An Effective Algorithm Research of Scenario Voxelization Organization and Occlusion Culling

Guangling Lai [1], Lu Ding [1], Zhiyuan Qin [1], Xiaochong Tong [1]

(1.Information Engineering University, Zhengzhou 450001, China)

E-mail: 923809012@qq.com

Abstract. Compared with the traditional triangulation approaches, the voxelized point cloud data can reduce the sensitivity of scenario and complexity of calculation. While on the base of the point cloud data, implementation scenario organization could be accomplishment by subtle voxel, but it will add more memory consumption. Therefore, an effective voxel representation method is very necessary. At present, the specific study of voxel visualization algorithm is less. This paper improved the ray tracing algorithm by the characteristics of voxel configuration. Firstly, according to the scope of point cloud data, determined the scope of the pixels on the screen. Then, calculated the light vector came from each pixel. Lastly, used the rules of voxel configuration to calculate all the voxel penetrated through by light. The voxels closest to viewpoint were named visible ones, the rest were all obscured ones. This experimental showed that the method could realize voxelization organization and voxel occlusion culling of implementation scenario efficiently, and increased the render efficiency.

1. Introduction

Voxel or three-dimensional pixels is short for volume pixels. Conceptually, voxel is to three-dimensional space what pixel is to two-dimensional space. It is the smallest unit of digital data in three-dimensional space, and mainly used in 3 D imaging, scientific data and medical imaging, etc. Compared to the traditional triangulation method, voxelized point cloud data is a kind of spatial-oriented data organization mode, the voxelized space is fixed which have constant space number, position, structure and relationships. This method could overcome some exist shortages, such as unable to express the object's internal structure; sensitivity to the complexity of the scene; when there are too many objects in the scene, triangulation efficiency is low and unable to implement dynamic scene update efficiently. This method owns a very important practical significance [1].

In order to improve the render efficiency without affecting the expression, visible cutting technology was usually used to express the voxel scenario. Depth buffer algorithm (Z-Buff) is mainly used to compare all the surfaces' depths of each pixels on the projective plane[2][3]. So that it could distinguish the visible surface, process each object surface of a polygon point by point and it is appropriate for the only polygon contained scene. However, two caches are needed in this method, and some unnecessary calculations were often executed; a pixel can only find one visible surface, and unable to deal with the transparent surfaces. A-Buff algorithm based on the Z-Buff, it establish a surface linked list for each pixel, not only can put each pixel's intensity values of multiple surfaces into consideration, and also can eliminate the aliasing of object's boundary[4]. BSP algorithm uses plane to divide the space, with the viewpoint position and the plane's visibility, divides the special

objects into two divided spaces, then process them with a recursion, and establish a visibility binary tree eventually. In the scene display, draw them in accordance from front to back. Key of this approach lies in the establishment of the tree structure. When scene or view point is changed, the tree structure needs to be rebuilt. So it was suitable for the static scene establishment[5]. Octree algorithm is a rather common spatial data organization structure[6]. When the observed object is described, the accordance from front to back was commonly used to map the octree nodes to the observed surface, and to eliminate the hidden surfaces. However, this approach is hard to manage dynamic scenes, when the scene is updated, the corresponding parts needed to be rebuilt either . Ray-tracing algorithm was originated from the ray-casting algorithm [7]; shade relations are mainly determined by means of the intersection of light and surface [8]. This method can realize real-time rendering of scene, but occlusion relations needs to be rejudged in each changing scene. And if there are more surfaces, it contained too many intersection times and larger calculated amounts by using triangular mesh to reconstruct the surface with large-scale scene point cloud data.

In order to implement the real-time updated display of dynamic scene, on the basis of advantages and disadvantages of the mentioned methods, this paper fully embodied the characteristics of scene regularization (each surface of voxel is paralleled to the coordinate plane, coordinate value could be quickly obtained, and it is easy for intersection calculation), and proposed a new ray-tracing algorithm.

2. Voxelization of point cloud data

The original point cloud data are some structurelessness discrete points. In order to be applied better, point cloud data needs to be organized. Method of this paper is to voxelized them, the main steps were as follows:

Step one: used comparative law to determine the whole point cloud data's minimum and maximum values in the three coordinate directions x, y, z:$x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$.

