Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8822624, 12 pages
https://doi.org/10.1155/2020/8822624

Research Article

A Novel Ray-Casting Algorithm Using Dynamic Adaptive Sampling

Huadeng Wang[1,2] **Guang Xu**,[1,2] **Xipeng Pan**[1,2] **Zhenbing Liu**,[1,2] **Rushi Lan**,[1,2] **and Xiaonan Luo**[3]

[1]*School of Computer and Information Security, Guilin University of Electronic Technology, Guilin, China*
[2]*Guangxi Key Laboratory of Image and Graphic Intelligent Processing, China*
[3]*National Local Joint Engineering Research Center of Satellite Navigation and Location Service, China*

Correspondence should be addressed to Xipeng Pan; pxp201@guet.edu.cn

Received 23 May 2020; Revised 4 August 2020; Accepted 14 September 2020; Published 14 October 2020

Academic Editor: Yin Zhang

Ray-casting algorithm is an important volume rendering algorithm, which is widely used in medical image processing. Aiming to address the shortcomings of the current ray-casting algorithms in 3D reconstruction of medical images, such as slow rendering speed and low sampling efficiency, an improved algorithm based on dynamic adaptive sampling is proposed. By using the central difference gradient method, the corresponding sampling interval is obtained dynamically according to the different sampling points. Meanwhile, a new rendering operator is proposed based on the color value and opacity changes before and after the ray enters the volume element, and the resistance luminosity. Compared with the state of other algorithms, experimental results show that the method proposed in this paper has a faster rendering speed while ensuring the quality of the generated image.

1. Introduction

The ray-casting algorithm was proposed by Levoy in 1988. As an important volume rendering algorithm, the ray-casting algorithm is widely used in medical images, fluid and quantum mechanics, geographical exploration, and finite element analysis [1, 2]. In the basic principle of the ray-casting algorithm, each pixel in the imaging plane sends out a ray along the line of sight through the volume data, resampling along the ray at the same distance, obtaining the color value and opacity of each resampling point, and then synthesizing the color and opacity of each resampling point on the ray from the front to the back or from the back to the front.

Although the traditional ray projection algorithm has a high quality of synthetic image, it has two disadvantages. One is that there are a lot of volume data and a lot of addition and multiplication operations in the interpolation process, which makes the whole rendering process very slow. Second, after the sampling point is determined, the use of cubic linear interpolation will affect the real-time rendering process.

Some researchers use the closest interpolation instead of trilinear interpolation to remove some unnecessary sampling points. Some researchers achieve this goal by reducing the intersection of the light field and the data field. For example, Siddon [3] replaced the intersection of ray and voxel with the intersection of ray and plane. Ross et al. [4] employed the Bresenham algorithm to determine the intersection voxels. The data domain octree organization method proposed by Levoy [5] could easily skip empty voxels and reduce the time complexity of the intersection to O(nlog n). Jian [6] and others chose the projection light according to the correlation of the critical voxels, but the sharpness of the image would be worse.

There are two ways to improve the ray projection algorithm. One is to simplify the operation of the sampling process, such as using trilinear interpolation instead of complex cubic convolution interpolation [7–10]. The other is to

reduce the drawing time by reducing the number of sampling points on each ray. For the existing empty sampling points which are not helpful for image rendering, most of the methods are to eliminate them directly without considering the correlation between them and the surrounding voxels. When users interact with each other, the image will be blurred and the user experience will be affected.

Considering that most of the adaptive sampling methods used in most papers are equal-interval sampling, the threshold value of the isosurface is set first, and the isodistance sampling is started from the viewpoint direction along the line of sight direction until the sampling value exceeds the set threshold value. Finally, the linear interpolation is done between the sampling points $P(n)$ and $P(n+1)$ to get the intersection coordinates of the line of sight and the isosurface. The disadvantage of this method is that there are too many sampling points. Therefore, a large number of sampling points are empty voxels, which wastes precious time and is not conducive to user interaction. Another method is based on a given sampling frequency, but the quality of voxel rendering is very poor. Some scholars try to improve the sampling frequency, adjust the sampling frequency by the distance proportion between the tangent point of the surface normal and the viewpoint, and simplify the image synthesis operator. Another is to resample at a lower sampling frequency first. If the data values of two adjacent resample points differ greatly, then increase the sampling frequency in such a high-frequency area. In the place where the voxel value changes slowly and quickly, the sampling interval should be small to avoid missing voxel information. If the voxel value changes slowly, increase the sampling interval appropriately, so that a large number of empty voxels can be skipped. However, if the sampling interval is too large, the rendering quality will be low, resulting in a large "void" effect [11, 12], that is, the phenomenon of the artifact.

There are two shortcomings in the current ray casting algorithm for medical image rendering, one is the slow rendering speed, the other is the low efficiency and high time complexity of the whole sampling process. In order to solve the above shortcomings, this paper proposes an algorithm. The algorithm uses the central differential gradient method. By this method, the specific sampling distance can be determined dynamically according to the different distances of sampling points, which can improve the sampling efficiency. On the other hand, according to the change of color value and opacity before and after the ray enters the voxel, we propose a new rendering operator to improve the final image rendering speed.

This paper solves the problem of how to find the most suitable sampling interval dynamically. Through the full increment formula of the voxel value difference between two adjacent sampling points, combined with the central difference method, the calculated gradient, and the preset voxel value threshold, the sampling point interval can be calculated dynamically according to the different positions of sampling points. Thus, we can update the sampling interval dynamically on the premise of ensuring the sampling density. Aiming at the problem that the search precision of the intersection point between the light and the actual isosurface will be reduced when the sampling point interval is large, this paper proposes a recursive estimation method based on the intersection point to estimate the intersection point between the sampling point and the isosurface, which is simple and easy to operate. The algorithm we proposed is a novel ray-casting algorithm using dynamic adaptive sampling, which is called the ray-casting DAS algorithm for short. After determining the sampling point, aiming at the general paper's shortcomings of using the nearest interpolation or trilinear interpolation to synthesize the image according to the opacity and color value of eight data points equidistant from the sampling point, this paper adopts a method of light blocking, which is to change the opacity and color values as well as the photometry before and after the ray enters the volume element and improve the composition operator. The basic algorithm structure and process are listed in Figure 1. Experimental results show that the improved method can effectively improve the speed of image rendering and avoid the generation of artifacts. The main innovation of this paper is dynamic adaptive sampling and improving the rendering operator to improve the efficiency and rendering time of the algorithm. Another unique innovation of this paper is the introduction of the distance function to improve the image noise and rendering quality.

The remainder of this paper is as follows. The second part summarizes the latest research results of some ray projection algorithms and compares their advantages and disadvantages. In the third part, the ray projection algorithm based on adaptive sampling is introduced. The fourth part is the comparison between the ray-casting DAS and other ray projection algorithms in several open-source datasets. The fifth part is the summary of ray-casting DAS and the assumption of improving in the future.

