Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 1, No. 2, 2022

Large-scale Aerosols Visualization Based on the Accelerated Ray Casting Algorithm

Zefan Qi [1, *]

[1] College of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China.

[*] **Corresponding author**: Zefan Qi (Email: 1269289295@qq.com)

Abstract: The ray casting algorithm is a widely used basic volume rendering algorithm, but traditional ray casting algorithms have some disadvantages such as low efficiency, slow speed and large resource consumption when rendering aerosol data such as smoke, fog and dust in 3D scenes. In this regard, this paper proposes a large-scale aerosol visualization method based on a ray-casting acceleration algorithm. A large part of the computational work in a traditional ray casting algorithm is to compute a large volume of data from the underlying scalar field. Therefore, the accelerated design of the ray casting algorithm should first consider avoiding sampling of empty spaces and invalid voxels, and culling areas in the volume data that do not contribute to volume rendering. The method in this paper firstly optimizes and accelerates the traditional ray casting algorithm through the hierarchical bounding box tree, and combines the illumination model to enhance the color rendering of each voxel. Unnecessary space debris reduces hardware load, improves rendering speed, and meets the real-time rendering requirements of large-scale aerosols.

Keywords: Volume rendering; Ray Casting; Empty Voxel Jump; Bounding Volume Hierarchy; Aerosols Visualization.

1. Introduction

The Volume rendering [1] technology is a technology that displays all volume details on a two-dimensional image at the same time based on three-dimensional volume data. Using volume rendering technology, the comprehensive distribution of various substances can be displayed in one image, and the condition of the isosurface can be reflected by controlling the opacity. Volume rendering is widely used in medical CT imaging, real-time meteorological observation, geological survey, molecular biology scanning imaging, aerospace and other fields. Today's volume rendering research field mainly focuses on how to improve the quality of the rendered image and how to reduce the rendering time of the algorithm. Usually, the rendering quality is positively correlated with the rendering time. How to reduce the algorithm rendering time as much as possible on the basis of ensuring the image quality is one of the hotspots in the current volume rendering technology research. Among many volume rendering algorithms, the ray casting algorithm calculates the highest image quality, but its algorithm complexity is higher than other volume rendering algorithms, and the rendering time is longer.

In order to improve the rendering speed of visualization, many volume rendering acceleration algorithms have been proposed. There are two commonly used methods based on software acceleration: acceleration through function optimization, such as multi-resolution based on wavelet transform [2], based on early opacity cutoff [3], based on parallel and distributed volume rendering [4], Interactive classification based on density-distance map [5], combined with graphics hardware acceleration and other algorithms, but these methods do not filter the data, and the acceleration is not complete; by processing the data to improve the drawing speed, such as based on the adjacent layers Similarity [6], based on bounding box space [7], based on octree [8], k-d tree [9] and other data structure methods, but these methods do not accelerate the algorithm process, and there is still a lot of acceleration time. room for improvement.

On the basis of related research, this paper optimizes function changes and data processing at the same time. By researching the use of algorithms to eliminate invalid data and empty voxel jumps when performing volume rendering, the rendering time is accelerated, and the image quality is guaranteed, while significantly improving Algorithm acceleration effect.

2. Ray Casting Algorithm

Ray Casting [10] is a direct volume rendering technique based on image sequences. The calculation of the ray casting algorithm does not stop at the surface of the object, but samples the interior of the object along the ray, and does not generate secondary rays. It draws volume data in a 3D scalar data field into a 2D image through scientific computing by tracing rays from the viewpoint to the object. The basic steps are to start from each pixel on the screen and emit a ray along the line of sight. When the ray passes through the volume data, it is sampled equidistantly along the ray direction, and the optical properties of the sampling point are calculated by interpolation (Such as color value and opacity); then according to the order from front to back (Front-to-back) or from back to front (Back-to-front), the sampling points on the ray are synthesized, and the corresponding value of this ray is calculated. The color value of the pixel on the screen.

