Efficient Simulation of Light Scattering Effects in the Atmosphere

Huiling Guo[1]*, Xiliang Ren[2], Jing Zhao[3], Yong Tang[4]*
College of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, China[1, 3, 4]
Department of Information Engineering, Hebei University of Environmental Engineering, Qinhuangdao, 066102, China[1]
The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province,
Qinhuangdao, 066004, China[1, 2, 3, 4]
Qinhuangdao Bank Co., Ltd, Qinhuangdao, 066004, China[2]

***Abstract*—Atmospheric light scattering encompasses intricate physical process, including diverse scattering mechanisms and optical parameters. Addressing the challenges posed by the computationally intensive task of deciphering this phenomenon, this study introduces an efficient real-time simulation strategy. The proposed approach employs a physics-driven atmospheric modeling, leveraging a unified phase function to emulate both Rayleigh and Mie scattering phenomena. The scattering integral is approximated and discretized using the concept of ray-marching to solve the scattering integral. Based on the characteristics of different light sources, accurate ray-marching lengths are determined, streamlining the computational trajectory of the light path. Additionally, the introduction of texture dithering enhances the randomness of the initial sampling positions. The Shadow Map algorithm is adeptly employed to generate shadow mapping textures, eliminating the need for light calculations within shadowed regions, thereby reducing the number of samples and computational workload. Finally, color synthesis is used to determine the rendering color of the atmosphere under various fog density conditions. Experimental results show that this approach significantly improves rendering efficiency, and achieves real-time rendering while maintaining a realistic light scattering effect compared with other advanced light scattering rendering methods.**

Keywords*—Light scattering; ray marching; jittered sampling; color synthesis; real-time rendering

I. Introduction

The real-time simulation of atmospheric light scattering is essential for enhancing the realism of virtual scenes [1]. In movies, games, and virtual reality applications, being able to render realistic skies, lighting effects, and weather conditions in real time is crucial for improving users' visual experiences and sense of immersion [2]. Moreover, efficient light scattering simulation also has a significant impact on scientific research in climate change, environmental monitoring, and the field of computer graphics [3].

Currently, the real-time rendering of atmospheric light scattering effects faces two major challenges: one is the high computational complexity, as it requires consideration of the propagation of light through the atmosphere and various parameters such as atmospheric density and scattering coefficients, which involve complex integral calculations; the other is insufficient real-time performance, particularly in large scenes, where even with GPU hardware acceleration, achieving satisfactory computational efficiency and rendering speed remains challenging [4].

In response to the aforementioned issues, we propose an efficient simulation method tailored for the scattering effects of various light sources' rays in the atmosphere. Building upon the physically-based integral solution for light scattering, the method approximates and discretizes the scattering integral, enhancing the ray-stepping algorithm and reducing the length of the computed light paths. Additionally, the Shadow Map algorithm is employed to generate shadow mapping textures, eliminating the need for lighting calculations within shadowed areas, thereby further reducing the number of sampling points. This approach aims to strike a balance between computational efficiency and rendering quality, maintaining realistic simulation effects while enhancing rendering performance to meet the demands of real-time rendering of light scattering. The main contributions of this research are the as follows:

- Enhanced Henyey-Greenstein phase function for Rayleigh and Mie scattering intensities, simplified single-scattering model, and efficient multiple scattering integral computation.
- Optimized Ray-Marching with novel down-sampling method for various light source scenarios, reducing samples while maintaining rendering quality, significantly improving efficiency.
- Enhanced scene realism and 3D effects with optimized ambient and sunlight gradient effects under various times and weather conditions, using scene blending techniques for realistic light scattering.

The paper is structured as follows. Section II reviews previous studies. Section III delves into the construction and optimization of light scattering models, introducing novel techniques and methods. Subsequently, it optimizes the sampling strategy for efficient rendering and presents our approach to atmospheric color synthesis. Section IV presents results and discussions. Finally, this paper concludes in Section V.

II. Related Work

The simulation of light scattering effects relies on the computation of light scattering integrals, which describe the physical phenomenon where light changes its direction of

*Corresponding Author.

propagation due to interactions with particles in a medium. This complex process involves principles of wave optics and variables such as the size, shape, and refractive index of the particles. In the field of computer graphics, light scattering is key to creating realistic lighting effects, particularly when rendering scenes involving fog, smoke, clouds, and other participating media.

To accurately mimic these visual effects, researchers and developers have employed a variety of scattering models that approximate the true scattering behavior. Some widely used theoretical models and methods include: Rayleigh scattering [5], Mie Scattering [6], Henyey-Greenstein Phase Function [7], Monte Carlo Method [8], Light scattering integrals in ray tracing algorithms [9] and so on. Each model and method have its own range of applicability and trade-offs, and the choice often depends on the desired level of accuracy, computational resources, and specific application scenarios.

Currently, optical scattering models in atmospheric scenes are primarily categorized into two main types: empirical models and physical models. Empirical models are typically derived from measurements and statistics of physical parameters such as the shape, size, and concentration of gas particles, in order to deduce the scattering and absorption characteristics of these particles towards light. Empirical models are usually suitable for scenarios with low gas particle concentrations and regular particle shapes. However, physical based models are more applicable for gas environments characterized by complex particle distributions, diverse compositions, or varying properties. Hillaire [10] introduced a novel method for real-time evaluation of multiple light scattering within the atmosphere. By introducing a set of simplified lookup tables and parametrization techniques, it aims to efficiently render skies and their aerial perspectives. This method enables dynamic variations in atmospheric composition to align with artistic visions and weather conditions, eliminating the need for cumbersome LUT updating processes.

To improve the rendering efficiency of atmospheric scattering effects, on the one hand, advancements in computer hardware performance have been leveraged. Modern Graphics Processing Units (GPUs) are utilized for parallel computation and optimized algorithms to accelerate calculation speeds [11]. On the other hand, approaches based on analytical formulae, numerical approximations, and pre-computation are applied to reduce the complexity of integral calculations in atmospheric scattering models, thereby improving computational efficiency. Huo et al. [12] presented an adaptive matrix column sampling and completion method to accelerate the rendering of participating media. However, this method could only handle single scattering scenarios and was not applicable for rendering participating media in dynamic scenes. In 2020, West et al. [13] introduced a novel method called Continuous Multiple Importance Sampling (CMIS) to solve the problem of multiple importance sampling in Monte Carlo integral estimation. This method improves the efficiency of rendering materials, including participating media.

In order to more realistically reproduce light scattering effects, significant progress has also been made in the study of multiple scattering. László et al. [14] improved the traditional light-medium interaction model, allowing control of the extinction coefficient and control variables through approximated sampled values, thereby enhancing rendering efficiency. In 2019, Vibert et al. [15] presented a new scalable hierarchical VRL method that preferentially samples VRLs according to their image contribution, yet this method requires further improvement for rendering anisotropic media. Deng et al. [16] proposed a novel unbiased volume density estimator, the photon surface, which is combined through multiple importance sampling to handle ray paths including single scattering and within-medium transmission. In 2021, Alexander et al. [17] introduced a fitting model for skylight radiance and attenuation in real land atmospheres, significantly enhancing the visual authenticity of existing analytical clear-sky models and the visual realism of interactive methods based on approximate atmospheric light transmission. In the same year, Kettunen et al. [18] proposed a method for improving the efficiency of unbiased volume transmittance estimators. This method reduces variance through various means, resulting in estimators with several orders of magnitude lower variance at the same computational cost, thereby improving the efficiency of ray marching. In 2022, Korkin et al. [19] extended the scope of previous research by considering the reflection of polarized light by a Rayleigh scattering spherical atmospheric layer with highly correlated single-scatter absorption rates. They employed three advanced radiative transfer models to generate numerical results, covering both single scattering and multiple scattering scenarios.

