ISPRS Journal of Photogrammetry and Remote Sensing 120 (2016) 25–36

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

The D-FCM partitioned D-BSP tree for massive point cloud data access and rendering

Zhang Yi

School of Geodesy and Geomatics, Wuhan University, Wuhan, Hubei, China

ARTICLE INFO

Article history:
Received 29 April 2016
Received in revised form 22 July 2016
Accepted 9 August 2016

Keywords:
Directional FCM
D-BSP tree
Spatial partitioning
Massive point cloud
KL transform

ABSTRACT

The spatial partitioning of massive point cloud data involves dividing the space into a multi-tree structure step by step, so as to achieve the purpose of fast access and to render the point cloud. The current methods are based on spatial regularity and equal division, which is not consistent with the irregular and non-uniform distribution of most point clouds. This paper presents a directional fuzzy c-means (D-FCM) method for irregular spatial partitioning. The distance metric is weighted by a direction coefficient, which is determined by the eigenvalue of the point cloud. The orientation of each node is adaptively calculated by principal component analysis of the point cloud, and Karhunen-Loeve (KL) transform is applied to the points coordinates in node. A binary space partitioning (BSP) tree structure is used to partition the point cloud data node by node, and a directional BSP (D-BSP) tree is formed. The D-BSP tree structure was tested with point clouds of 0.1 million to over 2 billion points (up to 60 GB). The experimental results showed that the D-BSP tree can ensure that the bounding boxes are close to the actual spatial distribution of the point cloud, it can completely expand along the spatial configuration of the point cloud without generating unnecessary partitioning, and it can achieve a higher rendering speed with less memory requirement.

1. Introduction

A point cloud is a set of data points in a coordinate system. To access and plot a massive point cloud with a huge amount of data is an important research topic. Point cloud is usually irregular and non-uniform in terms of the spatial distribution. Storing point cloud data is usually based on a sequential structure, which is inefficient for accessing and rendering. Therefore, to solve this problem, many scholars have studied different spatial partitioning techniques and data structures to better express point clouds.

Data structure and segmentation method are two kernel issues should be addressed in spatial partitioning. Conventional data structures include: (1) the octree, which intuitively divides the space into eight parts; (2) the binary space partitioning (BSP) tree, which divides the space into two parts; (3)the kdtree, which is a particular BSP tree whose super-plane is perpendicular to the axis; (4) the B-tree, which divides the space into multiple balanced parts; and (5) the R-tree, which is a natural high-dimensional extension of the B-tree. Bounding boxes of B-tree and R-tree are often overlapping, which reduces query efficiency. Therefore, recent studies have mainly focused on the octree and kdtree to solve point cloud data access and rendering.

Elseberg et al. (2013) proposed an efficient octree to store and compress one billion 3D data points without loss of precision. They implement octree by dividing an axis aligned cube into eight parts. Wand et al. (2008) described a new out-of-core multi-resolution data structure for real-time visualization, interactive editing, and efficient processing of large point clouds, using example data sets of up to 63 GB. Their basic data structure is regular octree. Gobbetti et al. (2008) presented an adaptive out-of-core technique for rendering massive scalar volumes employing single-pass GPU ray casting. Their method exploits the regular structure of octrees to reduce costly texture memory accesses by computing bounding boxes on-the-fly. Scheiblauer and Wimmer (2011) presented a method for the interactive selection of arbitrary parts of a point cloud using a so-called selection octree. They made some changes to the original Nested Octree data structure and replace them by regular grids. Zhu et al. (2007) and Gong et al. (2012) proposed a new spatial organization method called the 3DORtree, which integrates octree and 3D R-tree data structures. This method also equally splits space into eight parts. Oosterom et al.(2015) developed a benchmark (23 billion to 640 billion points) to compare various point cloud data management solutions based on Morton code

E-mail address: yzhang@sgg.whu.edu.cn

http://dx.doi.org/10.1016/j.isprsjprs.2016.08.002

computation of single cell and the Quad-Tree structures by bitwise interleaving x and y coordinates. Schön et al. (2013) implement a standard equal division octree index for 3D LiDAR data atop Oracle Spatial 11g.

Yang (2014) proposed a kind of hybrid index structure for organizing airborne LiDAR point cloud data to solve the problem of large-volume data organization and visualization, which combines a global quadtree with a local kdtree. The global quadtree is used to index the upper level of the point cloud data. The kdtree is used to index the data in a leaf of the quadtree. Experiments using 1 billion points were conducted which reached a rate of visualization of 30 fps. De et al. (2012) implement a balanced and regular octree. It only divides in the vertical dimension which behaves basically like a quadtree. And then orders the points by kdtree which divides the axis that shows the greatest extent. Mingyue and Yongjian (2008) introduced a nested structure integrating an octree and a binary tree to manage very large point cloud data sets.

Spatial partitioning is often implemented in a variety of software. Commercial software such as Leica Cyclone, Realworks Survey, RiSCAN PRO, and Polyworks use dynamic technology based on spatial partitioning. In the Cyclone export file, PTX is the original format of the sequential point cloud, while PTS represents the points after spatial partitioning. RiSCAN PRO uses kdtree to accelerate the display speed of the point cloud. Open source XGRT uses an octree to achieve real-time editing of point cloud data. The realization method utilizing an octree and kdtree is introduced in detail in the open source PCL.

