A Volume Rendering Acceleration Algorithm Based on Inter-layer Similarity and Empty Voxel Skipping (Postprint)

Authors: Mount Ao, Liu Mengying, Li Baokun, Liu Zhizhong

Date: 2020-09-28T00:00:00+00:00

Abstract

Splatting is a classic object-order direct volume rendering method whose image rendering speed is constrained by the volume of computational data. To further enhance rendering speed, an acceleration method combining inter-layer similarity and empty voxel skipping is employed. During the data reading process, 3D texture data within the images is filtered, and a footprint table is utilized to perform 2D projection of the filtered 3D texture data. Inter-layer similarity is exploited to calculate the grayscale value of each point, and data is classified according to grayscale values to identify empty voxels that have no effect on imaging, thereby skipping their rendering process to accelerate the algorithm. Experimental results demonstrate that the proposed algorithm can, while guaranteeing image rendering quality, to a certain extent address and improve the issues of spatial correlation and computational efficiency in Splatting algorithm data.

Full Text

Preamble

Research on Accelerating Volume Rendering Algorithm Based on Similarity Between Adjacent Layers and Void Voxel Jump

Ao Shan, Liu Mengying, Li Baokun, Liu Zhizhong
(School of Computer Science & Technology, Henan Polytechnic University, Jiaozuo 454000, China)

Abstract: Splatting is a classical direct volume rendering method based on object order, where the amount of computational data restricts image rendering speed. To further improve rendering speed, this paper proposes an acceleration method combining adjacent layer similarity with void voxel jumping. During

data reading, three-dimensional texture data in images is filtered, and footprint tables are used to project the filtered 3D texture data onto a two-dimensional plane. Adjacent layer similarity is employed to calculate the gray value of each point, and data is classified according to these gray values to identify void voxels that have no effect on imaging, thereby accelerating the algorithm by skipping their rendering process. Experimental results demonstrate that the proposed algorithm can solve and improve the spatial correlation and operational efficiency issues of the Splatting algorithm to a certain extent while ensuring image quality.

Keywords: volume rendering; splatting algorithm; footprint method; comparability of adjacency-layers; void voxel

0 Introduction

Volume rendering is widely applied in medical CT imaging, real-time meteorological observation, geological surveying, molecular biology scanning imaging, aerospace, and other fields. Different application domains correspond to different volume rendering algorithms with varying emphases, such as image-space ray casting, object-space Splatting, shear-warp algorithms, and frequency domain volume rendering. After years of development, current volume rendering research focuses on two main directions: improving image quality and reducing rendering time. Typically, rendering quality and rendering time are positively correlated, making it a key research topic to minimize algorithm rendering time while ensuring image quality. Among various volume rendering algorithms, ray casting produces the highest quality images but has higher algorithmic complexity and longer rendering times compared to other methods. While Splatting does not achieve the same quality as ray casting, its rendering time is relatively shorter.

With the development of computer technology, numerous volume rendering acceleration algorithms have been proposed to improve visualization rendering speed, primarily divided into hardware and software approaches. Hardware-based acceleration depends on computer hardware development, and the emergence of graphics processors has provided hardware support for solving real-time 3D visualization problems. Utilizing GPU programmability to accelerate volume rendering algorithms has become a mainstream research direction, but these methods do not improve the algorithm's fundamental principles. Software-based acceleration methods currently employ two common strategies: first, acceleration through function optimization, such as multi-resolution based on wavelet transforms, early opacity termination, parallel and distributed volume rendering, density-distance graph-based interactive classification, and Splatting algorithms combined with graphics hardware acceleration. However, these methods do not filter data, making their acceleration incomplete. Second, acceleration through data processing, such as methods based on adjacent layer similarity, bounding box space, octrees, and k-d trees. However, these methods do not accelerate the algorithm process itself, leaving significant room for improvement in rendering

time.

