Optik 126 (2015) 2157–2162

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

Point cloud simplification with preserved edge based on normal vector

Huiyan Han*, Xie Han, Fusheng Sun, Chunyan Huang

School of Computer and Control Engineer, North University of China, Taiyuan, 030051, Shanxi, China

ARTICLE INFO

Article history:
Received 10 April 2014
Accepted 21 May 2015

Keywords:
Point cloud simplification
Normal vector
Importance value
Tangent plane

ABSTRACT

This paper presents a point cloud simplification algorithm with preserved edge based on normal vector. Because edge points have more distinct features than non-edge points, these special points should always be preserved in the point cloud simplification process. The proposed algorithm establishes the spatial topology relationship for each point using octree first, then identifies and retains edge points using a simple but effective method. For non-edge points, delete the least important points until user-specified data reduction ratio is reached. The importance of a non-edge point is measured using the average of the Euclidean distances (based on normal vector) from the point to estimated tangent plane at its each neighborhood point. The experimental results on three test point cloud data sets and two practical data sets of our own demonstrate that the proposed algorithm performs much better compared with other methods.

© 2015 Elsevier GmbH. All rights reserved.

1. Introduction

Surface model reconstruction based on point cloud data has a wide range of applications in the reverse engineering, computer animation and 3D object reconstruction. The point cloud data set often contains redundant points, and this severely limits model storage, transfer, drawing and other subsequent data processing tasks due to the significant computational complexity. Point cloud simplification becomes an indispensable means of data preprocessing. According to the principle of reduction, the commonly used point cloud simplification methods can be divided into two categories: mesh-based simplification and point-based simplification. Mesh-based simplification methods construct polygonal meshes (triangular meshes or quadrilateral meshes) first, finish the simplification task through meta-operations on meshes. Representative taxonomy and comparison of mesh-based simplification methods are available in the literatures [1,2]. The disadvantage on such methods is that the mesh generation is complex and cumbersome. Compared with mesh-based simplification methods, the direct point-based simplification methods without constructing polygonal meshes is a new research domain, more and more scholars contribute themselves to the study and have made great success. Some studies sampled the user-specified number of points using heuristics-based approaches [3–6], Song and Feng [7] constructed Voronoi diagram and proposed a global clustering approach with a specified data reduction ratio using geometric deviation criterion, simplification methods using Fourier theory [8], based on estimation on local variance [3], point delete algorithms [4,9], Bao et al. [10] proposed an adaptive simplification of point cloud using k-means clustering.

The above-cited point-based simplification methods do not differentiate edge points and non-edge points and simplify all points as a whole. Edge points have more important and distinct characteristics than non-edge points, so these special points should always be retained in simplification process. Demarsin et al. [11] detected sharp edge points based on the idea that the neighborhood of a sharp edge point consists of points from distinct surface patches. Song and Feng [12] proposed a simplification algorithm with preserved sharp edge data using the sharp edge point detection method [11]. Galantucci and Percoco [13] detected edge points in tessellated point clouds using multilevel approach. Gumhold et al. [14] extracted feature points from point clouds. Zongyue and Hongge [15] used Alpha-Shapes approach to extract edge contour line from massive point clouds. Libaoshun et al. [16] detected edge based on rolling-circle.

In this paper, on the basis of detection of edge points and estimation of normal vector on each point, we propose a point cloud simplification algorithm with preserved edge points. Octree is constructed to accelerate search speed of k-neighboring points, the least square plane of the neighboring points is fitted, edge points are detected using the projection relationship of the neighboring points to the least square plane, for non-edge points, delete

* Corresponding author. Tel.: +86 13703516841.
E-mail address: hhy980344@163.com (H. Han).

http://dx.doi.org/10.1016/j.ijleo.2015.05.092
0030-4026/© 2015 Elsevier GmbH. All rights reserved.

the least important points progressively until the simplification rate is reached. Compared with curvature-based simplification method, equidistant simplification method and random simplification method, the accuracy of the resulting simplification point cloud is improved, although the time complexity is higher but can be accepted. The proposed point cloud simplification algorithm is described in Section 2. In Section 3, the simplification performance of the new proposed algorithm is applied to practical point clouds and compared with other simplification methods, in Section 4, we conclude our paper.

2. The improved point cloud simplification algorithm

We construct octree and establish the vicinity points for each point, detect edge points using a simple but effective method, these edge points are retained in the decimating process, for non-edge points, measure the importance and delete the least important points progressively until the user-designed simplification rate is reached and we get the ultimate decimated point cloud.

