An Algorithm of Neighbor Finding on Sphere Triangular Meshes with Quaternary Code

Sun Wenbin Zhao Xuesheng

Abstract The characteristic of Quaternary codes is analyzed. The rule of distinguishing triangle direction is given out. An algorithm of neighbor finding by decomposing the Quaternary code from back to front is presented in this paper. The contrastive analysis of time complexity between this algorithm and Bartholdi’s algorithm is approached. The result illustrates that the average consumed time of this algorithm is about 23.66% of Bartholdi’s algorithm.

Keywords Quaternary code; neighbor finding; sphere triangular mesh

CLC number P208

Introduction

In multi-scale spatial data management with the data structure of a sphere triangular mesh (STM), the neighbor finding of grid is very important for some spatial operations, such as spatial clustering, global indexing[1], spatial range query[2], dynamic dilation[3,4], etc. It has also been one of the hot research topics on the global discrete grid. STM encoding schemes include Dutton’s code[5,6], Goodchild’s code[7], Fekete’s Sphere Quatree (SQT) code[8], Otoo’s Semi-Quatree code[9], and Quaternary code[1]. These codes have excellent characteristics, such as good hierarchy and nesting. Although Quaternary code is the only continuous code among the aforementioned encoding schemes, finding the neighbor’s Quaternary code consumes more time than other codes. The reason is that the neighbor finding algorithm approach by Bartholdi is carried out by computing the distances of a point on the STM edge and the midpoint of different-level STMs, in which a number of floating-point operations, such as power and extraction, are required. To improve the efficiency of neighbor finding in Quaternary code, a new algorithm of neighbor finding by decomposing the Quaternary code from back to front is presented.

1 The principle of decomposing neighbor finding algorithm

1.1 The characteristics of Quaternary code

Quaternary code has some excellent characteristics, which are very useful for finding adjacent STMs’ codes. Following is a critical examination of these characteristics:

1) The direction of an STM can be judged by the parity of the number of figure “1” in the STM code. If it is odd, the direction of the STM is downwards and
this STM can be called down-STM. In the other case, the direction is upwards and the corresponding STM can be called upper-STM.

2) If two STMs are mutually adjacent, their father cells are also adjacent or the same.

3) A son-STM code is composed of its father cell’s code and one other number, which is less than four and is called the attached-code. When an STM is divided into four son-STMs, there are four encoding schemes of attached-code, shown as Fig.1. The detailed cases are described as follows:

(1) If the number of figure “3” in an STM code is even and its father cell is an upper-STM, the encoding scheme of the attached-code is illustrated as Fig.1(a);

(2) If the number of figure “3” in an STM code is odd and its father cell is an upper-STM, the encoding scheme of the attached-code is illustrated as Fig.1(b);

(3) If the number of figure “3” in an STM code is even and its father cell is a down-STM, the encoding scheme of the attached-code is illustrated as Fig.1(c);

(4) If the number of figure “3” in an STM code is odd and its father cell is a down-STM, the encoding scheme is illustrated as Fig.1(d).

2 The rules of code transformation

To describe code transformation clearly, some stipulations of normal symbol and operation are defined. Code indicates the code of STM to find neighbors; code-1 means that code is considered as a number of quaternary systems and it minus one; code-2 is that code minus two; code+1 is that code plus one; code+2 is that code plus two. The operation “<ab” indicates that the last two attached-codes of Quaternary code are changed into ab and other figures are not changed, while “>ab” indicates that the last two attached codes of Quaternary code are changed into ab and other figures are replaced by the neighbor’s code of corresponding second-father STM. Neighbors’ codes can be defined as Neighbor1, Neighbor2 and Neighbor3. The detailed transformation rules are shown in Table 1. The relationships between STM and its neighbor’s codes are shown in Table 2.