Step two: according to the distribution range and resolution requirements of the point cloud data, set the voxel size l;

Step three: use type (1) to judge each point position of point cloud data(x, y, z) on voxels. After calculated through all the points, only display the voxels which contained point cloud data. Then, voxelization of point cloud data was done.

$$\begin{cases} i = floor(\frac{x - x_{min}}{l}) \\ j = floor(\frac{y - y_{min}}{l}) \\ k = floor(\frac{z - z_{min}}{l}) \end{cases} \quad (1)$$

It can be seen by the whole process of voxelization, the level of similarity between voxelized scene and original scene mainly depends on the size of voxel, the smaller scale, the higher similarity. But a greater amount of data may lead a lower efficiency about the display of voxelized scene. In order to improve the efficiency of display, a common way is using occlusion cutting technology to insure occlusion relationship, reduce the draw number of voxel.

3. Occlusion cutting technology of voxel

In order to implement the real-time rendering of scene, this paper made full use of the regularity characteristics of voxelized scene, and then adopted the ray-tracing theory to determine the occlusion relationship. First, calculate light vector corresponded to each image point in the world coordinate system. Then, get the intersections of light and voxels, determine all the voxels which penetrated by light. The voxels which closest to viewpoint and contained point cloud data were named visible ones, the rest were all obscured ones. In the end, drew the visible voxels only. From the above principles, it could be seen there were three factors that influenced the efficiency of ray-tracing algorithm[9]: (1) the number of light; (2) intersection speed; (3) the number of intersection. Therefore, the efficiency could be improved by exacting the scope of pixels, reducing the amount of light; and took advantage of the

regular distribution of voxels to reduce the number of intersection and improve the intersection speed. Specific methods were as follows.

3.1. Calculation of light vector

Use ray-tracing algorithm to determine the occlusion relationship. First, we need to calculate and get the light vector which refers to a direction vector in the world coordinate system, which started from the coordinate of screen pixel and ended in the coordinate of camera's position.

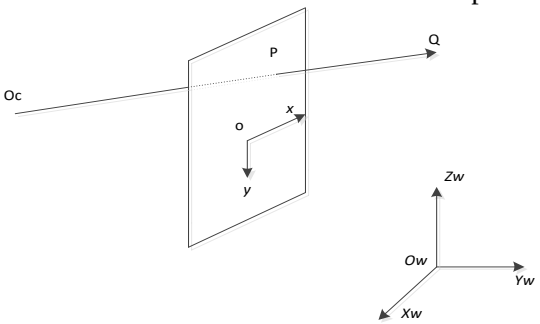

Figure 1 light vector diagram

In figure 1 O_c（x_c, y_c, z_c）stands for the coordinate of camera in the world coordinate system $O_w - X_w Y_w Z_w$, $o - xy$ coordinates refers to the image plane coordinate system, $p(x_p, y_p)$ represents a pixel, $\mathrm{Q}(x_q, y_q, z_q)$ is the coordinate of point p on the world coordinate system. So the light vector $\vec{n}(a, b, c)$crossed pixel point p was calculated by the following formulas:

$$sum = \sqrt[2]{(x_c - x_q) * (x_c - x_q) + (y_c - y_q) * (y_c - y_q) + (z_c - z_q) * (z_c - z_q)} \quad (2)$$

$$\vec{n} = (x_c - x_q, y_c - y_q, z_c - z_q)/sum \quad (3)$$

3.2. Confirmation of the pixel range

In the visual expression of the scene, not each pixel on the screen contains scene information. Thus, determine the effective pixel range on screen can reduce the amount of light, so as to improve the efficiency.

Using comparative method to determine the minimum and maximum of all the point cloud data in x, y, z coordinate directions: $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$, then convert $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$ in world coordinate system to screen coordinate system. The range determined on screen is the range of pixels which used for ray-tracing.

3.3. Visibility judgment

Key link of ray-tracing algorithm was to determine the whole voxels went through by light, as shown in figure 2.