2.1. Establishment of k-neighboring points and estimation of normal vector

Because the point cloud is often huge and disorganized, we construct the minimum space bounding volume cube to partition space and search the k-neighboring points using octree. The size c_size of the bounding volume cube is expressed in Eq. (1):

$$c\_size = \max(x, y, z) \quad (x = x_{\max} - x_{\min}, \quad y = y_{\max} - y_{\min}, \quad z = z_{\max} - z_{\min}) \tag{1}$$

where, $x_{\max}$ and $x_{\min}$, $y_{\max}$ and $y_{\min}$, $z_{\max}$ and $z_{\min}$ is the maximum and minimum x,y,z value respectively of the point cloud, and the minimum space bounding volume cube is the root node of the octree, divide the minimum space bounding volume cube into eight little cubes averagely, the eight cubes are the leaf nodes of the root node, divide each cube on the basis of this rule until the specified division level is reached as shown in Fig. 1. The octree is constructed as shown in Fig. 2. We can draw a conclusion that the octree has the following characteristics: If the octree is not null, any one node in the tree has not nodes or has eight child nodes. If the minimum bounding volume cube is divided into n layers, we can construct an octree that has $n+1$ *layer*, each cube corresponds to a unique node, the code Q of each node is expressed by Eq. (2):

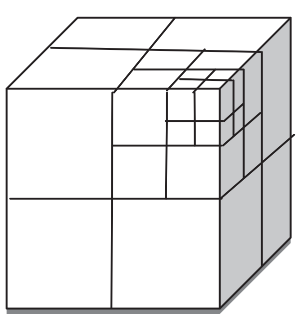

Fig. 1. Spatial division using octree.

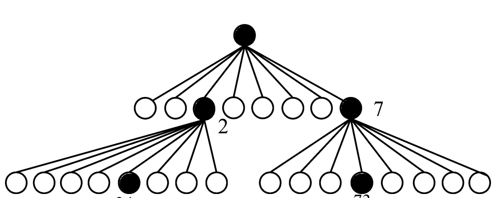

Fig. 2. Octree structure.

$$Q = q_{n-1} \cdots q_m \cdots q_1 q_0 \tag{2}$$

where, q_m is an octal number, $m \in \{0, 1, \cdots n-1\}$, if q_m is the sequence number of the node q in its brother nodes, then q_{m+1} is the sequence number of the parent node of q in its brother nodes, so we can use the code from q_0 to q_{n-1} to express the whole path from each leaf node to root node, the code of the boldface node in Fig. 2 are $\{2, 7, 24, 73\}$, respectively.

When search the k-neighboring points of point p, we can search the nearest k nodes of leaf node of p and the neighboring leaf node of p only, i.e., search the nodes in the cube(the cube that the node p locates in) and its 26 neighboring cubes, using the octree structure raises the neighboring points searching speed greatly.

Some point clouds may not contain the normal vector information, so we must compute the normal vector for each point. There are many methods to estimate normal vector, literature [17] discussed each normal vector estimation methods in detail, Ouyang and Feng proposed a novel algorithm based on fitted directional tangent vectors [18], but local Voronoi mesh partition added the computational complexity, Li et al. [19] proposed an approach based on an object function. We use the classical and simple PCA method [20] to estimate normal vector. Any point and its neighboring points can be fitted to a local plane using least square method. For a point p and its k-neighboring points in the point cloud, the fitted plane F is computed by Eq. (3), where n is the normal vector of plane F, d is the distance form F to the ordinate origin, $\bar{p}$ is the centroid and N_p is set of the k-neighboring points of point p.

$$F(n, d) = \operatorname{argmin}_{(n,d)} \sum_{i=1}^{k} (n\bar{p} - d)^2, \quad \bar{p} = \frac{1}{k} \sum_{i \in N_p} p_i \tag{3}$$

Plane F contains point p and normal vector n meets the requirement $\|\boldsymbol{n}\|_2 = 1$, so the problem is simplified to eigenvalue decomposition of the positive semi-definite covariance matrix M in Eq. (4), the eigenvector of the smallest eigenvalue is the estimation of normal vector of point p. The direction of normal vectors may be inconsistent, need to be adjusted using the virtual z-axis method.

$$M = \frac{1}{k} \sum_{i=1}^{k} (p_i - \bar{p})(p_i - \bar{p})^T \tag{4}$$

2.2. Detection of edge points

Various edge points detection methods have been proposed, and they use the topology relationship of the points and normal vectors to construct least square plane, project each point in the vicinity to the fitted plane and extract edge points according to the homogeneity of projection, the homogeneity of projection judgment criteria may be angular deviation and angular interval, but the computational complexity is high. The judgment criteria of homogeneity of projection of points in the vicinity to the plane employed in this paper is comparison of the coordinate values but not the angle.

Suppose p is a point in the point cloud, search its k-vicinity using octree, construct the least square plane and project the vicinity points to the plane, create three planes xpy, xpz and ypz, that parallel to plane xoy, xoz and yoz and go through point p, take planes xpy, xpz and ypz as reference planes, when the ratio of the difference of the number of the projections on the two sides of a reference plane and the sum number of the vicinity is bigger than a specified threshold, the point p is judged as an edge point. Take point p and the plane xpy for example in Fig. 3, there are 15 points above the

Download English Version:

https://daneshyari.com/en/article/848139

Download Persian Version:

https://daneshyari.com/article/848139

Daneshyari.com