The traditional ray casting algorithm does not filter the initial data, but draws the sampling points one by one layer by layer, which makes the process of calculating the ratio of each voxel to the screen pixel and so on consumes computing resources and time. In this regard, the acceleration design of ray casting algorithm is discussed from three aspects: sampling point optimization, empty voxel jumping method and the combination of the two improved methods.

3. Empty Voxel Skipping

The purpose of empty voxel skipping is to exclude from

volume data sampling the regions in which it does not contribute to the output image. A state-of-the-art image-sequential volume rendering method, GPU-based octree-accelerated ray casting [11], traces each ray from one tree node to another. Nodes marked as empty can simply be skipped, and each ray is computed independently, without exploiting consistency between rays or between successive rendered frames. More importantly, for sparsely structured volume data, the octree subdivides the space very finely, and many spatially adjacent nodes will have the same type (empty or not). Advanced tree refinement forces ray traversal to occur in unnecessarily small spatial increments. This huge overhead makes empty voxel skipping inefficient and significantly reduces the potential performance improvement. Spatial segmentation based on K-D tree [12] can better adapt to the spatial subdivision of sparse structure, but the subdivision based on K-D tree is very dependent on the spatial position of non-empty voxels. If the dataset of segmented objects changes frequently, the K-D tree cannot change in real time according to new spatial features. Hierarchical bounding box tree (BVH) [13] can perform efficient space segmentation, fully consider the area and shape of the volume data itself, filter out a large amount of unnecessary blank space, and can be dynamically updated.

4. Ray Casting Acceleration Algorithm

4.1. Early termination of light

Use an early termination condition to speed up the algorithm, i.e. stop the compositing operation when the cumulative opacity exceeds 1. In the process of pixel mixing iterative solution, the value of opacity a is gradually increased. When the value of a is close to 1, it means that the calculation result of the pixel value by the light tends to be saturated, even if the sampling continues, the final pixel value almost unaffected. Therefore, the method of terminating the ray calculation early can be used, which can improve the overall drawing time. A threshold can be set. As long as the value of a exceeds this threshold, the ray will be terminated. At the same time, combined with the adaptive accelerated sampling method, that is, when the ray traverses the volume data, if it encounters an invalid sampling point, the sampling interval is appropriately increased. Reduce the resampling frequency, and restore the resampling frequency if a valid sampling point is encountered. The overall rendering time can be further improved by combining early ray termination with adaptive sampling acceleration.

4.2. Volume data sub-components

4.2.1. OC tree

In the case of a large amount of data, it is more difficult to use the determination of the minimum granularity (leaf node). When the granularity is large, the data volume of some nodes may still be relatively large, and the subsequent query efficiency is still relatively low. On the contrary, the granularity is small. The depth of the fork tree increases, the required memory space is relatively large (each non-leaf node requires eight pointers), and the efficiency is also reduced. The division basis of equal division makes the efficiency not too high due to the limitation of division depth when the data center of gravity is skewed.

The empty voxel jumping method based on the octree structure, that is, recursively divides each block of volume data into eight parts, and records the in and out points of each block of non-empty data that the light passes through to sample. However, the octree-based method will additionally record many empty nodes, which increases the computational complexity of sampling.

4.2.2. AABB bounding box tree

The hierarchical bounding box is referred to as BVH; and the hierarchical bounding box tree is a binary tree used to store the shape of the bounding box. Its root node represents the largest bounding box, and the next 2 child nodes represent 2 sub-bounding boxes.