Despite significant advancements in the study of light scattering effects, further exploration is still needed on how to better balance high rendering quality with real-time requirements. In response to the efficiency challenges for rendering atmosphere light scattering, a real-time simulation method for the scattering effects of light in the atmosphere is proposed. This method utilizes approximate numerical calculations and down-sampling to effectively enhance rendering efficiency. Additionally, scene blending techniques are employed to improve the rendering color, resulting in a more realistic portrayal of light scattering effects in the atmosphere.

III. Modeling and Simulation Optimization Methods

A. *Constructing Light Scattering Model*

Light scattering in the atmosphere mainly occurs through two processes: Rayleigh and Mie scattering. Rayleigh scattering, caused by tiny particles like air molecules, is why the sky looks blue and red during sunrise and sunset. Mie scattering, from larger particles like water droplets and aerosol particles, makes clouds and fog appear white. Our model focuses on these two types of scattering.

The relationship between the intensity of Rayleigh scattering and the wavelength of incident light, as well as the scattering angle, is expressed as shown in Eq. (1):

$$I(\theta) = I_0 \frac{\pi^2 (n^2 - 1)^2 \rho(h)}{2N\lambda^4} (1 + \cos^2 \theta) \tag{1}$$

Where $I(\theta)$ is the intensity of scattered light, λ is the wavelength of incident light, θ is the scattering angle, h is the height of the point, I_0 is the intensity of incident light, n is the refractive index of air, N is the density of air molecules at standard atmospheric pressure, $\rho(h)$ represents the relative density of air molecules at height *h*. When h=0, then $\rho(h)$=1. From this, it can be seen that the intensity of Rayleigh scattering is inversely proportional to the fourth power of the wavelength of the incident light. In other words, shorter wavelengths result in stronger scattering. In optics, the phase function is commonly used to describe the scattering properties of light as it interacts with materials. Since Rayleigh scattering is nearly isotropic, meaning that light is scattered uniformly in all directions by particles, its phase function is shown in Eq. (2):

$$F_R(\theta) = \frac{3}{16\pi}(1+\cos^2\theta) \quad . \tag{2}$$

In our model, approximate calculations are performed for Mie scattering, with extinction coefficients and asymmetry factors pre-computed. Combined with the improved Henyey-Greenstein phase function, Rayleigh scattering and Mie scattering can be uniformly described.

In the atmosphere, larger particles interfere with the light collected at the observation point. At this point, the light mapped to the observation point mainly comes from two parts: the light from target reflection attenuated by particles and reaching the observation point, and the atmospheric light formed by light source scattering through particles. Therefore, based on light energy transmission, an atmospheric scattering illumination model is constructed, and the total radiation rate received at point x is shown in Eq. (3):

$$I_c(x,\omega) = I_0(x,\omega)\mathrm{e}^{-\int_0^x \beta_{ex}(x')\mathrm{d}x'} + \int_0^x g(x,\omega)\mathrm{e}^{-\int_{x'}^x \beta_{ex}(x'')\mathrm{d}x''}\mathrm{d}x' \tag{3}$$

Where ω is the incident direction, $I_c(x,\omega)$ is the outgoing light intensity at position *x*, $I_0(x,\omega)$ is the incident light intensity at position *x*, β_{ex} is the extinction coefficient, and $g(x,\omega)$ is the scattering distribution intensity at position *x*. The first part of this equation represents the intensity of light transmitted directly from the light source to the observation point, taking into account the absorption attenuation of light. The second part represents the scattering process through the medium, considering the scattering attenuation of light.

Eq. (4) represents the sum of light intensity scattered from direction ω at point *x*, where the rays from different directions ω_i interact with the medium at that location.

$$g(x,\omega) = \beta_{sc}\int_{4\pi} I(x,\omega_i)F(\omega,\omega_i)\mathrm{d}\omega_i \tag{4}$$

Where $\beta_{sc}=\beta\rho$ is the scattering coefficient, β is a tunable parameter, ρ represents the atmospheric density ratio to simulate atmospheric density, $I(x,\omega_i)$ is the incident light intensity from ω_i, and $F(\omega,\omega_i)$ is the phase function of the scattering medium. The angle between ω and ω_i is denoted as θ, which is the scattering angle. Therefore, the phase function can be expressed as $F(\theta)$.

Since this incident intensity is a collection of light emitted from various directions in the sky domain and lacks a specific directionality as a whole, a unified phase function can be applied here. Due to the complexity of the true physical functions for Rayleigh and Mie scattering, using an approximation for the phase function $F(\theta)$ can significantly reduce computational complexity. The directional characteristics of the scattering model vary with particle size. Unlike Rayleigh scattering, the direction of Mie scattering is anisotropic, where light is more scattered forward. Different scattering characteristics can be constructed by combining various linear phase functions and adjusting the values of the asymmetry factor. The Henyey-Greenstein phase approximation function is shown in Eq. (5):

$$F(\theta) = \frac{(1+\cos^2\theta)}{(1+g^2-2g\cos(\theta))^{3/2}} \tag{5}$$

Where g is the asymmetry factor. However, this phase function can only describe forward scattering and cannot accurately simulate the effects of backward scattering. To address this issue, an improved Henyey-Greenstein phase function, as shown in Eq. (6), was adopted to achieve the simulation of backward scattering while avoiding the introduction of excessive complexity coefficients.

$$F(\theta) = \frac{1}{4\pi}\frac{3(1\text{-g}^2)}{2(2+g^2)}\frac{(1+\cos^2\theta)}{(1+g^2-2g\cos(\theta))^{3/2}} \tag{6}$$

The value range of *g* is [-0.75, 0.99]. When *g* is negative, it corresponds to forward scattering, and when g is positive, it corresponds to backward scattering. When the g-value is 0, it results in isotropic scattering, which manifests as Rayleigh scattering.

B. Simplified Integral Solution and Multiple Scattering

During the propagation of light, particles in the air cause scattering of the light. When there are fewer particles in the air, light is usually scattered only once. However, in an atmosphere with a lot of larger particles, the scattered light may continue to be scattered by other particles, resulting in a multiple scattering effect. In conditions where the air quality is poor, multiple scattering can significantly affect our perception of the scene. Computing multiple scattering is complex as it involves extensive integration calculations, making real-time rendering challenging. To address this issue, one can utilize the fact that within a certain area, the variation in particle concentration is typically gradual. Therefore, it's not necessary to calculate the scattered light intensity for every single particle; instead, one can compute the scattered light intensity of some particles along the line of sight to represent the whole.

The specific method, as shown in Fig. 1, involves dividing the air space through which the line of sight passes into several segments, taking the average concentration of like particles within each segment. This allows for segmented sampling based on the variations in atmospheric particle concentration and incident light intensity with distance. The size of each segment needs to be determined by considering the changes in both atmospheric particle concentration and incident light intensity to ensure they remain roughly constant within each segment. The number of segments can be adjusted based on the required drawing precision. This method avoids point-by-point sampling along the line of sight, thereby significantly reducing the computational load.