Building upon related research, this paper simultaneously optimizes function modification and data processing. By studying the elimination of invalid data and void voxel jumping during volume rendering with the Splatting algorithm, rendering time is accelerated. A filtering method for data gray values is designed to significantly improve algorithm acceleration while maintaining image quality.

1 Classic Splatting Algorithm

Splatting is a classic volume rendering algorithm based on spatial data scanning. It calculates each data point' s contribution to screen pixels layer by layer, row by row, and point by point using a footprint function, projects it onto a two-dimensional plane through the algorithm, and composites it to form the final image. The Splatting algorithm first represents volume data as a matrix of overlapping basis functions, then calculates the influence range of each voxel' s projection by substituting each voxel into the footprint function, which is the process of image reconstruction.

Using a conical function to define intensity distribution, the overall contribution of each voxel to the image is calculated, thereby transforming the input image data into image space. The influence value of each voxel on image pixels is matched through a pre-computed universal footprint table obtained during data preprocessing and then composited to form the final image.

In the classic Splatting algorithm flow, reconstruction is the process of reconstructing a discrete three-dimensional data field into continuous data using convolution operations based on a reconstruction kernel. Its mathematical expression is:

$$D_{signal}(x,y,z) = \sum_{(x,y,z)\in\rho} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_v(u-x, v-y, w-z)\cdot\delta(x,y,z)\,du\,dv\,dw$$

where h_v is the reconstruction kernel; u, v, w are the coordinates of the reconstruction kernel; ρ is the density function of the 3D data field; and δ is the comb function used for sampling.

Traditional volume rendering algorithms do not filter the initial data during loading, rendering sampling points layer by layer, making processes such as footprint table lookup and contribution value calculation for each voxel extremely computationally intensive and time-consuming. To address these two issues, this research accelerates computation speed through adjacent layer similarity and void voxel jumping methods, discussing Splatting algorithm optimization from three aspects: adjacent layer similarity, void voxel jumping, and the combination of both improvements.

2.1 Adjacent Layer Similarity

In the Splatting algorithm' s convolution formula, triple integral operations are difficult to complete during the reconstruction process of large data fields. However, 3D data field reconstruction can also be accomplished through Equation (2):

$$D_{signal}(x,y,z) = \sum_{(x_i,y_i,z_i)\in\rho} h_v(x-x_i, y-y_i, z-z_i)$$

where (x_i, y_i, z_i) are sampling points within the reconstruction kernel centered at (x, y, z).

Considering only one sampling point' s influence on multiple points in image formation space, when calculating the contribution of voxel (x_i, y_i, z_i) to screen point (x, y), it can be viewed as integrating the contribution of voxel (x_i, y_i, z_i) to point (x, y) along the z-axis:

$$effect(x,y) = \int_{-\infty}^{\infty} h_v(x-x_i, y-y_i, w)\, dw$$

Examining Equation (3), the integral can be extracted, and its magnitude depends only on the distance to the reconstruction kernel center point. Therefore, the footprint function can be defined as the mapping distance between the pixel point in the footprint function and the reconstruction kernel center point on the image.

When calculating a voxel' s influence on the image, weighted calculations can be quickly performed by looking up the footprint function stored in a table calculated during the data preprocessing stage to determine the voxel' s impact on the image.

Through observation, the footprint function is only affected by the selected reconstruction kernel. With the footprint function determined, the gray value of a pixel point within the footprint function' s coverage area can be calculated:

$$footprint(x,y) = \int_{-\infty}^{\infty} h_v(x,y,w)\, dw$$

where (x, y) are the projected point coordinates of the sampling point on the image plane.

Literature [10] samples 3D volume data with an interval of approximately 1mm, which is relatively small, to determine similarity between adjacent layer data. Let the 3D volume data discrete sampling set be (i, j, k) where $1 \le i, j \le M, 1 \le k \le K$, assuming the same resolution in the x and y directions, which is true in

most cases. The density (energy) values of two adjacent layers are: $(r_{i,j,k})$ and $(r_{i,j,k+1})$. According to Equation (5), their contributions to screen pixels are:

If there is no density value change at point (i, j) between two adjacent layers L_k and L_{k+1}, the calculated contribution to screen pixels remains unchanged. If the density value changes, the contribution to screen pixels changes accordingly. From Equation (8), the magnitude of change is equivalent to the contribution of a point with density value $\Delta r_{i,j,k}$ to screen pixels.

