A Massive Laser Point Cloud Data Organization Strategy
Based on the Mixed Model

Yingying Wang*, Yuye Zhang, Shangqiang Wang, Xiaojia Ji
*Naval Aviation University Qingdao Campus, No.2, Siliuzhong Road, Qingdao 266041, China
*26946350@qq.com

Abstract: In the process of data interaction, neighborhood query, filtering, visualization, and dynamic update of LiDAR point cloud data, how to efficiently organize and process massive point cloud data, and quickly index and locate any point in the point cloud and its neighborhood Search is a key issue to be solved urgently. In this paper, combining the advantages of a virtual grid with no interpolation loss on original data, sT spatial relationship, and low memory occupation of the octree, we design an index method based on the combination of virtual grid and adaptive octree based on dynamic scheduling of internal and external memory. Realize the organization and scheduling of massive LiDAR laser scanning point cloud data.

Keywords: massive point cloud, virtual grid, adaptive octree, combined index

1. Introduction

With the popularity of high-resolution LiDAR equipment, point cloud data has gradually developed to a massive level. How to efficiently organize and process the acquired massive point cloud data, and be able to quickly index and locate any point and search in the neighborhood, has become a key issue in 3D laser scanning technology.

The core of point cloud data organization and management is spatial indexing[1]. At present, the commonly used method is to realize the spatial organization of data based on the pre-built index of the memory to the point cloud[2]. According to the index construction method of the point cloud data in the memory, the spatial index can be divided into a single index and a combined index. Commonly used single indexes include grid index, quadtree index, octree index[3], KD tree[4] and R tree.
The composite index is a combination of multiple data structures in the index construction to form a multi-level spatial index. The literature respectively introduced the composite index based on octree and R tree, octree and binary tree, combined index of octree and KD tree. However, the above-mentioned data organization methods are based on full memory, and the amount of data that can be processed is restricted by the memory capacity of the computer, which seriously affects the efficiency of the frequent I/O operations of massive point cloud data. This paper designs an index method that combines a virtual grid and an adaptive octree based on the dynamic scheduling strategy of internal and external memory to realize the organization and scheduling of massive point clouds.

2. Point cloud data organization based on virtual grid and adaptive octree

2.1 The division of virtual grid

The virtual grid divides the point cloud space into grids of equal size according to certain length and width rules, and records the three-dimensional coordinates of the point cloud in the grid, as shown in Fig.1.

The basic idea of dividing the virtual grid is to record the corresponding point cloud space coordinates in the virtual grid. When performing spatial query and retrieval, first locate the grid where the three-dimensional point to be retrieved is located, and then search for adjacent points in the grid. The steps are as follows:

1) Read the point cloud header file information, obtain the range of each point cloud bounding box and the total number of point clouds, calculate the density of the point cloud according to formula (1):
\[ d = \frac{\sum_{i=1}^{n} num_i}{(\max_y - \min_y)(\max_x - \min_x)} \]  

(1)

\(num_i\) is the number of the i-th point cloud, at the coordinate range of the bounding box of the i-th point cloud in the x and y directions.

2) Enter the number of node point k, calculate the side length \(l\) of each grid:

\[ l = \sqrt{\frac{k}{d}} \]  

(2)

3) Calculate the number of point cloud space in x and y direction:

\[ m = \text{ceil} \left( \frac{\max_x - \min_x}{l} \right) \]

\[ n = \text{ceil} \left( \frac{\max_y - \min_y}{l} \right) \]  

(3)

\(m\) is the number of blocks of the point cloud in the x direction, and \(n\) is the number of blocks of the point cloud in the y direction.

4) Traverse this point cloud, calculate the grid where the laser foot point \(p\) is located, and write the point cloud in the same grid to the hard disk:

\[ m_i = \frac{p_{x_i} - \min x}{l} \]

\[ n_i = \frac{p_{y_i} - \min y}{l} \]  

(4)
Where and are the row and column numbers of the grid where point p is located, \(0 \leq m \leq n\), \(0 \leq n \leq n\).