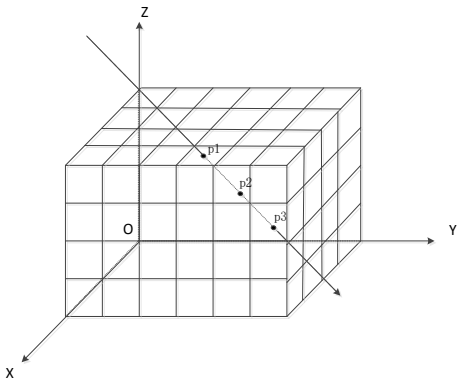

Figure 2 occlusion cutting diagram

The way how to confirm the voxels which went through by light was as follows.

Set the direction vector of light vector $\vec{n} = (a, b, c)$, $Q(x_q, y_q, z_q)$was set to any pixels' coordinate that converted to the world coordinate system, $O_c(x_c, y_c, z_c)$was camera's position. ON the basis of regularity characteristic of voxel, the cross point intersected by light and a voxel's upper surface or lower surface (as the points $p1$ and $p2$ shown in figure 2) could all be calculated. The formula is shown in (4)-(7), (x_i, y_i, z_i) stands for intersection coordinates. By computing the intersection in formula (1), voxels went through by a vertical direction light could be determined.

$$z_i = z_{max} - i * l \quad (i = 0,1,2, \ldots.) \tag{4}$$

$$k = (z_i - z_q)/c \tag{5}$$

$$x_i = x_q + k * a \tag{6}$$

$$y_i = y_q + k * b \tag{7}$$

l stands for length, width and height of a voxel in formula (4).

After obtaining voxels in z direction, we still need to judge the voxels passed by light in each layer. And this problem could be treated as a two-dimensional plane problem, as shown in figure 3.

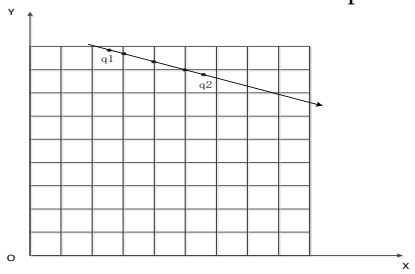

Figure 3 Occlusion cutting plane sketch

Based on z, according to the equally distributed characteristics of voxel, calculate the cross points intersected by light and two adjacent planes, mark them(x_1, y_1, z_1) and(x_2, y_2, z_2) (as $p1$ and $p2$ shown in figure 2). Projected these two points on the planexy, took advantages of the projection relationship to obtain the coordinates $q1(x_1, y_1)$ and $q2(x_2, y_2)$. (a, b) is the direction vector of light on the xy plane. Thus, cross point intersected by light and each grid boundary could be obtained.

For example, net values in y direction were an arithmetic progression, so the following could be obtained.

$$y_k = k * l + y_{min} \tag{8}$$

$$x_k = a * (y_k - y_1)/b + x_1 \tag{9}$$

In formula (8), $k = 0,1,2$ ... until x_k, y_k is smaller than x_2, y_2.

After x_k, y_k was got, according to the fixed z value and formula (1), voxel corresponded to this point could be calculated, and that is also the voxel passed by light.

Change the value of z until the point cloud contained voxel had been found, then went ahead to determine another occlusion relationship of voxel by next light.

4. The experimental results and analysis

This experiment used a city point cloud data, with its range was a cuboid with 1740.25 meters long, 1723.75 meters wide, and 183.14 meters high. As shown below in figure 4.

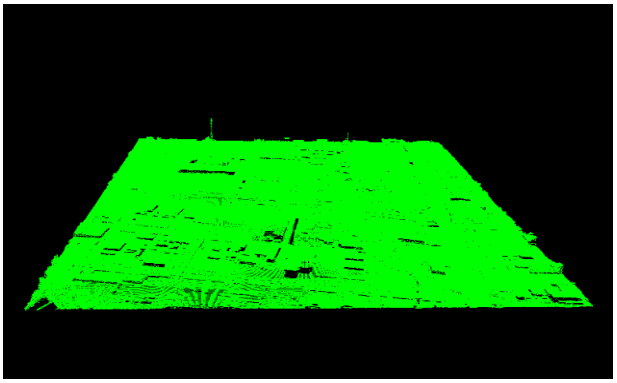

Figure 4 city point cloud data

After voxelizing the point cloud data with the method mentioned before, the result was shown in figure 5. To make sure the accuracy of ray-tracing algorithm, projection of a single voxel shouldn't be less than a pixel. In the experiment, size of each voxel was 8 meters, the resolution is less. Smaller voxel could make the voxelized scene as similar as the actual scene; their occlusion relationship remains the same.