Assuming all data points in layer L_k and their contributions to screen pixels have been calculated and stored in $effect_{i,j,k}$, with screen image resolution $M \times M$, when calculating the contribution rate of all data points in layer L_{k+1} to screen pixels, if the density value at point $(i, j, k+1)$ in layer L_{k+1} has not changed compared to point (i, j, k) in layer L_k, then all $effect_{i,j,k}$ can be reused. If the density value at point $(i, j, k+1)$ in layer L_{k+1} has changed compared to point (i, j, k) in layer L_k, only the data within that point' s influence range (determined by the footprint table) needs to be modified in $effect_{i,j,k+1}$, with the specific modification value being:

$$\Delta effect_{i,j,k+1} = \Delta r_{i,j,k} \times footprint_{i,j}$$

2.2 Void Voxel Jumping Method

Void voxels refer to voxel data that appear transparent after Splatting algorithm reconstruction and contribute nothing to final image formation, such as dust and other elements that do not contribute to real image synthesis. Skipping these void voxels during the rendering process can improve algorithm speed without sacrificing image quality.

In an image containing many coexisting elements, each voxel is typically composed of multiple elements. By analyzing the proportion of different elements in a voxel, we can determine which voxels are void voxels that do not require rendering. In a rendered image, the probability of gray value Y for any element can be expressed as:

$$P(Y) = \sum_{i=1}^{n} p_i \cdot p(Y|i)$$

where n is the number of element types in the voxel, p_i is the proportion of the i-th element type in the voxel, and $p(Y|i)$ is the conditional probability of gray value Y for the i-th element type.

With knowledge of each element' s gray value probability density distribution function, the probability of the i-th element in a voxel with gray value Y can be estimated using Bayes' formula. Based on the gray value obtained from 3D texture after volume data calculation using the footprint table method, sampling points are classified using probability, with the void voxel classification threshold

being *key*. If the gray value probability density distribution function for each element is unknown, the conditional probability density distribution function can be derived by analyzing the original curve of the element's gray distribution, and then the gray value is compared with the void voxel classification threshold. If the gray value is greater than the void voxel classification threshold, lighting and color compositing operations are performed; otherwise, the point is skipped and the next sampling point is processed.

By classifying elements, those requiring rendering are screened. The void voxel jumping formula derived from Equation (11) is:

$$Color = \begin{cases} \sum_{i=0}^{n} value_i & \text{if } key \geq C_p \\ 0 & \text{if } key < C_p \end{cases}$$

The void voxel jumping method accelerates algorithm operation time by skipping gray value calculations and image color compositing for void voxels, thereby reducing access to code texture memory. Images rendered using the void voxel jumping method show no quality loss compared to the original image.

2.3 Splatting Algorithm Based on Adjacent Layer Similarity and Void Voxel Jumping

The adjacent layer similarity acceleration algorithm utilizes the characteristic of small differences between adjacent layers of volume data to accelerate the process of calculating sampling point contributions, while the void voxel jumping method determines whether to skip subsequent calculations for a point by comparing its gray value with the void voxel threshold, thereby improving graphics rendering time. Combining both methods can further accelerate algorithm rendering time without compromising image quality. The flowchart of the new acceleration algorithm is shown in Figure 1 [Figure 1: see original paper].

a) **Footprint Table Construction**: Set the initial reconstruction kernel size and establish a universal footprint table through the reconstruction kernel. When calculating the spatial convolution domain of the reconstruction kernel in the data field, use the given projection direction of the 3D data field, then project voxels and rotation kernel scaling transformation onto a 2D plane, and further use the footprint table to calculate the values of plane projection and covered pixel points in the footprint function to achieve 3D convolution.

b) **3D Texture Data Filtering**: During data reading from the volume data file, filter the 3D texture data. When the 3D texture data $(x, y, z) = 0$, treat it as invalid data and eliminate it.

c) **Gray Value Calculation**: Project the filtered 3D texture data onto a 2D plane using the footprint table, and calculate the gray value of each point using the characteristic of small differences between adjacent layers.