No matter octree, kdtree or combined data structure, all the existing methods mentioned above are of axis aligned division and regular tree structure.

The core task of constructing an octree or kdtree is selecting the optimal segmentation position from the candidate segmentation plane to form the bounding boxes of sub nodes (Shan et al., 2009). Axis-aligned Bounding Boxes (AABB), Oriented Bounding Boxes(OBB) (Gottschalk 1996; Carvalho et al., 2003), Discrete Oriented Polytope(k-DOPs) (Klosowski et al., 1998) and Surface Area Heuristic (SAH) (Macdonald and Booth, 1990; Choi et al., 2009) are the most widely used methods, where polygon or triangular are the rendering primitives. But for point cloud, all the literatures mentioned above only referred to axis-aligned spatial partition. Oriented spatial partition for point primitive has not yet studied. The single types of spatial partitioning and the mixed types of spatial partitioning are both on the basis of spatial regularity and axis aligned division. However, this is not consistent with the irregular and non-uniform distribution of most point clouds. A good spatial partitioning should be able to express the spatial structure and directional distribution of the point cloud itself, and should also be adaptive.

Aiming at the irregular and non-uniform structure of point clouds, in this article, the direction information is introduced into the traditional fuzzy c-means (FCM), and a directional FCM (D-FCM) method for spatial partitioning is proposed. The direction of each node which contains a cluster of points is adaptively calculated by principal component analysis of the point cloud, and Karhunen-Loeve (KL) transform is applied to the points coordinates in node. A BSP tree structure is used to partition the point cloud node by node, and the directional BSP (D-BSP) tree is formed. As a result, the irregularity of the point cloud is characterized by the direction coefficient, while the non-uniform partitioning of the point cloud is completed by FCM.

2. The directional FCM model

Adaptive spatial partitioning is essentially an automatic classification problem. In this field, FCM is more suitable for point clouds because it is not certain which of the points in the point cloud are divided into which child nodes. This causes the fuzziness of the spatial partitioning. FCM can dynamically adjust the categorical attributes of each point by setting the membership degree rather than completely belonging to just one cluster as it is the case in the traditional c-means (CM), Although CM clustering can also be used to perform the spatial partitioning, the FCM resulting clusters are best analyzed as probabilistic distributions rather than a hard assignment of labels in CM. It should be realized that CM is a special case of FCM when the probability function used is simply one.

The traditional FCM model (Bezdek, 1981) is generally expressed by an objective function as follows:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\|p_i - c_j\|^2 = \min \tag{1}$$

where p_i is the i-th measurement, c_j is the cluster center of class j, $\|p_i - c_j\|^2$ is the distance metric, u_{ij} is the degree of membership of point i belonging to class j. The weight coefficient m controls the fuzziness of the clustering results. It is generally believed that the algorithm has the most practical significance when $m = 2$ (Bezdek, 1981). The clustering result can be achieved by solving the objective function.

For a 3D point cloud, the measurement of p_i is (x_i, y_i, z_i), and thus the corresponding objective function is:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left(\left|x_i - c_j^x\right|^2 + \left|y_i - c_j^y\right|^2 + \left|z_i - c_j^z\right|^2\right) = \min \tag{2}$$

The FCM model can solve the deficiency of dividing cube along axis in half. However, because the distance metric is based on an invariant coordinate system, the direction and the dividing surface are still fixed. In order to adapt the orientation of the point cloud, we must also consider how to introduce direction information. Among the exiting algorithms those introducing additional information, only GK (Gustafson and Kessel, 1978), AFCM (Wu and Yang, 2002) and RCP (Frigui and Krishnapuram, 1996) added coefficients to the distance metric. GK added a fuzzy covariance matrix, AFCM added an ambiguous parameter, and RCP added loss function. However, none of them are closely related to the direction.

The D-FCM method proposed in this article is an extension of FCM using directional weighting of the distance metric. It is an effectively directional FCM of the multi dimensional vector space. The objective function is defined as follows:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\left(a_1\left|x_i - c_j^x\right|^2 + a_2\left|y_i - c_j^y\right|^2 + a_3\left|z_i - c_j^z\right|^2\right) = \min \tag{3}$$

where a_1, a_2, a_3 are the directional weighting coefficients of the distance metric.

When $a_1 = 1, a_2 = a_3 = 0$, clustering is only implemented in the x-direction, and the corresponding division plane is $S(x, y, z) = x$ (Fig. 1a).

When $a_2 = 1, a_1 = a_3 = 0$, clustering is only implemented in the y-direction, and the corresponding division plane is $S(x, y, z) = y$ (Fig. 1b).

When $a_3 = 1, a_1 = a_2 = 0$, clustering is only implemented in the z-direction, and the corresponding division plane is $S(x, y, z) = z$ (Fig. 1c).

When $a_1 = a_2 = a_3 = 1$, clustering is implemented in the entire 3D space, and the corresponding division plane is $S(x, y, z) = x + y + z$ (Fig. 1d).

The form of the distance metric in Eq. (3) is the final expression of the division plane of D-FCM, and the corresponding division plane is $S(x, y, z) = a_1x + a_2y + a_3z$.