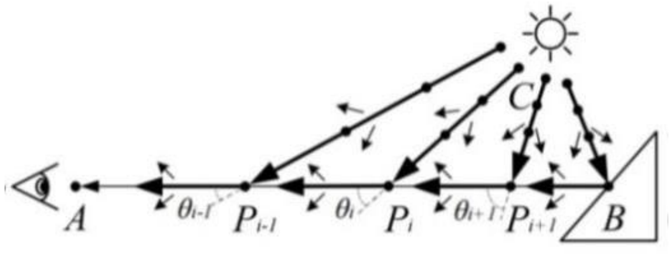


Fig. 1. Schematic diagram of scattered light intensity calculation.

Uniformly select N sampling points along the path AB, denoted as $P_1, P_2 \ldots\ldots P_N$. The total light intensity scattered into the viewer's eye along the line of sight is the sum of particle-scattered light intensities from N sampling segments along the line of sight. After these simplifications, Eq. (3) can be simplified to Eq. (7).

$$I_c = \sum_{i=1}^{N} I_c^i \tag{7}$$

Similarly, when calculating the attenuation coefficient along path CP_i, sampling is also required, with the number of selected points denoted as M. The more sampling points there are, the closer the result is to Eq. (3). However, having too many sampling points can impact real-time performance, necessitating a careful balance. Therefore, Eq. (3) can be discretized into a summation form to significantly simplify the computational complexity, as shown in Eq. (8) and Eq. (9).

$$I_c^{(\upsilon)}(x,\omega) \approx I_0^{(\upsilon)}(x,\omega)\prod_{i=1}^{N} \mathrm{e}^{\beta_{exi}^{(\upsilon)}\Delta x} + \sum_{i=1}^{N} g(x,\omega)\prod_{j=i+1}^{N} \mathrm{e}^{\beta_{exj}^{(\upsilon)}\cdot\Delta x} \tag{8}$$

$$g(x,\omega) \approx \sum_{i=1}^{M} I_i^{(\upsilon)} \beta_i^{(\upsilon)} F^{(\upsilon)}(\theta) \tag{9}$$

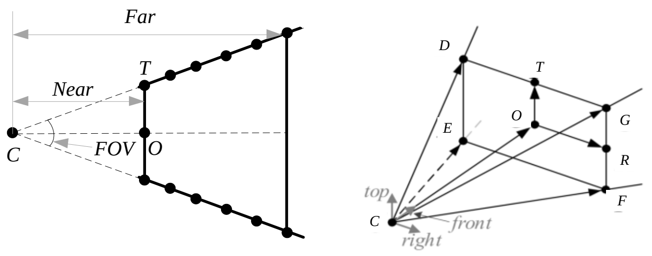
where υ represents the number of multiple computation iterations, N signifies the count of particles with different scattering properties, and M denotes the surrounding voxels that have already been calculated.

Ray-Marching is a ray stepping technique that works by shooting rays from the viewpoint and advancing them step by step, calculating the distance to the surface of objects at each step until a predefined termination condition is met. Its advantages over other methods lie in its ability to achieve extremely smooth effects and handle complex geometries. Rendering performance can be improved by reducing the calculated length of light paths, decreasing the number of samples, and avoiding the computation of unnecessary sample points.

C. Reducing the Length of Ray Marching

When using Ray-Marching for rendering, it's generally assumed to start from the camera position and sample along rays emanating from it until the ray reaches the camera's far clipping plane or intersects with an object. Lighting is a necessary condition for scattering, so it's only necessary to perform ray marching sampling within the range of the lighting. There's no need to march rays throughout the entire scene, which can help reduce the length of the calculated sampling path and improve rendering efficiency. However, different light sources possess varying lighting ranges and other characteristics.

1) Directional Light Ray Marching Path: Rays emitted by directional light sources are mutually parallel, and the illumination can cover the entire scene. Therefore, ray marching needs to take place between the camera's near clipping plane and far clipping plane. The starting point of the ray can be adjusted from the camera's position to its near clipping plane. The position of this new starting point can be determined by the geometric relationship between the camera and the near clipping plane, as shown in Fig. 2.



(a) Directional light ray marching range (b) The relative position of the camera and the near clipping plane

Fig. 2. Schematic diagram for calculating the distance between each point on the near clipping plane and the camera.

In Fig. 2(a), *FOV* represents the opening angle of the visual cone in the vertical direction, *Near* and *Far* respectively indicate the distance from the camera to the near clipping plane and the far clipping plane. In Fig. 2(b), the plane DEFG is the near clipping plane. $\overrightarrow{top}$, $\overrightarrow{right}$ and $\overrightarrow{front}$ represent the camera's upward, rightward, and forward directions. From this, the vectors to the four corner points of the near clipping plane from the camera can be obtained as shown in Eq. (10)-Eq. (13).

$$\overrightarrow{CD} = \overrightarrow{CO} + \overrightarrow{OT} - \overrightarrow{OR} \tag{10}$$

$$\overrightarrow{CE} = \overrightarrow{CO} - \overrightarrow{OT} - \overrightarrow{OR} \tag{11}$$

$$\overrightarrow{CF} = \overrightarrow{CO} - \overrightarrow{OT} + \overrightarrow{OR} \tag{12}$$

$$\overrightarrow{CG}=\overrightarrow{CO}+\overrightarrow{OT}+\overrightarrow{OR} \tag{13}$$

The current camera's aspect ratio is *Aspect*. Based on the positions corresponding to the sampling points in Fig. 4, $\overrightarrow{CO}$, $\overrightarrow{OT}$, and $\overrightarrow{OR}$ can be obtained as shown in Eq. (14)- Eq. (16).

$$\overrightarrow{CO}=\overrightarrow{front}\cdot Near \tag{14}$$

$$\overrightarrow{OT}=\overrightarrow{top}\cdot\left|\overrightarrow{OT}\right|=\overrightarrow{top}\cdot(Near\times\tan\frac{FOV}{2}) \tag{15}$$

$$\begin{aligned}\overrightarrow{OR}&=\overrightarrow{right}\cdot(\left|\overrightarrow{OT}\right|\times Aspect)\\&=\overrightarrow{right}\cdot(Near\times\tan\frac{FOV}{2}\times Aspect)\end{aligned} \tag{16}$$

Substituting Eq. (14)-(16) into Eq. (10)-(13), the computed results are passed to the vertex shader in the form of three-dimensional vectors. The rendering pipeline will automatically interpolate these vectors for each fragment. Subsequently, in the fragment shader, the interpolated vectors can be used to calculate the ray starting point corresponding to each pixel.

2) Point Light Ray Marching Path: Unlike directional light, the illumination range of a point light source is localized and forms a sphere. The actual effective range for ray marching is the intersection between the ray and the lighting sphere. As shown in Fig. 3.

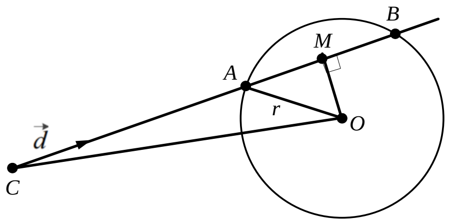


Fig. 3. Analysis of Intersection between light and point light sphere.

