times 2^8$ are analyzed to prove the obtained theoretical result. The gray level of these images is 8, *i.e.*, $m = 8$. We implement the algorithm for the I2DRLE and make a comparison with that of the classical linear quadtree and 2DRLE.

(a) Building (b) Cameraman (c) Peppers (d) Boats

Figure 6. Four Test Images of Size $2^8 \times 2^8$

Table 1 shows the comparison of the performance between Quadtree, 2DRLE and I2DRLE. From the table, it can be easily seen from the value of *N* that the blocks instances' number of the I2DRLE is much less than the nodes' number of the Quadtree and 2DRLE. In particular, the total nodes of the linear quadtree is 1.423 to 4.913 times that of I2DRLE, and the total nodes of the 2DRLE is 1.321 to 3.050 times of I2DRLE.

Table 1. Comparison of the Performance with Quadtree and 2DRLE

Image	*N*			*η*	
	Quadtree	2DRLE	I2DRLE	Q/I	R/I
Building	14887	9243	3030	4.913	3.050
Cameraman	61603	57177	43297	1.423	1.321
Peppers	58480	56175	37916	1.542	1.482
Boats	62002	58457	40279	1.539	1.451

Note: *N*: Number of blocks or nodes; *η*: compression ratio;

Q/I: Quadtree to I2DRLE; R/I: 2DRLE to I2DRLE.

The experimental results show that our algorithm for grayscale images is much more effective than that of the popular linear quadtree and the late 2DRLE, so it is a better method to represent grayscale images.

5. Conclusion

In this paper, we present an improved 2DRLE scheme for image representation, where we have introduced the scheme and analyzed the complexity and the data amount of this algorithm. By applying upon grayscale images, we implemented the algorithm and compared with the original and one of the typical adaptive 2DRLE algorithm. The theoretical and experimental results presented show that our implemented algorithm can reduce the data storage much more effectively than that of the 2DRLE and it is a better method to represent grayscale images.

The representation method of the I2DRLE has two purposes, just as the introduction section said. The first purpose has been elaborated in the paper. The other purpose, which has not been included in this study through, will be applied to improve the efficiency of some atomic operations, *e.g.* set operations, area computing, block searching and so on. Since the number of blocks has been reduced sharply, we believe the computation efficiency of the I2DRLE must be much higher than that of the pixel-based models and we will further this study in the future work.

Acknowledgements

This work has been partially supported by National Natural Science Foundation of China (Grant No. 41371422, 11304024), Science and Technology Support Project of Wuhan Science and Technology Bureau (Grant No. 2013010501010123), Development Fund Plan of Yangtze University(Grant No.2013cjp12), Open Fund Project of China University of Geosciences Teaching Laboratory (Grant No. 2013052).

References

[1] C. E. Shannon, "A Mathematical Theory of Communication," The Bell System Technical Journal, vol. 27, **(1948)**, pp. 623–656.
[2] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proceedings of the IRE, vol. 40, **(1952)**, pp. 1098 - 1101.
[3] S. Golomb, "Run-length encodings," IEEE Transactions on Information Theory, vol. 12, **(1966),** pp. 399-401.
[4] D. M. Mark and D. J. Abel, "Linear quadtrees from vector representations of polygons," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 7, **(1985)**, pp. 344-9.
[5] J. P. Lauzon, D. M. Mark, L. Kikuchi and J. A. Guevara, "Two-dimensional run-encoding for quadtree representation," Computer Vision, Graphics, and Image Processing, vol. 30, **(1985),** pp. 56-69,.
[6] D. M. Mark, "The Use of Quadtrees in Geographic Information systems and spatial data handling," Proc. AutoCarto, London, vol. 1, **(1986)**, pp. 517-526.
[7] H. Samet, "Hierarchical Spatial Data Structures," in Proceedings of the first symposium on Design and implementation of large spatial databases (SSD '90), New York, NY, USA, **(1990)**, pp. 193-212.
[8] Z. Qian and R. Chen, "Adaptive 2d run-length coding and its applications," Journal of PLA institute of surveying and mapping, vol. 2, **(1993)**, pp. 18-23.
[9] H. Samet, "Spatial Data Structure," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 8, **(1995)**, pp. 532-538.
[10] G. K. Wallace, "The JPEG still picture compression standard", Commun. ACM, vol. 34, no. 4, **(1991)**, pp. 30-44.
[11] X. Shen and M. Spann, "3D Region representation based on run-lengths: operations and efficiency," Pattern Recognition, vol. 31, **(1998)**, pp. 575-585.
[12] Y. Yang, K. Chung and Y. Tsai, "A compact improved quadtree representation with image manipulations," Image and Vision Computing, vol. 18, **(2000)**, pp. 223 - 231.
[13] F. K. H. Quek, "An algorithm for the rapid computation of boundaries of run-length encoded regions," Pattern Recognition, vol. 33, **(2000)**, pp. 1637-1649.
[14] J. Shin, H. Hwang and S. Chien, "Detecting fingerprint minutiae by run length encoding scheme," Pattern Recognition, vol. 39, **(2006)**, pp. 1140-1154.
[15] C. M. Agulhari, I. S. Bonatti and P. L. D. Peres, "An Adaptive Run Length Encoding method for the compression of electrocardiograms," Medical Engineering & Physics, vol. 35, **(2013)**, pp. 145-153.
[16] B. Žalik and N. Lukač, "Chain code lossless compression using move-to-front transform and adaptive run-length encoding," Signal Processing: Image Communication, vol. 29, **(2014),** pp. 96-106.

[17] S. Ishikawa, H. Wu, C. Bi, Q. Chen, H. Taki, and K. Ono, "Fluid Data Compression and ROI Detection Using Run Length Method," Procedia Computer Science, vol. 35, **(2014)**, pp. 1284-1291.

Authors

Peng Wu, he is a PhD candidate in Faculty of Information Engineering, China University of Geosciences, Wuhan, China. His current research interests include 3D geological modeling, image processing and pattern recognition.

Shunping Zhou, he is a professor in Faculty of Information Engineering, China University of Geosciences, Wuhan, China. His current research interests include spatial database analysis, GIS theory and software engineering.