2. Related Work

Wu et al. [13] proposed a GPU ray projection method for visualizing 3D pipelines in a virtual sphere. In their method, the pipeline data was initially divided into several data blocks. The pipeline centerline in each block was then segmented and coded, and a thicker pipeline envelope was then constructed using geometric shaders. Finally, elaborate 3D pipes could be rendered using pixel shaders. Compared with the traditional polygon-based pipeline data partition method, this method improves the rendering frame rate under the same pixel-level precision display effect. The visualization of 3D pipe thickness is realized without affecting the rendering efficiency.

Luo et al. [14] proposed a rendering algorithm based on the height field [15] and an improved algorithm of finding the intersection of isosurfaces to solve the problem of the slow speed of finding the intersection of light and volume data. However, because the volume data will be imaged many times due to the reflection of the surface of the object, so it needs to do many volume data rendering operations, which hurt the performance of the algorithm.

Francisco Sans et al. [16] proposed to combine CPU based ray projection with GPU, and reduce the time complexity of rendering by dynamic gradient interpolation of

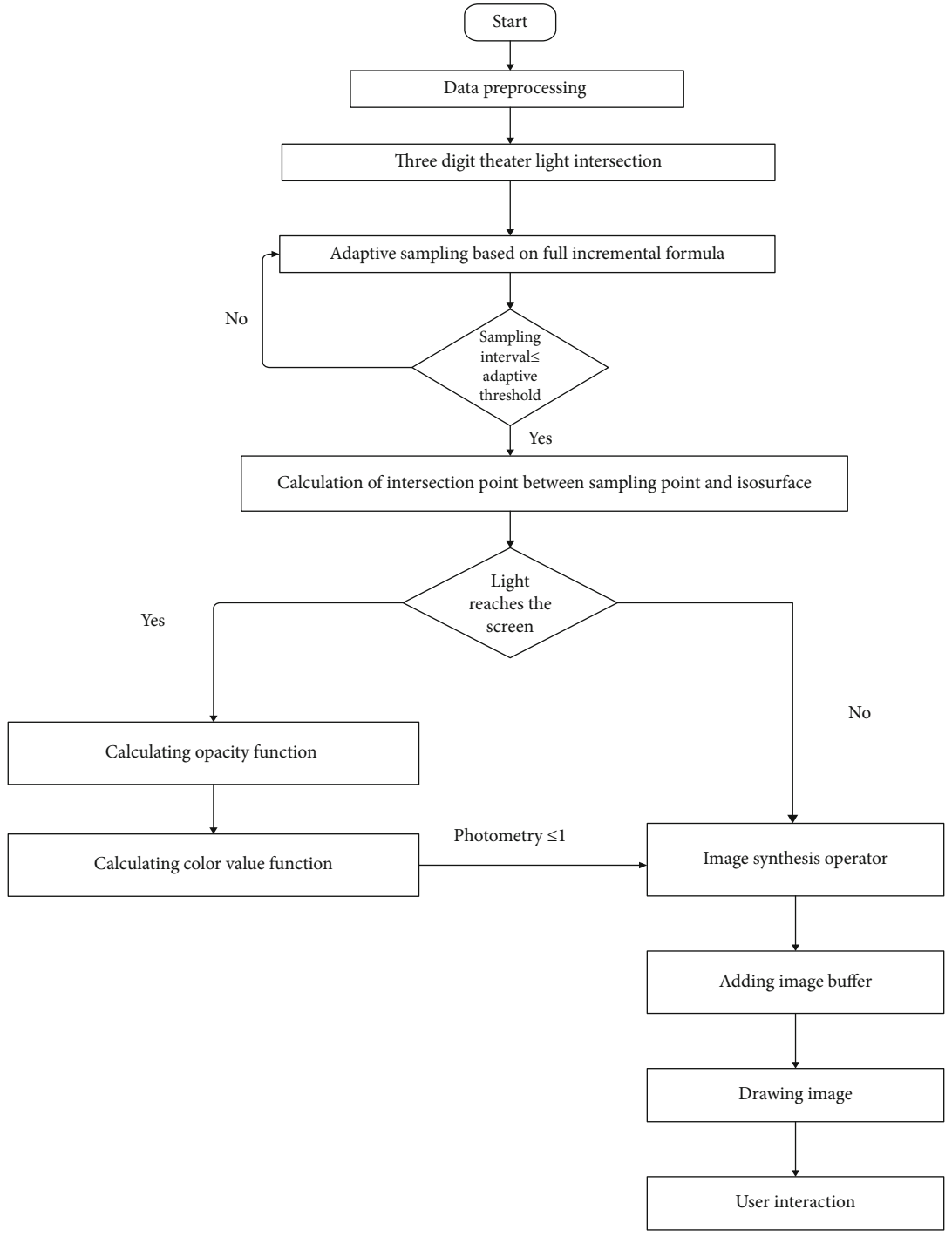


Figure 1: The brief structure of the ray-casting DAS algorithm.

voxels. It realized single channel GPU rendering and meet the real-time interaction requirements of users.

Binyahib et al. [17] proposed a parallel volume rendering algorithm which combines the classic object sequence and image sequence techniques. The algorithm runs on an unstructured grid (and structured grid), so it can deal with block boundary crossing in a complex way. It can also effectively deal with situations that are prone to load imbalance [18]. At the largest scale, it can be expanded to 8192 processors and run on datasets of more than 1 billion cells.

Tao et al. [19] proposed a ray projecting algorithm combining the fitness estimation based on ray-casting with the global optimization method of improved adaptive differential evolution. This method eliminated the fine registration step of the famous iterative nearest point (ICP) algorithm [15, 20, 21], and it is the first direct global registration algorithm.

Moreover, the calculation of the fitness of the global optimization method was accelerated, and the search space was effectively used to find the optimal transformation solution. The algorithm was successfully implemented in parallel mode on multicore computer processors.

3. Our Proposed Approach

3.1. Data Preprocessing Based on Fuzzy Enhancement. Because in the ray projection algorithm, in the mapping from three-dimensional to two-dimensional, the image edge, region, texture, and so on are inevitably blurred due to the loss of information. Therefore, the algorithm proposed in this paper first preprocesses the data, uses the method of enhancement operator [22] to calculate the blur rate of the enhanced image, and then selects the best threshold.

The purpose of introducing a fuzzy enhancement operator [23] is to enhance the contrast of the image region by different enhancement processing, that is, the black target area is attenuated and the white background area is enhanced. The enhanced image not only retains the information of the original image but also makes the layers of each region clearer, which further reduces the image ambiguity. In this way, the change of the processed data is beneficial to the extraction and selection of sampling points for the subsequent volume rendering algorithm.

3.2. Intersection Processes of Light and Plane. First, the parametric equation of light emitted by a pixel in the plane can be expressed as:

$$\begin{cases} x = x_0 + lt, \\ y = y_0 + mt, \\ z = z_0 + nt, \end{cases} \tag{1}$$

where (l, m, n) represents the direction vector of the line in the object space, (x_0, y_0, z_0)represents a point on the line. x, y, and z denote the three-dimensional coordinates of any point on the line, and t is the parameter of the linear parametric equation, indicating the number of directions of the line.

