Send Orders for Reprints to reprints@benthamscience.ae

The Open Automation and Control Systems Journal, **2015,** *7,* **879-883**

Compression Algorithm of 3D Point Cloud Data Based on Octree

Dai Cheng-qiu*, Chen Min and Fang Xiao-yong

Department of Computer and Information Science, Hunan Institute of Technology, Hengyang 421002, China

Abstract: Octree is a tree data structure which is used to describe three-dimensional space such as three-dimensional modeling, 3D collision detection, and 3D data compression and so on. This paper focuses on compression algorithm of 3D point cloud data based on the octree. Firstly, the coordinates of 3D point cloud data converted to Morton without distortion were discussed. As the Morton is too discrete, an optimization algorithm for the Morton is proposed by IT on this basis. Finally, this paper proposed a program which is attached to 1-byte-status bit to represent the Morton who had the same parent. The program can improve the result of compression

Keywords: Data compression, Morton, Octree, 3D point cloud data.

1. INTRODUCTION

The original data collection by three-dimensional laser scanning technology is called the point cloud data, which are discrete massive data sets without a distribution pattern. So, it is necessary to do some processing on these data, such as establishing three-dimensional data model and do efficient data compression on the three-dimensional data based on data model, in order to make the three-dimensional network data transmission easy, so that all users can have access to the network and use the three-dimensional data. Three-dimensional data compression is based on two ways of three-dimensional data representation: surface-based representation and volume-based representation. Volume notation usually adopts octree structure model, and node data consist of the spatial coordinates and characteristics.

Because of hardware and software support for the surface-based representation, the triangle mesh compression research is mainly divided into two directions: the single resolution compression and multi-resolution compression. Among the existing single resolution compression methods, Michael Deering's [1] based on common triangular mesh geometry compression, Stefan Gumhold's [2] the triangle mesh between compression algorithms and real-time connection, and Gabriel Taubin's [3] surgery based topology geometry compression algorithm are more pronounced. In the existing multi-resolution compression research methods, Stefan Gumhold's [4] compression of discrete multi-resolution models, and Gabriel Taubin's [5] progressive forest split algorithms are fairly representative.

Expression methods based on the volume are commonly carried out by octree model, the literature [6-9] take split on the three-dimensional volume data based on octree structure then do wavelet transformation and further develop the various transformations compression. The 3D spatial data model used in this paper is based on the octree model. First, it explores how to effectively convert three-dimensional coordinates of the point cloud data into octree's addressed coordinates Morton code, then considering that the converted code is too discrete, it proposes a Morton code optimization algorithm which further proposed a program attached to 1-byte-status-bit to represent the same root Morton code scheme which is used to optimize Morton code compression. Lastly, this and the evaluation algorithm compression results are compared with previous algorithms [1, 2, 8].

2. OCTREE MODEL

The octree model is a tree data structure to describe three-dimensional space, and it can be seen as the promotion of quadtree method in three-dimensional space. A three-dimensional array of voxels represents an improved physical method. Octree is the decomposition of a three-dimensional shape, it carries an external cube shape's division into eight smaller cubes from front, left, up and down directions, as shown in Fig. (**1**). If the small cube unit is full or empty, it indicates that the cube is completely in the form or not, then it stops decomposition. Physical possession of some small cubes needs to be further decomposed into eight sub-cubes, until all elements of small cubes are full or empty, or decomposed to a predetermined decomposition accuracy (similar to resolution of three-dimensional object). According to Storage structure, octree are divided into pointer octree and linear octree.

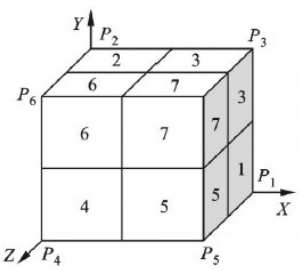

Fig. (1). Three dimensional cube corresponding to octree node.