If the difference with the point at the same planar position in the next layer is 0, the gray value remains unchanged; if the difference is not 0, calculate the gray value of that point based on the change amount.

d) **Void Voxel Skipping**: Classify data through gray values, skip the rendering process of void voxels that contribute nothing to the image, and render data useful to the image, thereby quickly drawing the image.

3.1 Experimental Environment and Data Construction

The Splatting algorithm based on adjacent layer similarity and void voxel jumping is evaluated using accelerated algorithm reference time as the evaluation metric. The experimental computer configuration is Intel® CoreTM i5-4210H CPU @ 2.90GHz, NVIDIA GeForce GTX 950M, Windows 10 64-bit, 12GB memory, completed in a C++ environment based on OpenGL. The experiment uses raw data from CT scan image sensors that capture light source signals and convert them into digital signals. A set of images including foot, head (frontal), and head (lateral) views serves as the experimental object (as shown in Figure 2) [Figure 2: see original paper].

3.2 Evaluation Results and Analysis

1) Based on Adjacent Layer Similarity

To verify the effectiveness of the adjacent layer similarity acceleration algorithm, experiments were designed to test CT image rendering speed. The Splatting voxel algorithm and the adjacent layer similarity-based algorithm were used to render data separately, and the experimental results were compared and analyzed. The time comparison for rendering three CT scan entities is shown in Table 1 .

The test results show that screening adjacent voxels can effectively improve algorithm speed. According to the experimental data, foot image rendering time is accelerated to 66.2% of the original, head image to 63%, and circular object to 78%, with an average of 69.1% of the original Splatting algorithm time. The lower the proportion of objects in the image, the less time is consumed. Since only invalid data is removed, image quality is not affected.

2) Based on Void Voxel Jumping Algorithm

Images rendered using the void voxel jumping method show no quality loss compared to the original image, effectively guaranteeing rendering quality. Image rendering experiments were conducted using the improved void voxel jumping algorithm, with acceleration effects compared to the original Splatting method as shown in Table 2 .

The experimental results demonstrate that the void voxel jumping method can significantly accelerate the algorithm, with an average time consumption

of 50.9% of the original Splatting algorithm. As the void voxel classification threshold calculated by Equation (10) becomes more precise and the number of elements requiring rendering decreases, the algorithm becomes faster. In terms of imaging quality, since only voxels that contribute nothing to imaging are skipped, image quality remains unchanged.

3) Acceleration Algorithm Based on Adjacent Layer Similarity and Void Voxel Jumping

The Splatting acceleration algorithms based on adjacent layer similarity and void voxel jumping each have their own advantages and disadvantages. After combining these two volume data processing methods, the CT image effects of various entities are shown in Table 3 . The differences in optimization effects among various algorithms are significant, with foot image rendering accelerated to 38.3% of original time, head (lateral) to 44%, and head (frontal) circular object to 58.1%, averaging 46.8% of the original Splatting algorithm time. The combined Splatting algorithm based on adjacent layer similarity and void voxel jumping shows the most significant rendering effect (Figure 3) [Figure 3: see original paper]. The algorithm is efficient and feasible, and since the two acceleration algorithms operate at different stages, they complement each other in the optimized algorithm to achieve better results.

4 Conclusion

This research focuses on optimizing Splatting acceleration algorithms. Based on the original algorithm' s advantages, a combined acceleration algorithm integrating adjacent layer similarity and void voxel jumping at different stages is implemented and experimentally studied. The results show that the optimized algorithm achieves significant acceleration in image rendering while guaranteeing image quality.