Let the size of the regular data field be $L \times M \times N$ and logical coordinates $(x, y, z) = (i \times X, j \times Y, k \times Z)$, in which, $i = 0, 1 \cdots, L-1$, $j = 0, 1, \cdots, M-1$, and $k = 0, 1, \cdots, N-1$. Here L, M, and N denote the number of X, Y, and Z plane families, respectively, and (x, y, z) represents the projection coordinates of a specific point in the X, Y, Z plane of the specific plane family. The data distribution of the volume data field is represented by the plane family. The definition of the plane family is as follows:

$$\begin{cases} x = i \times X, i = 0, 1, \cdots, L-1, \\ y = j \times Y, j = 0, 1, \cdots, M-1, \\ z = k \times Z, k = 0, 1, \cdots, N-1. \end{cases} \tag{2}$$

Because the normal vector of light $(l, m, n) \neq 0$, let us set $n \neq 0$. Then, the parameters of the intersection of light and plane $\{z = k \times Z, k = 0, 1, \cdots, N-1\}$ can be calculated as follows:

$$t_k = \frac{k \cdot Z - z_0}{n}, k = 0, 1, \cdots, N-1. \tag{3}$$

Further, the intersection of light and $z = k \times Z$ can be expressed as:

$$\left(x_0 + \frac{(-z_0 + kZ) \cdot l}{n}, y_0 + \frac{(-z_0 + kZ) \cdot m}{n}, kZ\right), k = 0, 1, \cdots, N-1. \tag{4}$$

Eq. (4) infers that light and two adjacent planes $z = k \times Z$ and $z = (k+1) \times Z$ intersection parameters t_k and t_{k+1} satisfying Eq. (5)

$$t_{k+1} = t_k + \frac{Z}{n}. \tag{5}$$

The intersection points of the straight line and z plane family can be obtained by using the above iteration formula:

$$\begin{cases} x_{k+1} = x_k + \dfrac{Z}{n}, \\ y_{k+1} = y_k + \dfrac{m}{n} Z, k = 0, 1, \cdots, N-2, \\ z_{k+1} = z_k + \dfrac{Z}{n}. \end{cases} \tag{6}$$

Similarly, the recurrence formula to the intersection of the line and the x, y plane can be obtained.

3.3. Dynamic Adaptive Sampling. The intersection points of the line and plane family obtained by the above operations are not the sampling points we need, but we can get the sampling points on the line based on these intersections. The general method is to obtain the sampling points by equidistant sampling with a fixed distance d. Here, we consider how to dynamically adjust the adaptive sampling interval.

Let g be a three-dimensional volume data field. According to the total differential formula of the binary function, the difference between the voxel values of two adjacent sampling points can be obtained as follows:

$$\begin{aligned} \Delta g &= g(n+1) - g(n) \\ &= g_x' \cdot \Delta x + g_y' \cdot \Delta y + g_z' \cdot \Delta z + o(\rho) \approx \nabla g \cdot L \cdot d_n, \end{aligned} \tag{7}$$

where ∇g is the gradient of the 3D digital theater and d_n is the coordinate distance of the sampling point. L is the unit vector of the light direction.

Because $g(n)$ has been estimated, and $g(n+1)$ is unknown. It may be assumed that the voxel value $g(n+1)$ of the sampling point $P(n+1)$ is T (T is the set voxel value threshold). According to Eq. (7), the sampling interval can be approximately expressed as:

$$d_n \approx \frac{T - g_n}{\nabla g \cdot L}. \tag{8}$$

Input: $P(n)$, $P(n+1)$ number_Current, Total.
Output: P0
1: Take the coordinate P0 of the geometric midpoint of $P(n)$ and $P(n+1)$ to obtain the sampling value g.
2: T is set threshold, and the number current represent the current iterations, which is required to less than the Total.
3: If $g > T$ then
4: $P(n+1)$ = P0
5: Else
6: $P(n)$ = P0
7: While $|P(n) - P(n+1)| < \varepsilon$ do
8: By linear interpolation of $P(n)$ and $P(n+1)$, the coordinates of the intersection point of $P(n)$ and $P(n+1)$
9: If number_Current>Total then return P0
10: If P0 is close to $P(n)$ then
11: $P(n)$ = P0
12: Else
13: $P(n+1)$ = P0
14: Return P0

Algorithm 1. solution process of intersection coordinates

It can be seen from Eq. (8) that if the projection value of the gradient at a point in the direction of light is smaller, and the difference between $g(n)$ and T is large, then the interval d_n will be larger; otherwise, d_n will be too small. In order to prevent the sampling interval from being too large or too small, an upper and lower bound can be added to the sampling interval of Eq. (8). It is as follows:

$$d_n = \begin{cases} D_1 & 0 \le \dfrac{T - g_n}{\nabla g \cdot L} \le D_1, \\ D_2 & \dfrac{T - g_n}{\nabla g \cdot L} \ge D_2, \text{ or } \dfrac{T - g_n}{\nabla g \cdot L} < 0, \\ \dfrac{T - g_n}{\nabla g \cdot L} & D_1 < \dfrac{T - g_n}{\nabla g \cdot L} < D_2. \end{cases} \tag{9}$$

D_1 and D_2 are upper and lower bounds of sampling, respectively.

In order to simplify the calculation, the center difference method is used to calculate the gradient, and the calculation method is as follows:

$$\nabla g = \frac{1}{2} \begin{cases} g(x+1, y, z) - g(x-1, y, z), \\ g(x, y+1, z) - g(x, y-1, z), \\ g(x, y, z+1) - g(x, y, z-1). \end{cases} \tag{10}$$

Because the gradient calculation will affect the rendering speed if it is placed in the sampling stage, so we put the gradient calculation in the data preprocessing stage and use a linear table to store the calculated gradient value. When calculating the sampling step length, we look up the gradient value closest to the sampling point in the linear table as the gradient of the sampling point. Assuming the number of sampling points is n, then the time complexity of this step is $O(n)$.

After determining the sampling interval, we need to find the intersection of two adjacent sampling points and the isosurface and then synthesize the image through the color value and opacity of the intersection. In order to improve the imaging quality, this paper presents a recursive method for intersection estimation.

Supposing that we need to find the intersection point of the two sampling points $P(n)$ and $P(n+1)$ and the isosurface. First, we take the midpoint coordinate P0 of $P(n)$ and $P(n+1)$. If the corresponding voxel value at P0 is greater than the set threshold T, then P0 is taken as the new $P(n+1)$. If the corresponding voxel value at P0 is less than the set threshold T, then P0 is taken as the new $P(n)$, and then continue to take the midpoint of $P(n)$ and $P(n+1)$ until the distance between $P(n)$ and $P(n+1)$ is very small (less than the given value). Then, the final $P(n)$ and $P(n+1)$ are linearly interpolated to obtain the coordinates of P0, which is the coordinates of the intersection.

The process of finding the intersection coordinates is shown in Algorithm 1. Through the algorithm, we can calculate the intersection point of sampling points $P(n)$ and $P(n+1)$ fast. Moreover, we set the threshold of the number of iterations, which improves the convergence speed of the algorithm and indirectly improves the rendering speed of the image.