The ray emitted from point *C* intersects the sphere with radius *r* and center *O* at points *A* and *B*, where $\vec{d}$ is the unit vector representing the ray direction. According to the definition of vector dot product, the length of the line segment CM is given by Eq. (17), and the square of the length of AM is given by Eq. (18).

$$\left|\overrightarrow{CM}\right|=\vec{d}\cdot\overrightarrow{CO} \tag{17}$$

$$\left|\overrightarrow{AM}\right|^2=r^2-\left|\overrightarrow{OM}\right|^2=r^2-(\left|\overrightarrow{CO}\right|^2-\left|\overrightarrow{CM}\right|^2)\ . \tag{18}$$

According to Formulas (17) and (18), it can be deduced that the intersection point vector between the ray and the point light source's sphere is given by Eq. (19).

$$\begin{cases}\overrightarrow{CA}=\overrightarrow{CM}-\overrightarrow{AM}=(\vec{d}\cdot\overrightarrow{CO}-\sqrt{r^2-(\overrightarrow{CO}-\vec{d}\cdot\overrightarrow{CO}\cdot\vec{d})\cdot\overrightarrow{CO}})\cdot\vec{d}\\\overrightarrow{CB}=\overrightarrow{CM}+\overrightarrow{AM}=(\vec{d}\cdot\overrightarrow{CO}+\sqrt{r^2-(\overrightarrow{CO}-\vec{d}\cdot\overrightarrow{CO}\cdot\vec{d})\cdot\overrightarrow{CO}})\cdot\vec{d}\end{cases} \tag{19}$$

From this, the starting and ending positions for the ray marching can be determined.

3) Spotlight Ray Marching Path: Similar to point lights, spotlights also have a localized lighting range and exhibit a conical shape, as shown in Fig. 4.

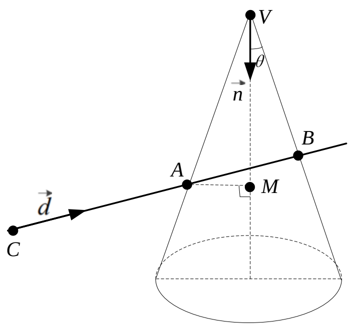


Fig. 4. Analysis of intersection between light and spotlight cone.

The rays emitted from point *C* intersect the cone at points *A* and *B*. Where $\vec{d}$ and $\vec{n}$ are both unit vectors representing the directions of the ray and the axis of the cone, respectively. θ is the angle between the axis of the cone and the generatrix. Therefore, the vector from point *C* to the intersection points between the ray and the cone can be expressed as shown in Eq. (20).

$$\overrightarrow{CA}=t\vec{d} \tag{20}$$

where *t* is the parameter along the ray. Based on the relationships in Fig. 6, Eq. (21) and Eq. (22) can be derived as follows:

$$\overrightarrow{VA}\cdot\vec{n}=\left|\overrightarrow{VA}\right|\cdot\left|\vec{n}\right|\cdot\cos\theta=\left|\overrightarrow{VA}\right|\cdot\cos\theta \tag{21}$$

$$\overrightarrow{VA}=\overrightarrow{VC}+\overrightarrow{CA} \tag{22}$$

By substituting Eq. (20) and Eq. (22) into Eq. (21) and solving, we can obtain the parameter *t* at which the ray intersects with the lighting cone, thus obtaining the coordinates of intersection point *A*. Similarly, the coordinates of intersection point *B* can be obtained. It's worth noting that due to light attenuation or obstruction by objects, the lighting cone has a certain height limitation. If the height of the lighting cone is set to *h*, it can be determined whether there is an intersection between the ray and the lighting cone by evaluating $0\le\left|\overrightarrow{VA}\right|\cos\theta\times h\le h$.

D. Reducing Sampling Number

During ray marching, increasing the number of sampling can lead to lower simulation efficiency. Therefore, minimizing the number of samples is crucial. However, excessively reducing the sample count may result in noticeable artifacts or banding in the image, significantly affecting realism. This is because larger intervals between sampling points fail to capture sufficient lighting information. From an image processing perspective, this issue belongs to quantization errors and can be mitigated by using dithering. Texture dithering introduces a certain level of randomness in atmospheric light scattering simulations to replicate the complex variability of atmospheric lighting phenomena in the real world. Due to the non-uniform distribution of atmospheric particles in the actual environment, light scattered by these particles creates a natural, seemingly random effect visually. By applying subtle random noise to the image, texture dithering achieves random offsets corresponding to screen pixels, breaking up the regular patterns that quantization errors might otherwise introduce. This method enhances the randomness of sampling points, not only improving the realism of simulated scenes but also optimizing rendering performance without increasing additional computational burden.

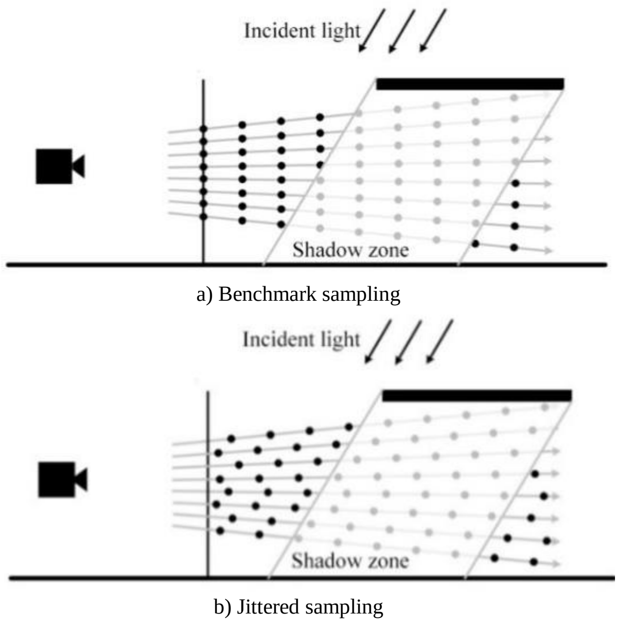


Fig. 5. Comparison of benchmark sampling and jitter sampling.

Using the sampling starting positions calculated in Section 4.1 as a reference, Fig. 5(a) depicts the benchmark sampling where the offset is 0. Fig. 5(b) shows the jitter sampling, which adds a random offset on the benchmark starting sampling position, allowing the sampling points between different rays to be staggered. It is worth noting that the random offset should be less than the step size value of the ray marching.

Jittered sampling effectively transforms the color band resulting from reducing the sampling count into noise point in the image. Common methods for generating jitter maps are based on Bayer matrix, white noise, and blue noise. Fig. 6 shows the sampling results under different jitter textures.

When the sampling count is set to 15, due to the limited number of samples, there will be significant banding distortion at the shadow without using the jitter sampling method, as shown in Fig. 6(a). The jitter texture generated using the Bayer matrix has strong regularity and generates duplicate lattice points, as shown in Fig. 6(b). The white noise jitter texture generates more noise points, resulting in poor performance, as shown in Fig. 6(c). The results generated using blue noise jitter textures exhibit relatively fewer noise points and higher randomness, which is beneficial for subsequent blur processing, as shown in Fig. 6(d). In comparison, the image quality generated by blue noise is better. Therefore, using blue noise to generate jitter textures is recommended. After jitter sampling, the generated noise can be removed using a Gaussian bilateral blur method, while preserving the clear contours of the image. The Gaussian bilateral blur weights of pixels are calculated as shown in Eq. (23).

$$\omega_i(x_i, d_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{x_i^2}{2\sigma^2} - d_i^2) \tag{23}$$

The color contribution value of the *i*-th pixel with a relative distance of x_i and a relative depth of d_i from the target pixel can be obtained by Eq. (23).