*Address correspondence to this author at the Department of Computer and Information Science, Hunan Institute of Technology, Hengyang 421002, China; Tel: 13973454583; E-mail: dcq7777@163.com

1874-4443/15

Table 1. The morton of original data (N=84774).

i	x	y	z	Morton-hex
1	-0.0072346	-0.05432	-0.04312	11BDEB6790ADB6DC01979
2	-0.0072346	-0.0543785	-0.043155	11BDEB67903FFF903744B
3	-0.0072385	-0.054361	-0.048235	11BDEB65D13F7FCFD9400
i	-0.0083175	-0.0645297	-0.0485673	11BDEB64D533862F5230E
N-1	3.048762	5.512875	7.4597612	8D7155C37616EC3B4F861
N	5.342715	6.4367151	7.834298	8DEE1E7496379E546F26B
	x	y	z	parameters
Min	-5.8743217	-6.517821	-8.236747	L=9AFA18D
Max	5.542315	6.7623155	8.0137631	n=24

The pointer octree represents each node in the tree by recording nine fields. One field is used to describe the characteristics of the node (full, empty, partially occupied), the rest of the eight fields are used as a store of its eight sub-point node pointer. The pointer octree has some defects, and the biggest problem is that the pointer takes up a lot of space. Assume that each pointer uses two bytes, description node uses a byte, and then amount of storage of the pointer is to be 94% of the total amount of storage. Thus, although it is easy to use this storage method, yet it cannot be applied to three-dimensional data compression.

The linear octree advantage is that it saves only the leaf nodes, and each leaf node only needs two fields to store for storage attribute and spatial location in each field. In the application it will encode each octree leaf node, and each octree's coded value is unique, which can be identified by the path from the root to leaves. Moreover, this value also determines the octree leaf node's position in the integer coordinate system, which we call Morton codes, and each code by Morton digit consists of R octal numbers, and R is the maximum number of octree decomposition. The linear octree saves a lot of storage space; the consequences are causing lower node query efficiency. Because when octree structure changes, storage nodes have to be reordered, which makes linear octree in practical applications, subject to many restrictions. For this we established linear structure and multi-index, so the query efficiency and operational efficiency in the application process octree nodes are greatly improved.

3. CONVERTING POINT CLOUD SPACE COORDINATES TO MORTON CODE

Three-dimensional point cloud is a set of points, the point cloud data is represented by a three-dimensional spatial coordinates (x, y, z), and the Morton belongs to voxel representation. The point cloud data are converted from the spatial coordinates to octree structure Morton code, which is equivalent to one point with a small cube, so the accuracy of the octree decomposition directly determines the degree of distortion point cloud data. In this paper, the spatial coordinates of the point cloud are converted into the application code algorithm Morton; detailed steps are as follows:

(1) The scanned point cloud data mainly consists of the x, y, z coordinates. Assuming that raw point cloud data set is S (x, y, z), and the point coordinate data type is a double-precision floating-point. First, screen out unqualified point cloud data, then expand decimal point cloud data S (x, y, z) to 10n times to get S '(x, y, z) according to n significant digits, so as to ensure maximum data valid bit number and enhance the computing speed of data.

(2) S '(x, y, z) is traversed, and we get x, y, z of the maximum and minimum values respectively. Then, a coordinate translation is done based on the point (xmin, ymin, zmin), obtaining a non-negative integer set S "(x ', y', z ').

(3) Solve the cube surrounded by the point cloud data S "(x ', y', z '), the cube edge length L is max (x', y ', z'), then get the resolution's base value of $n=\lceil \log_2 L \rceil$ of space requirements based on L , namely the depth of the octree is n, which means with a side length L, space size $2n \times 2n \times 2n$ cubes are translated to surround the point cloud, and finally convert the point cloud data S "(x ', y', z ') into a binary representation, then obtain BinaryS" (x', y ', z').

(4) We take reference's method bit computation to evaluates BinaryS "(x ', y', z ')'s Morton code, which are the x ', y', z sequencing result of BinaryS" (x', y ', z') in the order of $z \rightarrow y \rightarrow x$ arrangement. Finally, the results are shown in Table **1**.

4. FURTHER COMPRESSION OF MORTON BASED ON THE PRINCIPLE OF OCTREE CODE

We do Analysis of Morton code data in Table **1** and find if there is a possibility of further compression. The Morton code is very discrete, it retains significant long bit in order

Table 2. Morton differences in ascending list.