3.4. Improving Rendering Operator. After selecting the sampling point and intersection point in the above steps, the traditional method is to do linear interpolation around the eight closest data points of the intersection point to synthesize the image [24]. For example, the ray projection algorithm based on trilinear interpolation for fiber surface is adopted by Francisco Sans, and the time complexity of the trilinear interpolation is $O(n^3)$.

The formula of cubic linear interpolation is as follows:

$$\begin{aligned} S(\mathrm{i}, j, k) &= P_{000}(i-1)(j-1)(k-1) \\ &+ P_{100} i(j-1)(k-1) + P_{010}(i-1) j(k-1) \\ &+ P_{110} ij(k-1) + P_{001}(i-1)(j-1)k \\ &+ P_{101} i(j-1)k + P_{011}(i-1)jk + P_{111} ijk. \end{aligned} \tag{11}$$

Among them, $S(i, j, k)$ represents the voxel value of the corresponding spatial coordinate point $P(i, j, k)$, P000, P001, ⋯, P111 represents the voxel value of the eight adjacent data points of $P(i, j, k)$, respectively.

In this paper, we consider a method of ray blocking and synthesize a new rendering operator through the change of color and opacity before and after the ray enters and leaves the voxel, as well as the corresponding opacity. The experimental results show that it can effectively reduce the time complexity and improve the rendering speed without affecting the imaging quality.

Let the opacity and color values of the ith volume element be P_i and Q_i, respectively, the opacity and color values before the ray enters the ith volume element $P^{'}_i$ and $Q^{'}_i$, respectively, and the opacity and color values after the ray enters are $P^{''}_i$ and $Q^{''}_I$, respectively, and let the opacity of the cell be G_i; the composite operator of the point image is

$$P''_i \times Q''_i = P_i \times Q_i \times (1 - G_i) + P'_i \times Q'_i. \tag{12}$$

The relationship between the opacity and opacity of current voxels is

$$P''_i = P_i \times (1 - G_i) + P'_i. \tag{13}$$

At the same time, we notice that not all the light can finally reach the imaging screen for imaging. The initial point has the smallest opacity. With the increase of data points through which the ray passes, the opacity also increases. Finally, after the ray enters the nth data point, the opacity becomes the maximum 1, so the image synthesis operator can be simplified as:

$$Q''_i = \sum_{i=1}^{n} P_i \times Q_i \times (1 - G_i). \tag{14}$$

From the above formula, it can be found that when $g = 1$, that is the maximum photometry, the right side of the equation is equal to 0, which means that the ray that does not reach the screen does not participate in the calculation of the synthesis operator. At the same time, the time complexity of single P_i and Q_i calculation is $O(n)$, so the time complexity of the final synthesis operator is $O(n^2) < \mathrm{O}(n^3)$, so the algorithm effectively improves the rendering efficiency. On the other hand, it can not only improve the time of image rendering but also effectively prevent the generation of artifacts and improve the quality of rendering.

4. Experimental Results

In this section, several experiments and comparison results are reported to evaluate the performance of the proposed algorithm ray-casting DAS. We compare the ray-casting DAS with other algorithms in several aspects, such as time complexity, image rendering speed, and image quality.

4.1. Time Consumption Comparison. The following is a comparison of several algorithms' test data on the open dataset ISIC2018 [25]. Table 1 compares the drawing time of different algorithms for drawing the same picture of the human leg. VS3D, Dual-FHR, RIFT2, SHSR-UV, and ray-casting DAS in Figure 2 represent the experimental results of Wu's algorithm [13], Luo's algorithm [14], Francisco Sans' algorithm [16], Roba Binyahib's algorithm [17], and ray-casting DAS algorithm, respectively.

Table 1: Comparison of the time consumption of each stage of the above algorithms (unit: seconds).

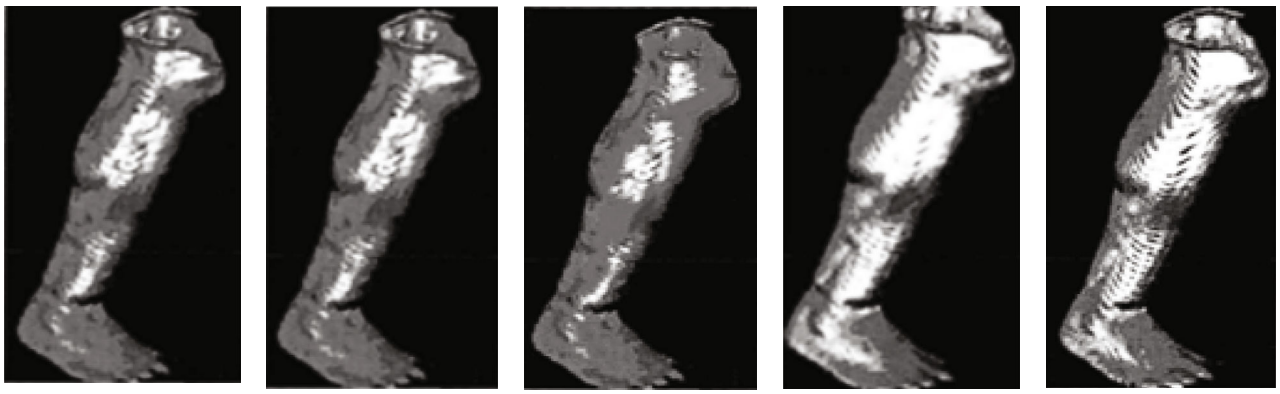
Algorithms	Data preprocessing time	Sampling time	Drawing time	Total time
VS3D	23	32	33	88
Dual-FHR	17	28	31	76
RIFT2	16	25	31	72
SHSR-UV	17	23	26	66
Ray-casting DAS	24	16	21	61

According to the experimental results in Table 1, compared with other algorithms, the preprocessing time of the algorithm in this paper is increased by about 6 to 7 seconds compared with other algorithms. This is because we add gradient calculations in the preprocessing stage, which consumes more time. However, due to the use of dynamic adaptive sampling, the acquisition of sampling points is more efficient, and the interference of unnecessary sampling points is eliminated. Compared with other algorithms, the time consumption of this algorithm in the sampling phase is greatly reduced, reducing 7-16 seconds. On the other hand, due to the improvement of the rendering operator, the idea of early termination is added in the rendering process, which greatly reduces the rendering time by 5-10 seconds. Finally, compared with other better algorithms, the total time is improved by 5-20 seconds, and the speed is significantly improved.

Through the results of Figure 2, we can find that the algorithm in this paper can better draw the details of the leg, including some texture information. These details are very valuable for medical diagnosis. Compared with other algorithms, the overall occlusion of the scene is captured coherently and close to the real value. The shadow area of the leg is less, and more texture details are preserved with ray-casting DAS.

4.2. Rendering Speed and Quality Comparison on CT Dataset. As shown in Figure 3, RIFT2, SHSR-UV, ICVC-GPU, DOS-Ray, IMPA, and ray-casting DAS represent the experimental results of Francisco Sans' algorithm [16], Roba Binyahibs' algorithm [17], Feiniu Yuan's algorithm [26], Leonardo's algorithm [27], Yan Zhang's algorithm [28], and ray-casting DAS algorithm, respectively. By comparing the results of the different algorithms on the dataset of ISBI2018 [29], the ray-casting DAS algorithm can reduce and avoid the phenomenon of artifact in CT image rendering and improve the visibility of the image.