Where σ is the standard deviation, and the larger the value, the stronger the blurring effect. These weights need to be normalized before it can be used. The normalized pixel weights are computed as shown in Eq. (24).

$$\varpi_i(x_i, d_i) = \frac{\omega_i(x_i, d_i)}{\sum_{i=1}^{N} \omega_i(x_i, d_i)} \tag{24}$$

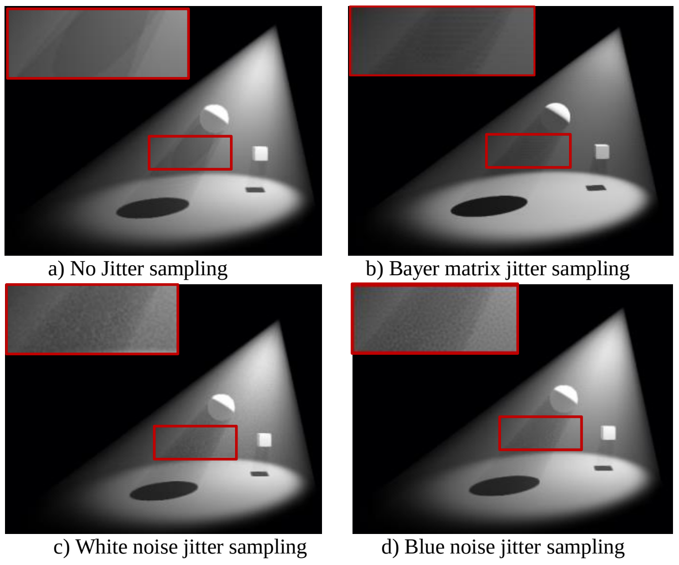

a) No Jitter sampling
b) Bayer matrix jitter sampling
c) White noise jitter sampling
d) Blue noise jitter sampling

Fig. 6. Comparison of rendering effects of different jitter textures.

E. Reduce the Range of Lighting Calculation

During the process of ray marching, some sampling points might be situated within shadows, as shown in Fig. 5. These sampling points within shadows do not contribute to scattering light. If we can quickly determine whether a sampling point is in shadows, it would help reduce computational workload and enhance simulation efficiency.

Shadow Map is a simple and fast shadow rendering algorithm, which means that when viewed from the position of the light source, all visible objects are illuminated by the light source, and all occluded and invisible objects are in shadow. Firstly, the light source is treated as the camera to render the entire scene and obtain the depth of each object. This depth information is then saved as a texture, which is referred to as the shadow mapping texture. Next, the scene is rendered using the camera's perspective, obtaining vertex coordinates. Transforming these vertices into the light source's coordinate space yields their clipping coordinates (x, y, z) within that space. By utilizing the transformed vertex coordinates (x, y), a sampling of the shadow mapping texture provides the maximum depth value z' for the illumination. If $z < z'$, then the vertex is in the shadow, otherwise it is not in the shadow.

By utilizing the Shadow Map, we can quickly determine whether each sampling point is in shadow. For points within shadows, light intensity calculations can be skipped, as their radiance is considered to be zero since they do not contribute to the final scattered light intensity, effectively reducing the number of sampling points for which light computations are required, thus significantly enhancing rendering efficiency.

F. *Improving the Ray Marching Rendering Algorithm*

Compared to the traditional Ray-Marching algorithm, the improved Ray-Marching rendering algorithm enhances sampling efficiency, and its pseudo code is shown in Algorithm 4.1.

Algorithm 4.1 Improved Ray Marching Rendering Algorithm:

```
Input: Pixel information X, Number of ray marching samples N, Light source color Ic
Output: Color corresponding to pixel points
1: Function ImprovedRayMarching(X, N, Ic):
2: W = CalculateWorldCoordinates(X)
3: StartPoint = CalculateStartPoint(W)
4: R = ApplyJitterSampling(StartPoint)
5: S = GetStepSize(R, N)
6: I = 0
7: For i = 1 to N:
8:        If InShadowRegion(R):
9:            Continue
10:       End If
11:       Ii = CalculateIncidentLightIntensity(R)
12:       ρ = CalculateFogDensity(R)
13:       I0 = CalculateScatteredLightIntensity(Ii, ρ)
14:       I += I0
15: End For
16: Pc = Ic * I
17: Return Pc
18: End Function
```

When performing atmospheric rendering based on the improved Ray-Marching approach, the initial stage involves recalculating the starting position and step size for ray marching along the direction of rays emitted from the camera. Additionally, jittered sampling is applied along the ray direction. If a ray intersects with an object, its advancement is halted, and distance information is returned. Furthermore, the sampled data and coverage information are utilized to calculate the atmospheric density. Subsequently, the computation of fog area shadows is performed. Rays situated within the shadowed region do not necessitate scattering calculations, and efficiency can be enhanced by excluding these rays. Afterward, atmospheric color values are sampled at the current ray position, and a scene shadow map is generated. This shadow map is then stored in the command buffer for later storage in a texture. Then, it's combined with sunlight and ambient light colors. In addition, lighting calculations are necessary, including extinction coefficient, scattering contribution, and transmittance, among others. After completing these calculations through iterative processes, the final atmospheric color is generated.

G. *Atmospheric Color Synthesis*

In order to enhance the representation of lighting details, the light scattering effect in fog is simulated using scene blending techniques. Ambient light, as diffused rays, pervades the entire scene and traverses through dynamically changing fog regions. The fog is denser near the ground, resulting in deeper colors. The ambient light generated based on height and time will influence the color of the fog region. Here, a negative cosine function curve is introduced and the time difference range is extended to $[0,\ 2\pi]$. The ambient light color for different heights and times is obtained by multiplying with the ratio of the height field, calculated as shown in Eq. (25).

$$C_{\text{amb}} = \left[\frac{H_{\text{cur}}}{2H_{\text{total}}} \cdot (1 - \cos((\frac{T_{\text{cur}} - T_{\text{start}}}{T_{\text{end}} - T_{\text{start}}}) \cdot 2\pi)) C_{\text{inc}} + C_{\text{stan}} \right] \tag{25}$$

Where $H_{\text{cur}} / 2H_{\text{total}}$ represents the height field ratio, T_{cur} is the current time, thus the range of $(T_{\text{cur}} - T_{\text{start}}) / (T_{\text{end}} - T_{\text{start}})$ is [0, 1], C_{inc} is the incremental color, and C_{stan} is the standard ambient light color. The color and position of sunlight dynamically change with time. Therefore, the corresponding color domain is defined, and the current color value of sunlight is obtained through interpolation based on the time ratio, as shown in Eq. (26).

$$C_{\text{sun}} = [\frac{T_{\text{cur}} - T_{\text{start}}}{T_{\text{end}} - T_{\text{start}}} \cdot C_0 + (1 - \frac{T_{\text{cur}} - T_{\text{start}}}{T_{\text{end}} - T_{\text{start}}}) \cdot C_{\text{s-end}}](V_{s \to f} \cdot V_{v \to f}) \tag{26}$$

Where C_0 is the starting value of the sunlight color domain, $C_{\text{s-end}}$ is the ending value of the sunlight color domain, $V_{s \to f}$ represents the vector from the sun to the center of the fog, and $V_{v \to f}$ is the vector from the vertex to the center of the fog.