3 Experiment and analysis of the algorithm efficiency

3.1 Analysis of the algorithm efficiency

Decomposing algorithm for finding STM neighbor’s code can be carried out in most cases. However, not
all of neighbor codes can be calculated with this algorithm, in the cases where:

- The end of code is 230 or 330;
- The end of code is 032 or 332;

| Code | Neighbor1 | Neighbor2 | Neighbor3 |
|------|-----------|-----------|-----------|
| End of 0 | code-1 | code <32 | code+1 |
| End of 20 | code-1 | code >02 | code+1 |
| End of 030 or 130 | code<33 | code >22 | code+1 |
| 300 | code-1 | code >22 | code+1 |
| 100, 200, 000 | code-1 | code >00 | code <23 |
| End of 2 | code-1 | code >10 | code >23 |
| End of 02 | code+1 | code >10 | code >23 |
| End of 12 | code-1 | code >00 | code >33 |
| End of 22 | code+1 | code >10 | code >23 |
| End of 132 or 232 | code+1 | code >10 | code >23 |
| End of 3 | code+1 | code >10 | code >23 |
| End of 103 or 203 | code+1 | code >10 | code >23 |
| End of 13 | code<20 | code >22 | code <12 |
| End of 023 or 123 | code<20 | code >22 | code <12 |
| End of 133 | code>22 | code >22 | code+1 |

Table 2 Neighbor relations

| Odd/Even of the Number | Left Neighbor | Right Neighbor | Upper/Down Neighbor |
|------------------------|---------------|----------------|---------------------|
| Odd “1” & Even “3” | Neighbor1 | Neighbor2 | Neighbor3 |
| Odd “1” & Odd “3” | Neighbor3 | Neighbor2 | Neighbor1 |
| Even “1” & Even “3” | Neighbor2 | Neighbor3 | Neighbor1 |
| Even “1” & Odd “3” | Neighbor2 | Neighbor1 | Neighbor3 |

- The end of code is 203 or 303;
- The end of code is 223 or 323; or,
- The end of code is 033, 233 or 333.

Of those STMs, 11 out of 64 cannot use this algorithm to find neighbors. In those cases, Bartholdi’s algorithm substitutes for it.

Bartholdi’s algorithm for finding neighbors is based on calculating the distances between a point on the STM edge and mid-point of each subdivided son-STMs. Each subdivision operation and related calculation is taken as a single-operation. Similarly, the “>” operation is taken as a single-operation of the decomposing neighbor finding algorithm. On the one hand, each single-operation of Bartholdi’s algorithm involves complex operations, such as floating-point multiplication, radication and so on. On the other hand, a single-operation of decomposing algorithm can be transformed into a binary operation. Therefore, single-operation of this algorithm is simpler than that of Bartholdi’s. Furthermore, suppose that there are $K$-level STMs. $K$ single-operations are required to find the neighbor code in Bartholdi’s algorithm while $K/2$ single-operations are needed in the decomposing algorithm.

Average time complexity of a decomposing algorithm is presented by calculating the occurring probability of the “>” operation. The probability of codes ending with “20” is 1/16 and the probability of “>” operation in the code transformation is 1/3. The corresponding probability is (1/16)*(1/3). The sum of the probability of “>” operation can be calculated using the same way and the total probability is 5/32 in the decomposed algorithm. Decomposing algorithm is used to find the neighbor in 53/64 cases. Therefore, average time complexity is presented as follows:

$$(53/64)*(5/32)*K/2 + K*11/64 = 0.23657K$$

In other words, the average time to find the neighbor is about 23.66% of Bartholdi’s.

3.2 Experiment

The experiment on comparing the efficiency of Bartholdi’s algorithm with the decomposing algorithms is done. All Quaternary codes of STMs from 6th to 13th subdivision level are searched out. The times using Bartholdi’s and the decomposing algorithm to find the neighbor are calculated respectively.
Also, the same method is used to calculate the times from 14th to 25th partial subdivision level STMs (shown in the Table 3). The reason is that the time for finding 1st to 5th level STMs is so little that it is ignored in this experiment because of the small number of each level STMs and short length of codes. On the other hand, when the subdivision level comes to 14, the number of STMs is up to 268,435,456 and the time to find all STMs’ neighbors exceeds 6 hours. For this reason, only part STMs are carried out in the experiment. The result indicates that: 1) with increment of STM subdivision level, the ratio of time for finding neighbors using the decomposing algorithm and Bartholdi’s is prominently reduced and this ratio falls from 34.97% to 25.42%; 2) The results of the experiment coincide with the theoretical analysis above.