Table 2 compares the rendering time of the ray-casting DAS algorithm with the other five algorithms. The



(a) VS3D (b) Dual-FHR (c) RIFT2 (d) SHSR-UV (e) Ray-casting DAS

Figure 2: The experimental results of the above algorithms on dataset ISIC2018. The experimental results show that compared with other algorithms, this algorithm can significantly reduce the sampling time required for image rendering, improve the rendering speed, and also improve the overall time consumption.

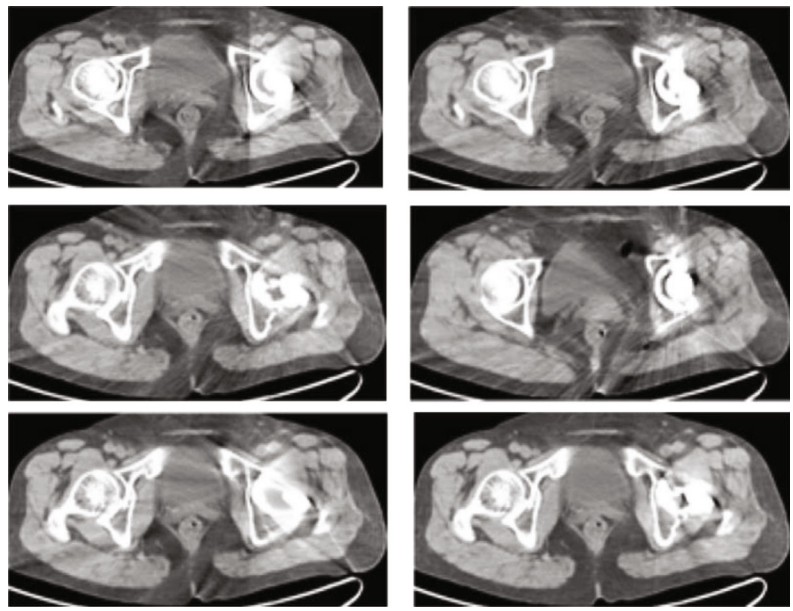


Figure 3: Comparison of the phenomenon of artifact and respective visibility of several algorithms.

Table 2: Comparison of the rendering speed of various algorithms (drawing times/s, signal-to-noise ratio/dB).

Algorithms	Iteration	Drawing time	Signal-to-ratio
RIFT2 algorithm [16]	46	24	38
SHSR-UV algorithm [17]	53	29	42
ICVC-GPU algorithm [26]	34	21	33
DOS-ray algorithm [27]	49	25	34
IMPA algorithm [28]	45	22	36
Ray-casting DAS algorithm	32	15	60

experimental results show that ray-casting DAS has better performance in rendering time and better image signal-to-noise ratio than the previous algorithms.

Similarly, from the experimental results in Figure 3, it can be seen that the rendering algorithm in this paper can effectively solve the problem of artifacts in CT image rendering. From (a) to (e), we can clearly see the existence of artifacts in CT images, they obviously cause interference to the analysis image, and the image (f) drawn by ray-casting DAS algorithm can effectively eliminate the existence of these artifacts. It can be clearly seen from the graph that in the experimental results of this paper, each part of the image has no artifacts interference, and the image is clear.

4.3. Influence of Different Distance Functions on Image Rendering. The distance function expresses the relationship between distance and distance intensity value. The distance determines the distance intensity value. Through the distance function, the distance and depth are reflected in the final drawing process. This paper proposes an interactive

Table 3: Comparison of several distance functions.

Algorithms	LCC	SPOCC	SSIM	MSSIM	PSNR
Without distance function	0.69	0.71	0.792	0.813	29 db
Ray-casting DAS with f1 function	0.72	0.76	0.814	0.839	33 db
Ray-casting DAS with f2 function	0.76	0.79	0.857	0.874	36 db
Ray-casting DAS with f3 function	0.75	0.78	0.849	0.867	35 db
Ray-casting DAS with f4 function	0.86	0.85	0.896	0.918	43 db

algorithm. Interaction is based on distance first. The interaction process specifies the distance value, and the distance value is used in the distance function. Then, the final transparent value is determined by combining the distance function in the rendering process. The corresponding distance function is a piecewise function:

$$f(x) = \begin{cases} \text{high}, x \in [0, k), \\ \text{low}, x \in [k, \max). \end{cases} \tag{15}$$

The value k is the distance value obtained by interaction. When the distance is less than k, it is a high-intensity value, and when the depth is greater than k, it is a low-intensity value.

In order to improve the quality of rendering and reduce the interference of noise and other factors, we introduce the concept of distance function before color and opacity composition. The four contrast distance functions used in this experiment are as follows:

$$f_1(x) = \max_i \{\|x_i\|, 1 \le i \le p, \|x_i\| \le 52\}, \min f_1 = 0, \tag{16}$$

$$f_2(x) = \sum_{i=1}^{p} i\|x_i\|^4 + \text{random}[0, 1), 1 \le i \le p, \|x_i\| \le 52, \min f_2 = 0, \tag{17}$$

$$f_3(x) = \sum_{i=1}^{p} \|x_i\|^2 - 10\cos(2\pi\|x_i\|), 1 \le i \le p, \|x_i\| \le 52, \min f_3 = 0, \tag{18}$$

$$f_4(x) = -20\sum_{i=1}^{p} \exp\left(-0.2\sqrt{\frac{1}{30}\|x_i\|^2}\right) - \frac{1}{30}\exp\left(\sum_{i=1}^{p}\cos(2\pi\|x_i\|)\right) + 20 + e, 1 \le i \le p, \|x_i\| \le 52, \min f_4 = 0. \tag{19}$$

Here, x_i is a three-dimensional column vector, representing the spatial coordinates of the ith voxel, and p is the total number of all effective voxels.

As is shown in Table 3, LCC [31] is short for linear correlation coefficient, and SROCC [32] is short for Spearman's rank-order correlation coefficient, which are both evaluation indexes to measure the similarity between drawn image and real image. The structural similarity theory is simplified as SSIM [33], whose value is between 0 and 1. The larger LCC, SROCC, and SSIM value indicate the higher similarity between the drawn image and the real value of the original image. The peak signal-to-noise ratio (PSNR) [32] measures the ratio of useful information and noise in an image. For the same image, the larger the signal-to-noise ratio is, the smaller the noise is, and the higher the image quality is.

In the experiment, four different distance functions are used, and their performances are compared. The experimental results in Table 3 show that, compared with the method without distance function, adding distance function can significantly increase the similarity between the drawn image and the real image; at the same time, it has a higher signal-to-noise ratio and relatively less noise content. On the other hand, Figure 4 shows that different distance functions have advantages and disadvantages in image rendering. The distance function f_4 is the best in the quality of completion. It can eliminate the interference of noise to the greatest extent. The image drawn is also the clearest and contains the least noise interference.