Although the optimized algorithm solves and improves the spatial correlation and operational efficiency issues of the Splatting algorithm to a certain extent, there remains a gap in image rendering quality compared to ray casting algorithms, and it cannot significantly improve the inherent color diffusion problem of the Splatting algorithm. Future research can incorporate image quality improvement ideas into the phased acceleration algorithm to achieve better results.

References

[1] M. Levoy. Display of surfaces from volume data. IEEE Computer. Graph Application, 8 (3): 29-37, 1988.

[2] Zeng Yanyang, Pei Qingqing, Li Baokun. Ray-casting algorithm based on adaptive compound interpolation [J]. Journal of System Simulation, 2018, 30 (11): 4187-4194.

[3] Marathe C V, Gadre V M. Wavelet regularization for frequency domain volume rendering [C]. IEEE Communications, 2013.

[4] Piccand S, Noumeir R, Paquette E. Region of interest and multiresolution for volume rendering [J]. Information Technology in Biomedicine, 2008, 12 (5): 561-568.

[5] Zhu Shi, Chang Jinyi, Improved algorithm of volume rendering combined texture mapping with ray casting based on CUDA [J]. Application Research of Computer, 2015, 32 (06): 1884-1887.

[6] He Yongjun, Zeng Wenquan, Yu Aimin. Certain body of 3D medical image based GPU rendering technology summary [J]. Computer&Digital Engineering, 2016, 44 (01): 141-147.

[7] Li Lin, Lu Cai, Tang Zhiliang, et al. Codebook initialization algorithm based on data stream clustering using GPU [J]. Application Research of Computer, 2017, 34 (02): 426-430.

[8] Hassan A H, Fluke C J, Barnes D G. 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.

[9] Beyer J, Hadwiger M, Pfister H. State-of-the-Art in GPU-Based Large-Scale Volume Visualization [M]. The Eurographs Association & John Wiley & Sons, Ltd. 2015.

[10] Cao Yi, Ai Zhiwei, Wang Huawei. A distributed multi-node GPU accelerated parallel rendering scheme for visualization cluster environment [C]. International Conference on Virtual Reality and Visualization. IEEE Computer Society, 2013: 153-160.

[11] Zhou Fangfang, Gao Fei, Liu Yonggang, et al. Interactive Volume Data Classification Based on Density-Distance Graph [J]. Journal of Software, 2016, 27 (05): 1061-1073.

[12] Zhang Jianxun, Sun Jizhou, Han Fengqing, et al. The Accelerating Splatting Algorithm Based on Comparability of Adjacency-Layers [J]. Journal of System Simulation, 2005 (02): 421-423, 428.

[13] Wang Rui, Chen Chunxiao, Liu Gao, et al. Study on accelerated volume rendering method based on adaptive bounding box division [J]. Chinese Journal of Scientific Instrument, 2014, 35 (11): 2560-2566.

[14] Li Guohe, Duan Zhongxiang, Wu Weijiang, et al. GPU-based volume rendering for full-empty subdata blocks [J]. Journal of Image and Graphics. 2014, 19 (04): 577-582.

[15] Zellmann Stefan, Schulze Jurgen P, Lang Ulrich. Binned k-d tree construction for sparse volume data on multi-core and GPU systems [J]. IEEE transactions on visualization and computer graphics. 2019 (03).

[16] Li Zeyu, Chen Yimin, Zhao Yan, et al. Three dimensional reconstruction of medical images based on improved ray projection algorithm [J]. Application Research of Computers. 2017, 34 (12): 3866-3868, 3888.

[17] He Nannan. 3D reconstruction algorithm for medical images [D]. Henan: Henan University of Technology. 2018.

[18] Tian Liang. Research on visualization of footprint method based on CUDA [D]. Nanjing: Nanjing University of Science and Technology, 2011.

[19] Lin Jinhua. Research on 3D reconstruction algorithm based on spacial voxel fusion [D]. Changchun: Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, 2017.

Note: Figure translations are in progress. See original paper for figures.