2.2 Construction of adaptive linear octree

The octree is to continuously recursively divide the spatial geometric entities into eight geometric objects with the same time and spatial resolution until each subregion meets a certain threshold condition, forming a direction graph with root nodes. For point cloud data, the number of point clouds in the leaf node, whether the local curvature is greater than a preset threshold, or whether the tree depth reaches the preset maximum depth, etc. can be used as the judgment condition to stop the segmentation. The split form is shown in Fig.2.

![Fig. 2 Schematic diagram of the octree structure](image)

Selecting a linear octree suit for 3D point cloud data only needs to record the address code and attribute value of the leaf node, and the memory consumption is small. Based on the linear octree construction of the point cloud, the three-dimensional array grid of the point cloud (X, Y, Z) is calculated through the coordinate value of the point cloud, which is the address code (code) of the node. The length of the address code represents the octree The resolution, the direction, and path information from the root node to a specified node is implicit in the code, so the octree addressing can be directly performed. The Morton code of each grid is calculated and sorted by the octal Morton code, so that block query, retrieval, and display operations can be performed according to Morton. For a three-dimensional object with a side length of 2n, the Morton code of any leaf node after the linear octree tree-making is an n-digit code, and its octal system can be expressed as:

\[ M = p_{n-1}8^{n-1} + p_{n-2}8^{n-2} + \ldots + p_08^0 \]
represents the number of the node in the sibling nodes of this layer, and is the number of the parent node of the node in its sibling nodes.

2.3 Point cloud organization model based on virtual grid and adaptive octree

The virtual grid better retains the original point cloud information, and the index construction is efficient and easy to code. The octree algorithm is simple, and the spatial scale division is suiT for the spatial distribution of three-dimensional point cloud data. In theory, it is the most efficient for data organization with relatively uniform spatial distribution in three dimensions, and its search efficiency has a greater relationship with the scale of the leaf node. Therefore, the above two data structures can be combined to divide the nodes with suiT magnitudes globally through the virtual grid, and then organize the evenly distributed nodes in an octree.

When querying and retrieving point clouds, read the point cloud bounding box information and node size information stored in the virtual grid index structure file and locate the node number where the index point is located according to formula 4, and then use the search radius R/l Calculate all the node numbers involved in the radius, read the corresponding node data in external memory through disk mapping, add it to the memory queue directly or through internal and external memory scheduling, and then pass the required node in the memory Construct an octree so that the search and location of neighboring points can be realized at a small memory cost.

2.4 Internal and external memory dynamic scheduling strategy

The original point cloud data is divided into multiple leaf node files according to the grid, and then the nodes loaded into the memory are organized in an octree. When performing data indexing, first calculate the node-set S involved in the search range. If all nodes in the set S have been loaded into the memory, move the nodes in the set S to the head of the LRU (Least Recently Unused) queue And organize the octree of nodes in S; if only some nodes in the set S are loaded into the memory, or the corresponding nodes are not loaded into the memory at all, use the node index ID in the virtual grid index structure file through the disk The mapping technology transfers the nodes corresponding to the set S from external storage to the memory, and at the same time judges whether the node data stored in the LRU queue exceeds the memory capacity limit, if it exceeds the memory limit, removes the end node data of the LRU team from the memory, And insert the newly added node into the head of the LRU queue, and then organize the octree organization of the nodes added to the set S in the memory.

3. Experiment and analysis
In order to test the effectiveness of the point cloud data index and organization strategy combined with the virtual grid and the octree, based on the Visual Studio 2010 platform, the C++ language is used to realize the functions of the virtual grid division, dynamic scheduling, and neighborhood query.

The experimental data is airborne lidar data in a certain area. The data format is las1.2. There are about 400 million point cloud data. After setting the block, the number of point clouds for each node is about 1 million, and 10 samples are randomly selected from it. To search, in order to verify its dynamic scheduling capability, RNN (Raidus Nearest Neighbor) search uniformly sets the radius to 50m, KNN (K Nearest Neighbor) search uniformly sets the neighborhood K to 10, and the average time is taken as the search time. Compare the data organization method in this paper (Method 1) with the single octree index organization method (Method 2).