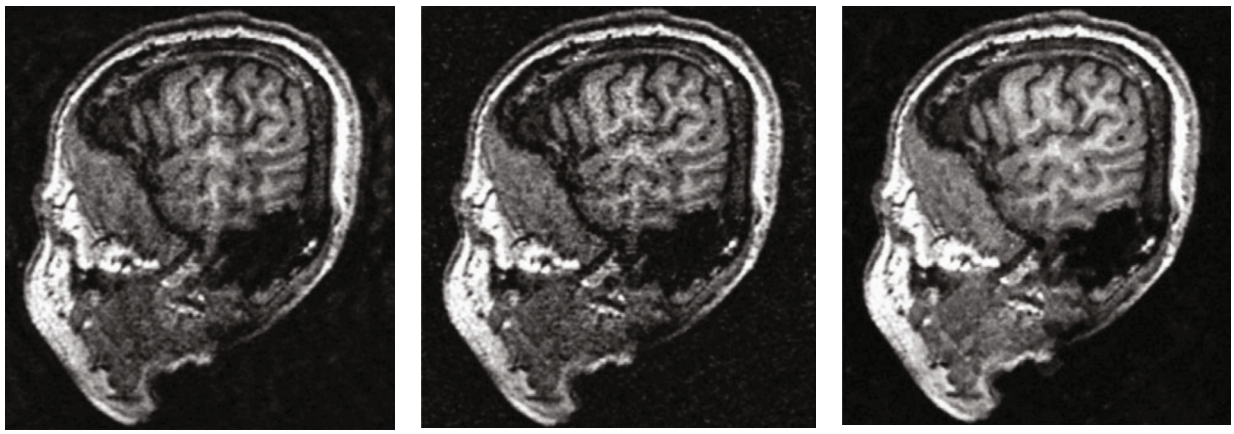
4.4. Image Rendering Quality Comparison on Other Dataset. We use the open-source dataset VisMale [34] for experiments. The simulation environment is VS2016 and OpenGL. Table 4 shows the comparison of the six algorithms for human head image rendering, and ICVC-GPU, MVRC2, SigIg+FMI, SHSR-UV, DBRay, and Ray-casting DAS in Figure 5 represent the experimental results of Feiniu Yuan's algorithm [26], Mohammandmehdi Bozorgi's algorithm [32], R. Mehaboobathunnisa's algorithm [35], Roba Binyahibs' algorithm [17], Alec G.Moore's algorithm [36], and ray-casting DAS algorithm proposed in this paper, respectively.

In Figure 6, information entropy, contrast, sharpness, SNR [31, 33], and PSNR [32] are three objective indicators reflecting image quality. Their definitions are as follows:

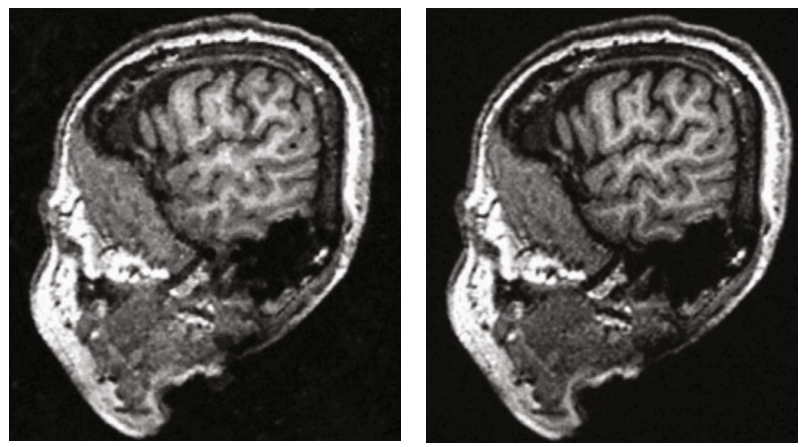
(1) Information entropy represents the amount of information contained in the image, which is proportional to the amount of information contained in the image [33]. The formula is

$$\text{Ent} = -\sum_{i=0}^{L-1}\sum_{}^{L-1} p(i, j)\log p(i, j), \tag{20}$$

where i represents the gray value of the pixel $(0 \le i \le 255)$, j represents the mean gray value of the domain $(0 \le j \le 255)$. And the formula $p(i, j) = f(i, j)/\text{N2}$ reflects the comprehensive characteristics of the gray value of a pixel position and the gray distribution of its surrounding pixels, where $f(i, j)$ is the frequency of the feature binary (i, j), and N is the scale of the image.



(a) Eithout distance function (b) f 1 function (c) f 2 function



(d) f 3 function (e) f 4 function

Figure 4: Comparison of algorithms using different distance functions in the MICCAI [30] dataset.

Table 4: Comparison of completion time of several algorithms (unit: seconds).

Algorithms	Data preprocessing time	Sampling time	Drawing time	Total time consumption
ICVC-GPU	15	25	26	66
MVRC2	14	22	19	56
SigIg+FMI	13	24	32	69
SHSR-UV	15	24	33	72
DARay	17	26	23	66
Ray-casting DAS	22	19	15	57

(2) Contrast reflects the influence of image on visual effect [32]. Contrast is proportional to the clarity of the image. The formula is as follows:

$$\text{con} = L\sqrt{\frac{1}{MN}\sum_{x=0}^{L-1}\sum_{y=0}^{L-1}\left[I(x,y) - \frac{1}{MN}\sum_{x=0}^{L-1}\sum_{y=0}^{L-1}I(x,y)\right]^2}. \quad (21)$$

Among them, L denotes the average gray level of the pixels, M and N denote the width and height of the image, and $I(x, y)$ denotes the gray level of the pixels (x, y).

(3) Sharpness represents the clarity of the image, and the calculation formula is as follows

$$\text{Def} = \sum_{x=0}^{M-2}\sum_{y=0}^{N-2}G(x,y). \quad (22)$$

(4) Firstly, the mean square error (MSE) [37] is defined. The mean square error is used to calculate the mean square value of the pixel difference between the

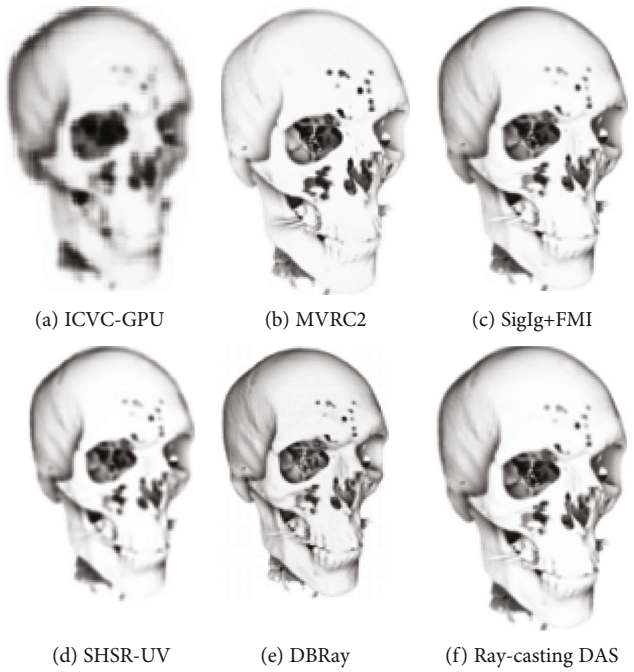


Figure 5: Comparison of the visual effects of the different algorithms in image rendering.