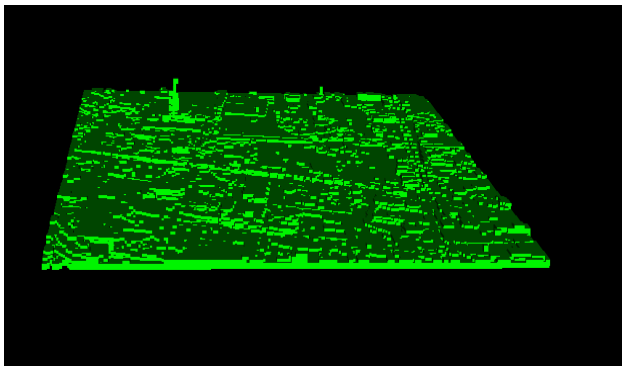

Figure 5 voxelized point cloud scenes

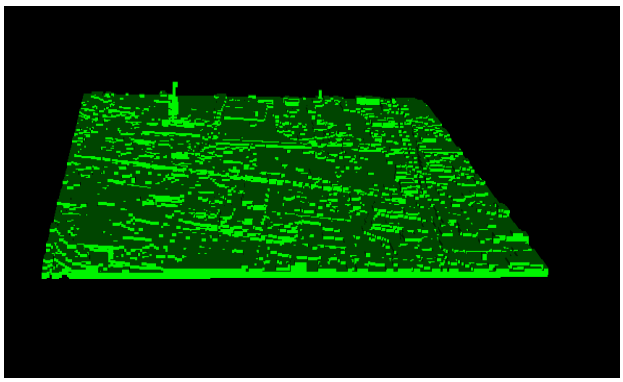

Figure 6 occlusion cutting results

Figure 5 represents the voxelized result; and figure 6 showed the visible voxels after confirming occlusion and cutting relationships. Specific quantities of occlusion and cutting were shown in Table 1.

Table 1 statistics of the occluded voxel

The total number of voxel	The visible number of voxel	Occlusion rate
39713	31722	20.12%

It could be told that, without influencing the expressive effect, method in this paper reduced the draw number of voxel efficiently, and improved the efficiency.

5. Conclusion

In this paper, voxelization about city point cloud data of a big scene was made. And on this fundament, made a full use of the regular distribution character of voxel; and adopted a modified ray-tracing method in occlusion and cutting study. The experimental results indicated that the proposed method could implement the voxelization of scene quickly; on the basis of voxelization, it could be very efficient to determine the occlusion relationship between voxels, reduced the draw number, and improved the draw efficiency.

References

[1] LI J 2005 Voxel imaging technology and application *Harbin Engineering University Press* pp 6-20

[2] Greene N, Kass M and Miller G 1993 Hierarchical Z-buffer visibility *Conference on Computer Graphics and Interactive Techniques* 27 pp 231-238

[3] Greene, Ned, Kass and Michael 1994 Error-bounded antialiased rendering of complex environments *Conference on Computer Graphics and Interactive Techniques* pp 59-66

[4] Schilling A, Stra&# and Er W 1993 EXACT: algorithm and hardware architecture for an improved A-buffer *Conference on Computer Graphics and Interactive Techniques* pp 85-91

[5] Fuchs H, Kedem Z M and Naylor B F 1980 On Visible Surface Generation by a Priori Tree Structures *Acm Siggraph Computer Graphics* 14 pp 124-133

[6] Egenhofer and Max J 1991 Book Review: The design and analysis of spatial data structures. Hanan Samet: Addison-Wesley Publishing Company, Reading, MA, 1989, 493 pp. ISBN 0-201-50255-0 *Isprs Journal of Photogrammetry & Remote Sensing* 46 pp 49-51

[7] Wald, Ingo, Kollig, Thomas, Benthin and Carsten 2002 Interactive global illumination using fast ray tracing *EG Workshop on Rendering Eurographics Association* pp 15-24

[8] ZHOU P 2012 Reach onray tracing based photorealistic global illumination issues *Shandong University* pp 3-10

[9] ZHAO S and LI X 2006 Algotithm of efficient ray tracing of complex scenes *Computer Engineering* 32 pp 224-225