Numbers	Diff_Morton $_{(10)}$	$\lfloor \log_8 Diff\_Morton \rfloor$
1	145218705	K=8
2	2013741534	K=10
3	9043758721	K=11
N-1	89037651281227550540	K=22

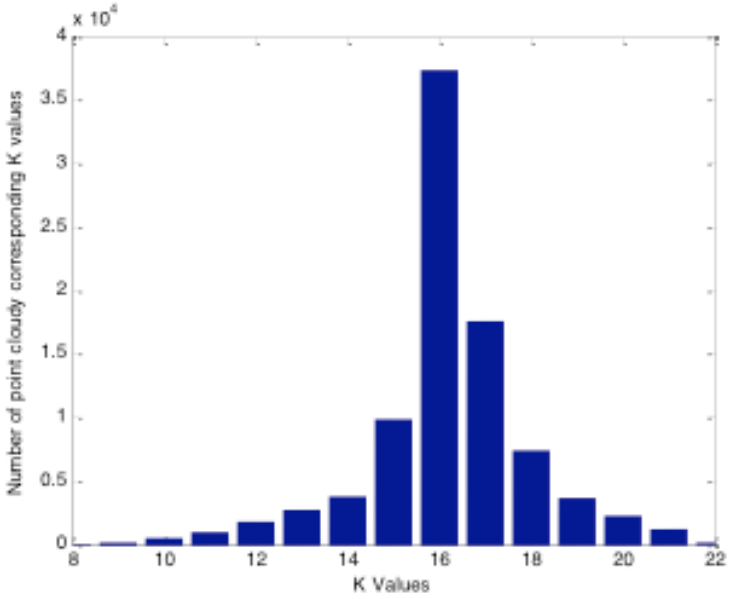

Fig. (2). Distribution of K values.

to achieve the highest resolution, which results in a space on the leaf nodes of the octree for discrete distributions and low storage space utilization. Usually for simple applications or online browsing, there is not much requirement for three-dimensional data accuracy, which can be further compressed, using the following two ideas for further compression.

4.1. Directly Reducing Morton Code

Table **1** calculates Morton code's high resolution, according to their own point cloud image accuracy requirements, it can get appropriate value of Diff_Morton by the use of the difference between adjacent Morton codes, and $\lfloor \log_8 Diff\_Morton \rfloor$ is taken as a direct shrink multiple Morton code, and hence the Morton code continuity can be improved.

Take Table **1** as an example. First, we will calculate Diff_Morton difference values between all adjacent two Morton codes in the entire table, the results are shown in Table **2**. Then, calculate the Morton code between the minimum Morton code Diff_Mortonmin, and the maximum Diff_Mortonmax. Morton code directly reduce must be an integer multiple of 8. In order to ensure that x, y, z three directions simultaneously amplify, Morton code directly reduces multiple K ranges to statistical distributions K values, which are shown in Fig. (**2**). Fig. (**2**) shows K values follow a normal distribution, indicating that the relative concentration of point cloud are data point distribution, so the accuracy is within the allowable range, replacing part of the distribution of available minority point with over-concentration points. Figure 2 shows calculation of K with the accuracy requirements allowing its optimal value; in this example, the optimal K value is 15.

4.2. Further Compression of Morton Code

Morton code's directly reduction substantially reduces octree resolution and the accuracy of the data. It is mainly achieved by substituting the upper leaf nodes with the root node of the plurality of data compression. When the distribution of point cloud data is dense, this method is of high efficiency. For a large point cloud data, abandoning certain details' influence on the accuracy of the impact of the image data is limited, but it can greatly reduce the length of the Morton code, reducing storage space and improving the efficiency of data processing.