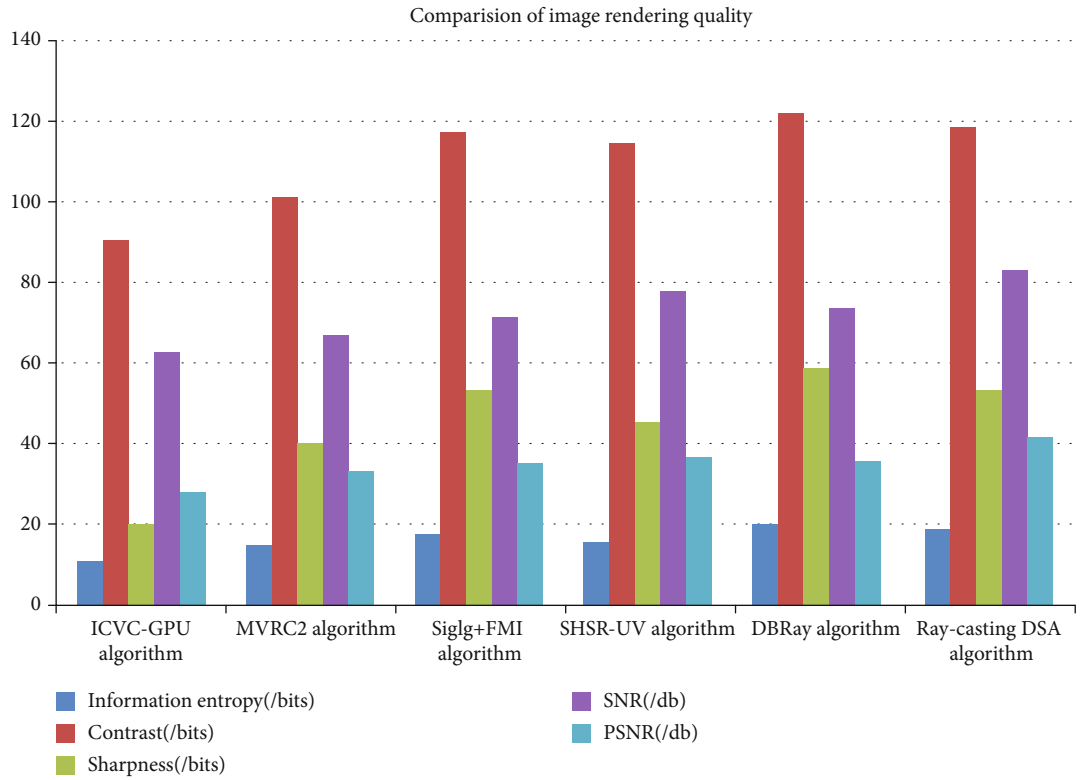


Figure 6: Comparison of image quality generated by several algorithms.

original image and the distorted image. The formula is as follows:

$$\mathrm{MSE} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}(R(m,n) - I(m,n))^2}{M \times N}. \quad (23)$$

Peak signal-to-noise ratio (PSNR) [32] is the ratio of maximum signal quantity to noise intensity. Since digital images are represented by discrete numbers, generally $L = 255$.

$$\mathrm{PSNR} = 10 \ln \frac{l^2}{\mathrm{MSE}} = 10 \ln \frac{255^2}{\mathrm{MSE}}. \quad (24)$$

(5) Signal to noise ratio (SNR)'s definition is as follows [31, 33]

$$\mathrm{SNR} = 10 \lg \left(\frac{\sum_{m=1}^{M} \sum_{n=1}^{N} R(m,n)^2}{\sum_{m=1}^{M} \sum_{n=1}^{N} (R(m,n) - I(m,n))^2} \right). \quad (25)$$

The higher the SNR is, the less noise the image owns.

Through the experimental results in Figure 3, we can find that ray-casting DAS is slightly worse than algorithms E in image rendering quality, because the dynamic adaptive sampling used in this paper will inevitably skip some small voxels containing useful information due to the setting of the threshold value and other problems, leading to slightly poor effective information of the image, but it is close to or much better than A, B, C, and D in image rendering quality. Because the distance function is introduced into the process of rendering operator composition and the most appropriate distance function is selected, the image drawn by this algorithm has a high signal-to-noise ratio and contains less noise and other interference factors, compared with other algorithms. This is a significant progress. Simultaneously, the sampling and drawing time of ray-casting DAS, as shown in Table 3, is much better than all the other five algorithms, which indicates that our method is effective and saving time consumption.

5. Conclusion

Based on the low efficiency of the sampling point selection and slow rendering speed of the existing ray-casting algorithm, a new method based on dynamic sampling and improved rendering operator is proposed in this paper. The experimental results show that this method can effectively shorten the time of sampling and rendering and improve the efficiency of the algorithm on the premise of ensuring the high quality of graphics rendering.

At the same time, this paper introduces the concept of distance function in the process of image rendering, which makes the image contrast high and contains less noise. However, there are other shortcomings in this algorithm, such as the need for gradient calculation, and the gradient calculation is placed in the preprocessing stage, which aggravates the calculation amount in the preprocessing stage. It results in more time in the pretreatment stage. The efficiency of this algorithm can be further improved by a more efficient gradient calculation method.

Data Availability

The data used to support the findings of this study are available from the following websites: https://challenge2018.isic-archive.com/, https://grand-challenge.org/All_Challenges/, http://adni.loni.usc.edu/data-samples/accessdata/, and http://academictorrents.com/details/a9e2741587d42ef6139aa474a95858a17952b3a5.

Disclosure

The earlier version of this paper has been presented as a conference abstract in "ISAIR 2020: THE 5TH INTERNATIONAL SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND ROBOTICS 2020" according to the following link: https://easychair.org/smart-program/ISAIR2020/2020-08-09.html#talk:157507.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank Campagnolo for publishing the dataset and Francisco Sans for making the code available. We are grateful for the comments from the anonymous reviewers. This research was supported in part by National Natural Science Foundation of China (Grant Nos. 61562013, 61772149, 61866009, 62002082), Guangxi Key Research and Development Project (Grant No. AB19110038), Guangxi Natural Science Foundation (Grant Nos. 2019GXNSFAA245014, 2017GXNFDA198025, 2018GXNSFAA294132, 2020GXNSFBA238014), and Guangxi University Young and Middle-aged Teachers' Research Ability Improvement Project (Grant Nos. 2019KY0238, 2020KY05034).

References

[1] A. Beristain, J. Congote, and O. E. Ruiz, "Volume visual attention Maps (VVAM) in ray-casting rendering," in *Medicine Meets Virtual Reality (MMVR)*, pp. 53–57, Newport Beach, California, USA, April 2012.