On the other hand, employing the Alpha blending technique with translucent textures enhances the realism of the fog. To further improve real-time performance, a distance threshold *l* is set to divide the rendering range. Particles within the threshold *l* are only rendered with the particle fog color, while those beyond the threshold *l* exhibit varying lighting effects using UV gradient animation and BillBoard techniques. As shown in Eq. (27):

$$C_{final} = \begin{cases} C_{\text{fog}} & d < l \\ (1-\alpha)\cdot(C_r + C_{\text{sun}} + C_{\text{amb}}) + \alpha \cdot C_{\text{fog}} & d \geq l \end{cases} \quad (27)$$

where C_{final} represents the blended rendered color, C_r is the fragment scene color, C_{fog} is the fog color, α is the blending coefficient, which is used to control the degree to which the fog color is influenced by the light.

IV. Discussion and Analysis of Results

To validate the effectiveness of the real-time rendering algorithm for light scattering effects proposed in this paper, simulations of atmospheric light scattering were conducted using Unity. The rendering quality and efficiency of the algorithm were demonstrated, and its performance was evaluated across multiple test scenarios by fine-tuning various algorithmic parameters to assess its robustness in different settings. The experimental environment of the testing platform includes the following hardware components: Intel(R) Core (TM) i7-5820K CPU @3.30 GHz, 16 GB RAM, and the graphics processor is NVIDIA GeForce GTX 1060. The main software used includes OS: Windows 10 (64 bit), development platform: Unity 3D engine, Visual Studio 2017, and the combination of C # scripting language and CG/HLSL shading language to design and render different scenes.

A. *Comparison of Scattering Effects under Different Light Sources*

The sampling range for ray marching varies under different light sources. Rendering the scene at a resolution of 1280×720 with the same sampling rate, Fig. 7 shows the scattering effects generated under different light sources.

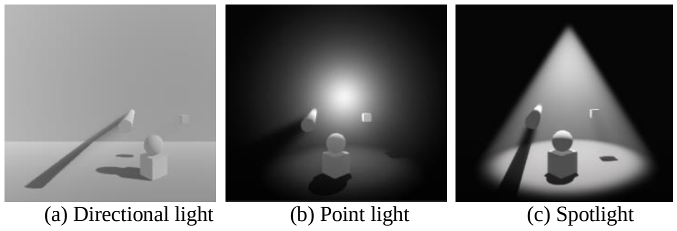

(a) Directional light (b) Point light (c) Spotlight

Fig. 7. Comparison of light scattering effects under different light sources.

Fig. 7(a) shows a directional light that generates light scattering throughout the entire visual space. When using the traditional ray marching algorithm, the average rendering efficiency is 76.4 FPS. However, with the method proposed in this paper, rendering efficiency has improved to 89.5 FPS, representing a performance increase of 17%. Fig. 7(b) represents a point light source, with its scattering range forming a spherical shape. Fig. 7(c) shows a spotlight, generating a cone-shaped scattering range. Due to their radiation characteristics, point lights and spotlights require more ray sampling to capture variations of light in different directions. Because the scattering range of light in this scene is relatively concentrated, rendering efficiency demonstrates a slight improvement.

B. *Comparative Analysis of Rendering Effects with Different Sample Counts*

Fig. 8 shows a comparison of volumetric lighting rendering effects in the small town scene with different sample counts (N).

Fig. 8(a)-(e) shows the volumetric lighting rendering effects without Gaussian bilateral blur processing, while Fig. 8(f)-(j) shows the volumetric lighting rendering effects for the corresponding sample counts after the application of blur processing. In 1920×1080 resolution, when the sample count is low, as shown in Fig. 8(a) and 8(b), a lack of sufficient lighting information leads to the presence of numerous artifacts in the scene and the absence of beam effects. Upon applying the blur processing, extensive light spots are formed within the lighting area, as shown in Fig. 8(f) and 8(g). This results in significant distortion of the volumetric lighting rendering effect in the scene. As the number of samples increases, as shown in Fig. 8(c), the lighting area begins to exhibit noticeable beam effects, although some noise points still exist. With further increase in sample counts, more lighting information is gathered, resulting in a more refined and realistic beam effect. When the sample count reaches 256, the generated beam effect becomes quite realistic, as shown in Fig. 8(e). After blur processing, it can be observed that when the sample count reaches 64, the rendered lighting effect appears highly realistic. The beam boundary transition is natural, and the result is very close to the rendering achieved with higher sample counts, as shown in Fig. 8(h)-(j). This implies that, for practical purposes, the rendering effect achieved with a sample count of 64, enhanced by the blur process, can effectively substitute for higher sample counts. The rendering frame rates for different sample counts are shown in Table I. Increasing the sample count enhances rendering quality, but it also raises computational costs. In this complex scene, the rendering efficiency with 64 samples and blur processing can reach 46.4 FPS, which is 82% higher than the rendering efficiency without blur processing with 256 samples.

Therefore, our approach is able to achieve a more efficient and artifact-free realistic atmospheric lighting effect with fewer samples, achieving a balance between the desired visual quality and performance.

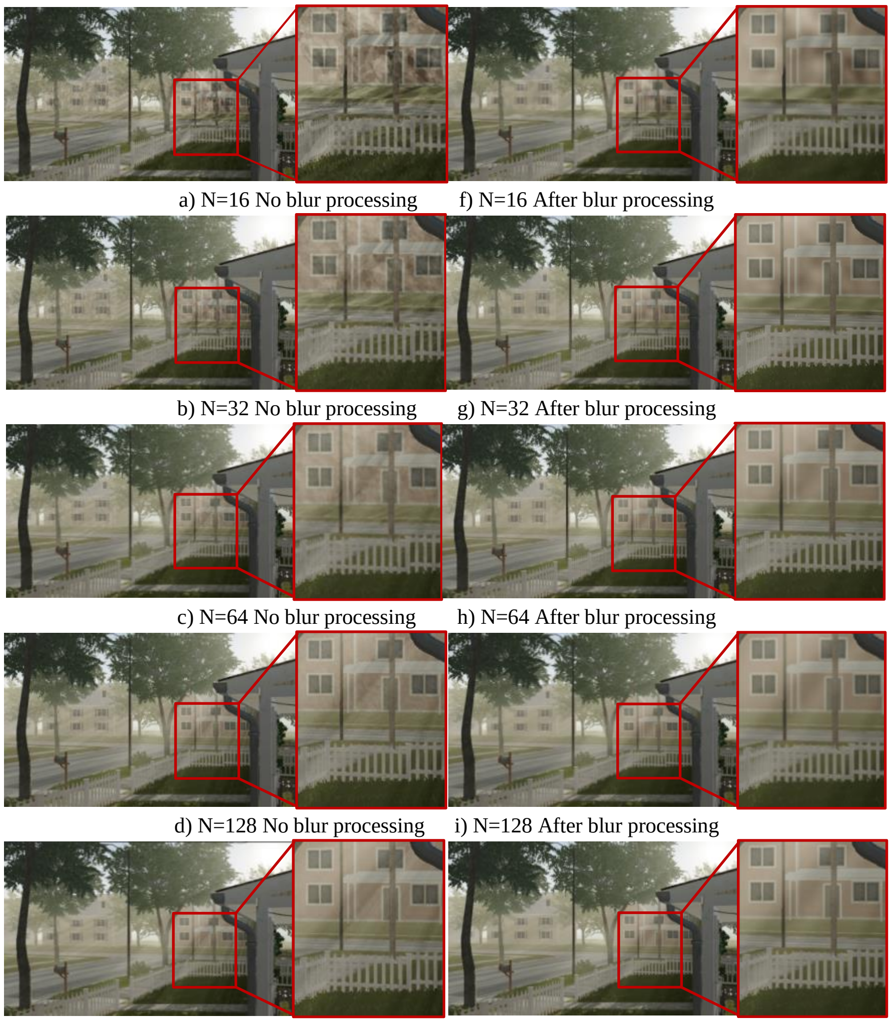

a) N=16 No blur processing
f) N=16 After blur processing
b) N=32 No blur processing
g) N=32 After blur processing
c) N=64 No blur processing
h) N=64 After blur processing
d) N=128 No blur processing
i) N=128 After blur processing
e) N=256 No blur processing
j) N=256 After blur processing

Fig. 8. Comparison of volume light rendering effects with different sample counts.