**Table 3  The ratio of neighbor-finding with Bartholdi’ and decomposing method**

| Layer | Frequency of neighbor finding | Consumed time of decomposing | Consumed time of Bartholdi | Decomposing algorithm / Bartholdi algorithm |
|-------|------------------------------|-----------------------------|---------------------------|------------------------------------------|
| (N)   | 4^N                          | Times                       | Decomposing algorithm / ms| Bartholdi algorithm                     |
| 6     | 4^N                          | 4,096                       | 179                       | 512                                       | 34.97%                                   |
| 7     | 4^N                          | 16,384                      | 806                       | 2,418                                     | 33.34%                                   |
| 8     | 4^N                          | 65,536                      | 3,564                     | 11,003                                    | 32.39%                                   |
| 9     | 4^N                          | 262,144                     | 15,938                    | 49,578                                    | 32.15%                                   |
| 10    | 4^N                          | 1,048,576                   | 67,874                    | 221,389                                   | 30.66%                                   |
| 11    | 4^N                          | 4,194,304                   | 290,761                   | 975,923                                   | 29.79%                                   |
| 12    | 4^N                          | 16,777,216                  | 1,221,196                 | 4,196,468                                 | 29.10%                                   |
| 13    | 4^N                          | 67,108,864                  | 5,092,387                 | 17,834,989                                | 28.55%                                   |
| 14    | 4^N-1                        | 67,108,864                  | 5,665,024                 | 20,085,120                                | 28.21%                                   |
| 15    | 4^N-2                        | 67,108,864                  | 5,972,032                 | 21,551,040                                | 27.71%                                   |
| 16    | 4^N-3                        | 67,108,864                  | 6,364,160                 | 23,308,992                                | 27.30%                                   |
| 17    | 4^N-4                        | 67,108,864                  | 6,601,024                 | 24,339,968                                | 27.12%                                   |
| 18    | 4^N-5                        | 67,108,864                  | 6,913,024                 | 26,049,088                                | 26.54%                                   |
| 19    | 4^N-6                        | 67,108,864                  | 7,327,040                 | 27,674,880                                | 26.48%                                   |
| 20    | 4^N-7                        | 67,108,864                  | 7,595,008                 | 29,131,072                                | 26.07%                                   |
| 21    | 4^N-8                        | 67,108,864                  | 7,867,968                 | 30,360,960                                | 25.91%                                   |
| 22    | 4^N-9                        | 67,108,864                  | 8,188,992                 | 32,061,056                                | 25.54%                                   |
| 23    | 4^N-10                       | 67,108,864                  | 8,512,000                 | 33,372,992                                | 25.51%                                   |
| 24    | 4^N-11                       | 67,108,864                  | 8,924,992                 | 35,052,032                                | 25.46%                                   |
| 25    | 4^N-12                       | 67,108,864                  | 9,332,190                 | 36,712,000                                | 25.42%                                   |

**References**

[1] Bartholdi III J J, Goldsman P (2001) Continuous indexing of hierarchical subdivisions of the globe[J]. *Int J Geographical Information Science*, 15(6): 489-522

[2] Nulty W G (1993) Geometric searching with spacefilling curves [D]. Atlanta:Georgia Institute of Technology

[3] Lee M, Samet H(2000) Navigating through triangle meshes implemented as linear quadtrees[J]. *ACM Trans on Graphics*, 19(2), 79-121

[4] Chen J, Zhao X S(2003) An algorithm for the generation of voronoi diagrams on the sphere based on QTM[J]. *Photogrammetric Engineering & Remote Sensing*, 69(1): 79-89

[5] Dutton G (1990) Locational properties of quaternary triangular meshes[C]. The Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland

[6] Dutton G (1996) Encoding and handing geospatial data with hierarchical triangular meshes[C]. The 7th International Symposium on Spatial Data Handing, Netherlands

[7] Goodchild M F, Yang S (1992) A Hierarchical data structure for global geographic information systems[J]. *Computer Vision and Geographic Image Processing*, 54(1): 31-44

[8] Fekete G (1990) Rendering and managing spherical data with sphere quadtrees[C]. Visualization’90, San Francisco

[9] Otoo E I, Zhu H (1993) Indexing on spherical surfaces using semi-luadcodes[C]. SSD’93, singapore