[2] Z. Lesar, "Real-time ray casting of volumetric data," in *IEEE EUROCON 2015 - International Conference on Computer as a Tool (EUROCON)*, pp. 45–62, Salamanca, Spain, March 2015.

[3] H. Ray, H. Pfister, D. Silver, and T. A. Cook, "Ray casting architectures for volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 210–223, 1999.

[4] J. R. Mitchell, P. Dickof, and A. G. Law, "A comparison of line integral algorithms," *Computers in Physics*, vol. 4, no. 2, pp. 166–172, 1990.

[5] M. Levoy, "Volume rendering: display of surface from volume data," *IEEE Computer Graphics & Applications*, vol. 8, no. 3, pp. 29–37, 1998.

[6] Z. Jian, "The study of volume rendering techniques for medical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 7, pp. 463–473, 2005.

[7] Z. Liu, H. Seo, A. Castiglione, K.-K. R. Choo, and H. Kim, "Memory-efficient implementation of elliptic curve cryptography for the Internet-of-Things," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 521–529, 2019.

[8] K.-K. R. Choo, C. Esposito, and A. Castiglione, "Evidence and forensics in the cloud: challenges and future research directions," *IEEE Cloud Computing*, vol. 4, no. 3, pp. 14–19, 2017.

[9] I. Demir and R. Westermann, "Vector-to-closest-point Octree for surface ray-casting," in *Vision, Modeling and Visualization(VMV)*, pp. 65–72, Aachen, Germany, October 2015.

[10] I. Herrera, C. Buchart, I. Aguinaga, and D. Borro, "Study of a ray casting technique for the visualization of deformable volumes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 11, pp. 1555–1565, 2014.

[11] A. Averbuch, G. Lifschitz, and Y. Shkolnisky, "Accelerating X-ray data collection using pyramid beam ray casting geometries," *IEEE transactions on image processing*, vol. 20, no. 2, pp. 523–533, 2011.

[12] B. Lee, J. Yun, J. Seo, B. Shim, Y.-G. Shin, and B. Kim, "Fast high-quality volume ray-casting with virtual samplings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1525–1532, 2010.

[13] Z. Wu, N. Wang, J. Shao, and G. Deng, "GPU ray casting method for visualizing 3D pipelines in a virtual globe," *International Journal of Digital Earth*, vol. 12, no. 4, pp. 428–441, 2019.

[14] J. Luo, G. Hu, and G. Ni, "Dual-space ray casting for height field rendering," *Journal of Visualization and Computer Animation*, vol. 25, no. 1, pp. 45–56, 2014.

[15] D. Tost, S. Grau, M. Ferre, and A. Puig, "Ray-casting time-varying volume data sets with frame-to-frame coherence," in *Visualization and Data Analysis 2006*, vol. 6060, pp. 505–522, San Jose, CA, USA, January 2006.

[16] F. Sans and R. Carmona, "A comparison between GPU-based volume ray casting implementations: fragment shader, compute shader, OpenCL, and CUDA," *CLEI Electronic Journal*, vol. 20, no. 2, pp. 643–668, 2017.

[17] R. Binyahib, T. Peterka, M. Larsen, K.-L. Ma, and H. Childs, "A scalable hybrid scheme for ray-casting of unstructured volume data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 7, pp. 2349–2361, 2019.

[18] C. Schulz and A. Zell, "Sub-pixel resolution techniques for ray casting in low-resolution occupancy grid maps," in *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–6, Prague, Czech Republic, Czech Republic, August 2019.

[19] L. Tao, T. Bui, and H. Hasegawa, "Global ray-casting range image registration," *IPSJ transactions on computer vision and applications*, vol. 9, no. 1, pp. 122–138, 2017.

[20] S. Lim and B.-S. Shin, "A half-skewed Octree for volume ray casting," *IEICE Transactions on Information and Systems*, vol. E90-D, no. 7, pp. 1085–1091, 2007.

[21] F. Sans and R. Carmona, "Volume ray casting using different GPU based parallel APIs," in *2016 XLII Latin American Computing Conference (CLEI)*, pp. 1–11, Valparaíso, Chile, October 2016.

[22] L. Xiao, C. Li, Z. Wu, and T. Wang, "An enhancement method for X-ray image via fuzzy noise removal and homomorphic filtering," *Neurocomputing*, vol. 195, no. 2, pp. 56–64, 2016.

[23] M. Mouzai, C. Tarabet, and A. Mustapha, "Low-contrast X-ray enhancement using a fuzzy gamma reasoning model," *Medical & Biological Engineering & Computing*, vol. 58, no. 6, pp. 1177–1197, 2020.

[24] J. P. Wang, F. Yang, and Y. Cao, "Cache-aware sampling strategies for texture-based ray casting on GPU," in *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 19–26, Paris, France, November 2014.

[25] https://challenge2018.isic-archive.com/.

[26] F. Yuan, "An interactive concave volume clipping method based on GPU ray casting with Boolean operation," *Computing and Informatics*, vol. 31, no. 3, pp. 551–600, 2012.

[27] L. Q. Campagnolo and W. Celes, "Interactive directional ambient occlusion and shadow computations for volume ray casting," *Computers & Graphics*, vol. 84, pp. 66–76, 2019.

[28] Y. Zhang, P. Gao, and X. Li, "A novel parallel ray-casting algorithm," in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, 2016.

[29] https://grand-challenge.org/All_Challenges/.

[30] http://academictorrents.com/details/a9e2741587d42ef6139aa474a95858a17952b3a5.

[31] L. Lin, S. Chen, Y. Shao, and Z. Gu, "Plane-based sampling for ray casting algorithm in sequential medical images," *Computational and mathematical methods in medicine*, vol. 2013, Article ID 874517, 5 pages, 2013.

[32] M. Bozorgi and F. Lindseth, "GPU-based multi-volume ray casting within VTK for medical applications," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 3, pp. 293–300, 2015.

[33] A. Mastmeyer, T. Hecht, D. Fortmeier, and H. Handels, "Ray-casting based evaluation framework for haptic force feedback during percutaneous transhepatic catheter drainage punctures," *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 3, pp. 421–431, 2014.

[34] http://adni.loni.usc.edu/data-samples/access-data/.

[35] R. Mehaboobathunnisa, A. A. H. Thasneem, and M. M. Sathik, "Fuzzy mutual information-based intraslice grouped ray casting," *Journal of Intelligent Systems*, vol. 28, no. 1, pp. 77–86, 2019.

[36] A. G. Moore, J. G. Hatch, S. Kuehl, and R. P. McMahan, "VOTE: a ray-casting study of vote-oriented technique enhancements," *International Journal of Human-Computer Studies*, vol. 120, no. 12, pp. 36–48, 2018.

[37] M. Fröhlich, C. Bolinhas, and A. Depeursinge, "Holographic visualisation and interaction of fused CT, PET and MRI volumetric medical imaging data using dedicated remote GPGPU ray casting," in *POCUS 2018, BIVPCS 2018, CuRIOUS 2018*, pp. 102–110, Granada, Spain, 2018.