The AABB bounding box is an axisymmetric bounding box, that is, the six faces of the box are parallel to the xy, xz, yz planes

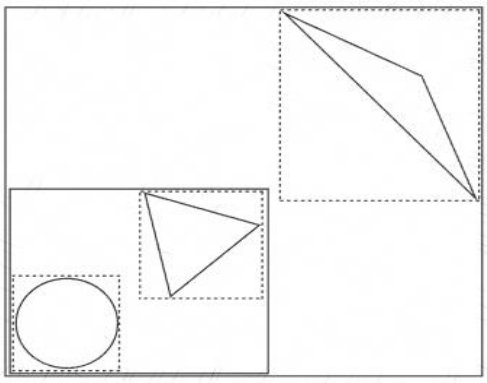


Figure 1. Bounding Volume Hierarchy

Subdivide the body data with an improved bounding box structure, with each leaf node corresponding to a subblock of data. In the sub-division process, each sub-data block needs to be expanded by a voxel to avoid errors during boundary interpolation (the value obtained by interpolation may be greater than the maximum value of the original sub-data block or smaller than the minimum value of the original sub-data block.), and then count the minimum and maximum values of each sub-data block, and store them in the original bounding box structure for generating 3-dimensional textures.

4.2.3. Improvement to empty voxel skipping

Recursively divide each block of data into eight parts to form an octree. For the volume data block of each leaf node, the marked attribute is one of empty, non-empty, and unknown; for each non-leaf node, the attribute proportion of its child nodes is counted, and its attribute is assigned as proportion highest attribute.

Perform a breadth-first search on the octree of the volume data, and construct a bounding box for the volume data blocks whose attributes are different from their parent nodes.

For each ray, when it passes through the bounding box, it is merged and deleted according to the rules to determine the sampling range. Volume rendering of ray casting algorithm using lazy loading method

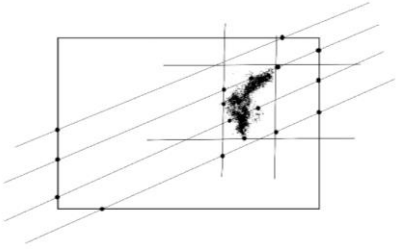


Figure 2. Empty Voxel Skipping

4.2.4. Grouped linear interpolation

The traditional linear interpolation calculation process is

quite complicated and consumes a lot of computing resources, which is the main reason why the traditional ray casting algorithm cannot draw in real time

In the interpolation calculation stage, all resampling points on a ray are divided into 3 levels according to the distance from the viewpoint, and different linear interpolation is used for each level.

Compared with the traditional algorithm, the hierarchical grouping linear interpolation method does not reduce the number of resampling points and does not reduce the authenticity of the image. According to the different positions of the resampling points on the light, it gradually decreases from the first level to the third level. The amount of interpolation calculation of each level of resampling point greatly shortens the total interpolation calculation time and achieves the purpose of improving the rendering efficiency of the algorithm.

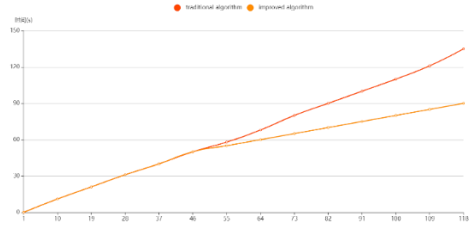
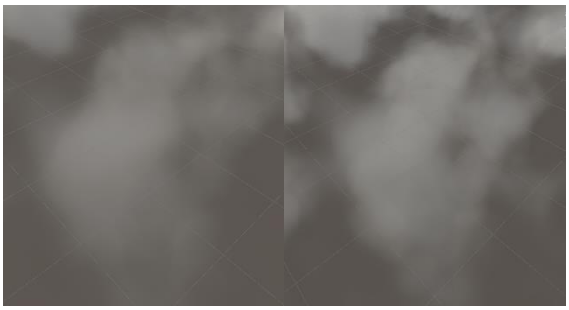


Figure 3. Computation time comparison

5. Experiment Analysis

In this experiment, the computer configuration is AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz, Windows 10 64-bit, 16GB memory. The programming environment is Visual Studio 2019 and Unity3D, using C# programming and HLSL shading language.