C. *Time-Varying Ray Scattering Effect*

Fig. 9 shows the comparison of light scattering effects in non-uniform media that dynamically vary with the position of the sun.

Fig. 9(a) and (c) respectively show the light scattering effect under dynamic mountain fog in the morning and after sunset. At these times, the sun is near the horizon, and the sunlight enters the atmosphere at an oblique angle. This leads to significant scattering of blue light along the transmission path, preventing it from reaching the camera. As a result, the closer to the ground, the sky appears orange- yellow or orange-red. Fig. 9(b) shows the scene at noon, where the sky displays a gradient blue due to Rayleigh scattering. However, a noticeable mist-like effect is observed near the ground due to Mie scattering, significantly enhancing the realism of the entire scene. Fig. 9(d) shows the scene at midnight, without considering the influence of light from other celestial bodies, presenting a completely dark effect.

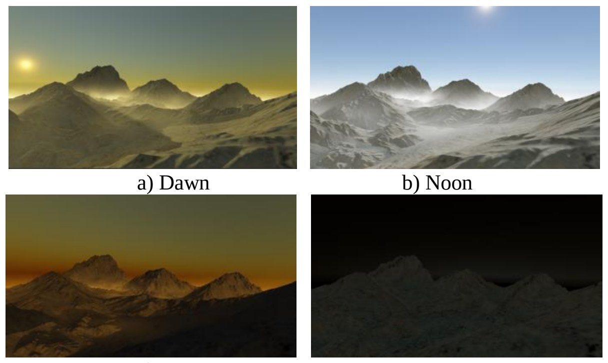


Fig. 9. Comparison of time-varying light scattering effects in non-uniform media.

This indicates that the proposed method can effectively present dynamic light scattering effects caused by varying solar positions, enhancing the realism of the rendered scenes. Moreover, it can maintain a rendering efficiency of over 45FPS, meeting real-time requirements.

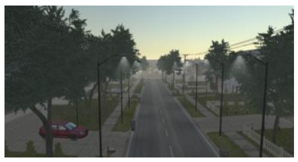
D. Rendering Effects for Different Atmospheric Density Ratios

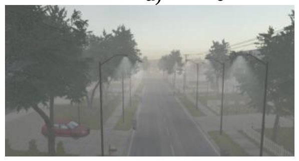
The concentration of aerosol particles significantly affects the propagation, color, and intensity of light. Due to the fact that most air pollutants belong to aerosol particles, the intensity of Mie scattering can be affected by adjusting the atmospheric density ratio, thereby simulating the light scattering effect under different air qualities.

Fig. 10(a) presents the lighting scene of an urban street in the early morning, where the atmosphere does not contain aerosol particles. As a result, only the Raleigh scattering effect is evident in the distant sky, and there is no Mie scattering effect near the ground. The light from street lamps is primarily concentrated near the light sources, and there is no noticeable scattering effect. In Fig. 10(b), $\rho=0.1$, the lighting scene under good air quality is shown. In this case, the Mie scattering effect is evident near the ground. Fig. 10(c) and Fig. 10(d) respectively show the light scattering effects for conditions with $\rho=0.3$ and $\rho=0.5$. In both cases, the entire scene is covered by fog, with street lamps forming light beams within the fog.

As the atmospheric density ratio increases, the light scattering effects gradually enhances. The clarity and visibility of objects in the scene significantly diminish, and the sky gradually appears gray blue, in complete accordance with the physical principles of light scattering. This notably enhances the scene's depth and layering, and effectively elevating the realism of light scattering simulation under various levels of atmospheric pollution.

a) ρ =0


b) ρ =0.1


c) ρ =0.3

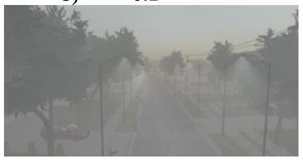
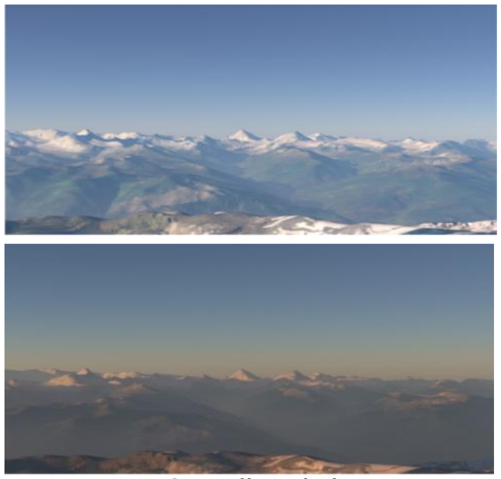

d) ρ =0.5

Fig. 10. Comparison of light scattering effects under different atmospheric density ratios.

E. Experimental Comparison and Results

Fig. 11 shows the comparison of light scattering effects between this paper and reference [17] in mountain scenes under clear skies.

Fig. 11(a) illustrates the rendering results achieved by the method used in literature [17], which is based on an extensive atmospheric scattering dataset. This technique employs data fitting to reduce the size of the dataset and attain high-quality rendering effects. Despite its ability to produce images with a high degree of realism, the method's substantial demand for computational resources and storage space limits its applicability in real-time rendering scenarios. In contrast, Fig. 11(b) presents the light scattering effects realized in large-scale scenes using the method proposed in this paper. By adopting the same solar elevation angle as in literature [17], with the top image set at 15 degrees and the bottom image at 2 degrees, our method achieves visually realistic results comparable to those in literature [17]. Meanwhile, the average rendering frame rate of our method reached 43.6 FPS, fulfilling the requirements for real-time performance. The Mie scattering effect is not obvious in the near area, while it becomes more evident in the far area. This results in clear visibility of objects in the foreground and a gradual blurring and fading of objects in the distance, in accordance with the natural physics of light. Furthermore, as the elevation increases, the Mie scattering effect gradually weakens. And after reaching a certain height, it can still maintain a clear sky color, effectively achieving a realistic light scattering effect under a clear sky. It is worth noting that the frame rate of this method can reach 43.6 FPS, fully meeting the requirements of real-time rendering. In conclusion, our approach significantly improves rendering efficiency while maintaining light scattering effects comparable to those in reference [17]. This not only meets the requirements for real-time rendering but also satisfies the demands of applications with high real-time performance requirements.