The experimental running platform is Win7 64-bit operating system, and the main hardware configuration is Intel Core i7 and 7.25 G memory. The specific experimental results statistics are shown in Tab.1 and Tab.2:

| Quantity level | Data preprocessing time (ms) | Search time (ms) |
|----------------|-----------------------------|-----------------|
|                | reading | Grid block | mulberry time | RNN search | KNN search |
|----------------|---------|-------------|---------------|------------|------------|
| 5million       | 135.9   | 499         | 137.2         | 5.291      | 5.937      |
| 30 million     | 698.9   | 2649.3      | 214.65        | 5.81       | 6.155      |
| 1billion       | 1673.8  | 6681.2      | 152.1         | 5.018      | 5.318      |
| 4billion       | 475463.8| 40144.8     | 222.3         | 2.249      | 2.561      |

**Tab.1** Statistical analysis of method 1 experimental data

| Quantity level | Octree preprocessing time (ms) | Octree search time (ms) |
|----------------|-------------------------------|-------------------------|
|                | Read | Build time | RNN search | KNN search |
|----------------|------|------------|------------|------------|
| 5million       | 70.2 | 524.1      | 4.819      | 5.644      |
| 30 million     | 384.8| 3140.3     | 6.163      | 5.773      |
| 1billion       | 973.2| 8105.9     | 5.036      | 5.064      |
Tab. 2 Statistical analysis of method 2 experimental data

It can be seen from the above statistical information that the point cloud preprocessing time in Method 1 increases with the increase of the data volume, and point cloud reading and node storage account for most of the proportion, which also restricts preprocessing. Main factor: Under the condition of assigning equal weights, method 1 search preparation time (including disk mapping and data scheduling) slowly increases in the range of point cloud magnitude from 5 million to 400 million, when the point cloud magnitude reaches 400 million, the available memory of the computer is exceeded, there is a small "step" due to the dynamic scheduling of internal and external memory of the data. Compared with method 2 which is a full-memory data organization method, the biggest advantage of this method is that it breaks through the limitation of computer memory. the method in this paper takes the least time for a complete point cloud index, the search preparation time and memory consumption do not increase sharply with the increase of the amount of point cloud data, but gradually increase and stabilize at a certain level.

4. Summary

This paper designs a hybrid spatial index method that combines virtual grid and octree and combines internal and external memory dynamic scheduling strategies. In the whole world, the nodes of suiT magnitude are divided by the virtual grid, and then the nodes with relatively uniform distribution are organized in the octree. During data retrieval, first, quickly locate the grid range where the index point is located, and then transfer the point cloud in the grid range into the memory for octree organization. And the feasibility and efficiency of this method are proved through experiments of hundreds of millions of point clouds.

This article needs further improvement in that the database can be used to manage externally stored multi-node files to support efficient parallel read and write operations to increase the preprocessing speed, and combined with the powerful floating-point computing capabilities of the GPU to solve a large amount of data in the point cloud processing process. Simple and repetitive calculation problems.

References

[1] Dianpeng S, Fanlin Y and Bo S 2015 Real-time point cloud display technology of shipborne multi-sensor integrated measurement system J.Ocean mapping 35 pp 29-32

[2] Baoming Z, Haitao G, Z and Hongwei Z 2011 A DEM acquisition algorithm based on adaptive window filtering J. Ocean mapping 31 pp 28-30

[3] Jiansi Y 2014 A massive airborne LIDAR data organization and management method
combining quadtree and KD Tree J. Journal of Wuhan University 39 pp 918-921

[4] Hong X, Boyi W and Zhan Z 2013 Research on a new management method of mass point cloud data J. Remote sensing information 28 pp 26-33

[5] Wang M and Tseng Y H 2010 Automatic segmentation of LIDAR data into coplanar point clusters using an octree based split-and-merge algorithm J. Photogrammetric Engineering and Remote Sensing 76 pp 307-320

[6] Yongzhi W, Lusheng Y and Lixia L 2017 Laser point cloud data storage structure integrated with octree and three-dimensional R* tree J. Journal of Geoinformation Science 19 pp 588-591