(a) traditional ray casting **(b)** accelerated ray casting

Figure 4. Rendering Effect Comparison

Aerosol particles have the characteristics of uneven distribution, small variation scale and complexity. In this paper, the cloud is used as a representative to conduct experiments. Comparing the algorithm in this paper with the traditional ray casting algorithm, it can be seen that under the same cloud data. Compared with the traditional ray casting algorithm, the acceleration algorithm in this paper is better in rendering results, with more details, more obvious diffuse effect, and more obvious distinction between light and dark. At the same time, the acceleration algorithm in this paper has better performance in terms of accuracy and realism.

6. Conclusion

In order to solve the shortcomings of traditional ray-casting algorithm when drawing aerosol data such as smoke, fog and dust in 3D scenes, it has some disadvantages such as low efficiency, slow speed and large resource consumption. This paper proposes a large-scale aerosol visualization method based on ray casting acceleration algorithm, optimizes resampling, performs hierarchical grouping operations on the sampled resampling points, and uses the hierarchical bounding box algorithm to segment the space, and then uses empty voxels to jump. The method culls empty space and invalid voxels, avoids unnecessary space fragmentation, reduces hardware load, and improves rendering speed. Comparing the algorithm in this paper with the traditional algorithm, in terms of rendering speed, with the increase of the amount of data, the advantage of the algorithm in this paper is also increasing. At the same time, the algorithm in this paper has better performance in terms of authenticity and accuracy. The experimental results show that the algorithm in this paper is feasible, effective and stable.

References

[1] Pravin P Kalyankar,S S Apte. 3D Volume Rendering Algorithm[J]. International Journal of Engineering and Advanced Technology (IJEAT),2013,2(5).

[2] Hassan A H ,Fluke C J , Barnes DG. A Distributed GPU-based Framework for real-time 3D Volume Rendering of Large Astronomical Data Cubes[J]. Publications of the Astronomical Society of Australia, 2012, 29(3):340-351.

[3] Johanna, Beyer, Markus, et al. State-of-the-Art in GPU-Based Large-Scale Volume Visualization[J]. Computer Graphics Forum, 2015, 34(8):13-37.

[4] Bin Ma,Yaohe Liu,Hyder Abbas. Parallel Volume Rendering Algorithm of Volume Mineralization Model based on GPU[J]. International Journal of Performability Engineering,2018,14(8).

[5] Takanashi, I. , Lum, E. B. , Ma, K. L. , & Muraki, S. (2002). ISpace: Interactive Volume Data Classification Techniques Using Independent Component Analysis. Pacific Conference on Computer Graphics and Applications. IEEE Computer Society.

[6] Lachlan J.Deakin,Mark A.Knackstedt.Efficient ray casting of volumetric images using distance maps for empty space skipping[J].Computational Visual Media,2020,6(01):53-63.

[7] Liu Baoquan,Clapworthy Gordon J,Dong Feng, Prakash Edmond C. Octree rasterization: accelerating high-quality out-of-core GPU volume rendering.[J]. IEEE transactions on Bernhard Finkbeiner,Alireza Entezari, Dimitri Van De Ville,Torsten Möller. Efficient volume rendering on the body centered cubic lattice using box splines[J]. Computers & Graphics,2010,34(4).

[8] Zellmann S , Schulze J , Lang U . Binned k-d Tree Construction for Sparse Volume Data on Multi-Core and GPU Systems[J]. IEEE transactions on visualization and computer graphics, 2021, 27(3):1904-1915.

[9] Levoy M . Display of Surfaces from Volume Data[M]. IEEE Computer Society Press, 1988.

[10] Francisco Sans,Rhadamés Carmona. A Comparison between GPU-based Volume Ray Casting Implementations: Fragment Shader, Compute Shader, OpenCL, and CUDA[J]. CLEI Electronic Journal,2017,20(2).

[11] Foley T , Sugerman J . Graphics Hardware (2005) M. Meissner, B.- O. Schneider (Editors) KD-Tree Acceleration Structures for a GPU Raytracer. 2008.

[12] Li T , Wang S , Cheng H , et al. Integral Image Generation Based on Improved BVH Ray Tracing[M]. 2021