Morton optimized code can be further reduced. The current article proposes compensation algorithm to get further compression of the lowest resolution Morton; the compres-

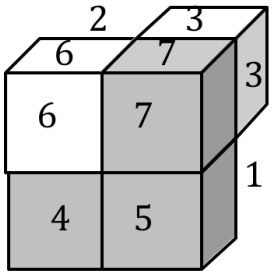

Fig. (3). Cloud node of octree.

sion algorithm idea is as follows: First, get through all the leaf nodes, and simply leave a leaf node with the same root; with the same root of the remaining leave we take additional one byte node information to reflect the status bits. In Fig. (**3**), if the leaf nodes 4,5,7 of the Morton code are respectively 11BDEB64,11BDEB65,11BDEB67, we can only keep the coding leaf node 4, and the addition of a status bit uses 10,110,000 to represent up to nearly 3 times compression. The best case can get nearly seven times of the compression efficiency. The function of the pseudo code in the above algorithm is:

```
Void Comp_Morton(V)
{
 int i, j, k,
For(i=1;i<=n&&Vi!=NULL;i++)
{
 For(j=i+1;j<=i+8;j++)
 {
 if(Comp_root(Vi ,Vj )==True) // Comparing neighbor nodes' root
{
Strcat(Vi, Vj_tag); // one byte for status
Delete Vj ; // delete another node of the same root
}
}
}
}
```

5. EXPERIMENTAL RESULT

Take Stanford Bunny model as an example, compare Morton code's direct reduction result and further compression result to find out data loss rate, compression ratio, accuracy and octree depth and effects; the data comparison results are shown in Table **3**, the effect is shown in Fig. (**4**). As can be seen, Morton code optimization and further compression makes octree depth greatly reduced and improve the compression ratio of the data. Compare the results of the evaluation of compression algorithm in Table **4** and literature [1, 2, 8]. Although the algorithm has some disadvantages in accuracy and time, yet in terms of compression ratio it has a better performance.

(**a**) Before compression

(**b**) Morton'direct compression

(**c**) Further compression

Fig. (4). The effect of comparison before and after compression.

CONCLUSION

This paper focuses on the compression algorithm of 3D point cloud data using linear octree spatial data structure, Morton coding optimization and proposing further compression algorithms. For the three-dimensional surface scanning data large space dispersion characteristics, we proposed a simplified Morton code algorithm, which can greatly reduce the depth of the octree, enhance continuity of Morton code. And on this optimization coding basis, the paper further proposes the attached 1-byte-status-bits to represent the same

root Morton code, and the algorithm can receive up to 7 times compression effect. On the basis of this article, other compression algorithms may be further discussed, such as the application of Huffman compression method optimized Morton code.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by Natural Science Foundation of China (No. 61402164), Science- Technology of Hunan Province, China (No. 2014GK- 3060), Science-Technology of Hengyang City, Hunan Province, China (No. 2012KG75) Foundation of Hunan Educational Committee, China (No. 10C0587), Students research and innovative experiment projects of Hunan Province, China (No. 2011415).

REFERENCES

[1] M. Deering, "Geometry compression," In: *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, ACM Press: New York, 1995, pp. 13-20.

[2] S. Gumhold, and W. Stroβer, "Real time compression of triangle mesh connectivity," In: *Computer Graphics Proceedings, Annual Conference Series,* ACM SIGGRAPH, ACM Press: New York, 1998, pp. 133-140.

[3] G. Taubin, and J. Rossignac, "Geometry compression through topological surgery," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 84-115, 1998.

[4] S. Gumhold, and R. Klein, "*Compression of Discrete Multiresolution Models*," Tech Rep: WSI-98-1, WSI-GRIS, 1998.

[5] T. Gabrial, G. Andre, H. William, and L. Francis, "Progressive forest split compression," In: *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, ACM Press: New York, 1998, pp. 123-132.

[6] S. Guthe, M. Wand, J. Gonser, and W. Strasser, "Interactive rendering of large volume data sets," In; *Proceedings of the conference on Visualization,* IEEE Computer Society, Washington DC, 2002, pp. 53-60.

[7] K. G. Nguyen, and D. Saupe, "Rapid high quality compression of volume data for visualization," *Computer Graphics Forum*, vol. 20, no. 3, pp. 49-57, 2001.

[8] G. Wetekam, D. Staneker, U. Kanus, and M. Wand, "A hardware architecture for multi- resolution volume rendering," In: *Proceedings of the* ACM SIGGRAPH/EUROGRAPHICS *Conference on Graphics Hardware,* ACM Press: New York, 2005, pp. 45-51.

[9] J. Woodring, and H.W. Shen, "Multiscale time activity data exploration *via* temporal clustering visualization spreadsheet," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 123-137, 2009.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014