a) Rendering effects of reference

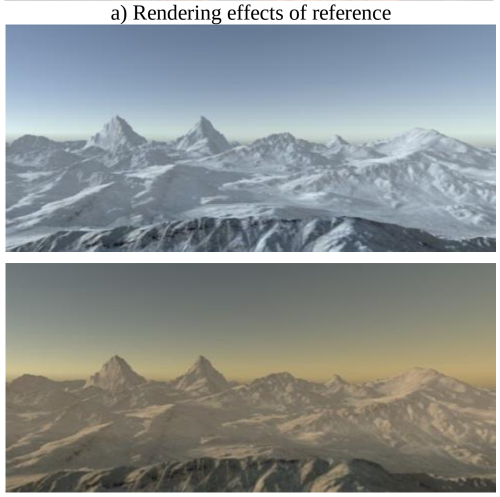
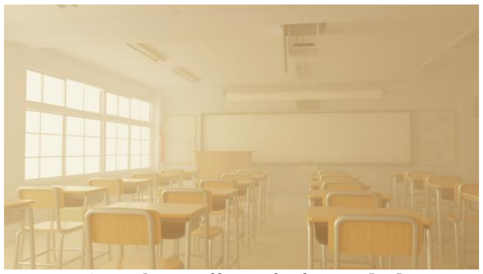

b) Rendering effect of this paper

Fig. 11. Comparison of light scattering effects in mountain scenes between this paper and reference [17].


a) Rendering effects of reference [16]

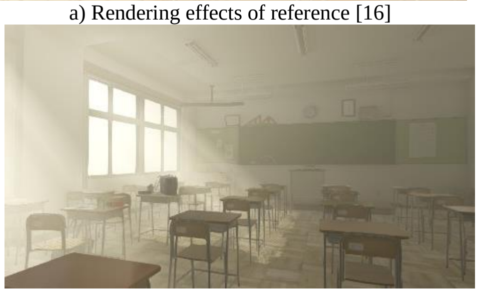

b) Rendering effect of this paper

Fig. 12. Comparison of indoor lighting scattering effects between this paper and reference [16].

The above experiments were all conducted in outdoor scenes, and our approach is equally applicable to indoor environments. Fig. 12 provides a comparative display of the light scattering effects in a classroom scene between our approach and the method proposed in reference [16].

Fig. 12(a) illustrates the rendering of a scene using the photon surface density estimation method from literature [16], which is suitable for handling cases with highly anisotropic phase functions. However, it fails to meet the requirements for real-time rendering. In contrast, Fig. 12(b) displays the rendering results obtained using the method proposed in this paper, achieving a frame rate as high as 76 FPS, meeting the standards for real-time performance. Moreover, the distribution of light and shadow in our method is similar to that in literature [16], presenting a trend of gradually diminishing brightness from the window to the interior of the room. Our method also achieves the effect of sunlight passing through glass windows, producing noticeable volumetric light beams. This greatly enhances the realism of the scene.

F. Performance Analysis

Table I lists the relevant data for each experimental scenario in this paper. The experimental data indicates that the method proposed in this paper can achieve real-time rendering frame rates in various experimental scenarios, with the sampling numbers not exceeding 256. Particularly, in smaller-scale scenes, higher rendering frame rates can be achieved. Compared to the methods in references [16], our approach has significant advantages in real-time rendering performance while maintaining consistent rendering quality.

TABLE I. EXPERIMENTAL DATA OF DIFFERENT EXPERIMENTAL SCENARIOS

Scenes	Sample Count	Atmospheric Density Ratio ρ	Scattering Coefficient	Resolution	Average Rendering Frame Rate (FPS)
Fig.7 a)	64	0.1	0.33	1280×720	89.5
Fig.7 b)	64	0.1	0.33	1280×720	96.5
Fig.7 c)	64	0.1	0.33	1280×720	98.6
Fig.8 a) (No blur)	16	0.1	0.29	1920×1080	54.2
Fig.8 c) ((No blur))	64	0.1	0.29	1920×1080	48.2
Fig.8 e) ((No blur))	256	0.1	0.29	1920×1080	25.5
Fig.8 f)	16	0.1	0.29	1920×1080	52.6
Fig.8 h)	64	0.1	0.29	1920×1080	46.4
Fig.8 j)	256	0.1	0.29	1920×1080	24.3
Fig.9	64	0.1	0.33	1920×1080	45.8
Fig.10	64	0, 0.1, 0.3, 0.5	0.33	1920×1080	46.2
Fig.11 a) (Reference [17])	-	-	-	1500×1000	<0.003
Fig.11 b)	64	0.1	0.33	1920×1080	43.6
Fig.12 a) (Reference [16])	-	-	-	-	<30
Fig.12 b)	128	0.3	0.65	1280×720	76

V. CONCLUSION

In this paper, a real-time simulation method is proposed for light scattering effects in the atmosphere. By utilizing a physically-based atmospheric model that incorporates both Rayleigh and Mie scattering mechanisms, the total radiance equation is simplified and discretized, to construct a multiple scattering model. In addition, based on different light source characteristics, the method employs accurate ray marching path lengths, jittered sampling, and exclusion of shadowed region samples for lighting computations. This significantly reduces the required number of samples, resulting in a noteworthy enhancement in rendering efficiency, while maintaining stable rendering quality. Rendering is accomplished by utilizing an improved ray marching algorithm to achieve a more accurate simulation of scattering effects. In terms of simulating ambient light and sunlight, this paper optimized the gradient effects under different time and weather conditions. It blends the fog color with scene fragments during rendering, accurately presenting the various scattering properties of light in the atmosphere. In summary, the proposed method has demonstrated excellent performance in both real-time processing and realism, achieving a good balance between computational efficiency and visual effects. It provides valuable reference for further research and applications in the field of light scattering simulation, effectively enhancing the realism and three-dimensional sense of the scene while meeting real-time requirements.

Although our method can achieve real-time rendering in most scenarios, rendering efficiency can still be significantly impacted in cases involving large-scale scenes or high sampling counts. Future research could explore more efficient computational optimization methods to reduce the computational overhead. Meanwhile, the application of blur effects might result in some loss of detail. Efforts can be directed towards finding improved methods that allow for preserving the intricate details of simulation results to their maximum while still maintaining real-time rendering.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China under Grant No. 61902340.

REFERENCES

[1] Song G, Pan W-J. (2021) "Real-Time Rendering Algorithm of Aerial Scene Based on Atmospheric Scattering Model," Computer Simulation 38(8), 43-47+322.

[2] Bauer F. (2019) "Creating the Atmospheric World of Red Dead Redemption 2," SIGGRAPH 2019 course, 1-79.

[3] Li J, Carlson B E, Yung Y L, et al. (2022) "Scattering and absorbing aerosols in the climate system," Nature Reviews Earth & Environment 3(6), 363-379.

[4] Wang R, Hua W, Huo Y, et al. (2022). "Real-time Rendering and Editing of Scattering Effects for Translucent Objects," ArXiv, abs/2203.12339, 1-10.

[5] Nishita T, Sirai T, Tadamura K, et al. (1993) Display of the earth taking into account atmospheric scattering. Proceedings of the international conference on computer graphics and interactive techniques. NewYork:ACM Press, 175-182.

[6] Jackel D, Walter B. (1997) "Modeling and Rendering of the Atmosphere Using Mie-Scattering," Computer Graphics Forum 16(4), 201-210.

[7] Kuz'Min V. L, Val'Kov A. Y, Zubkov, L. A. (2019). "Photon diffusion in random media and anisotropy of scattering in the henyey-greenstein and rayleigh-gans models," Journal of Experimental and Theoretical Physics (3), 128.

[8] Su G. Y, Huang Q, Sun C. J. (2023). "A study on light extinction model and inversion of mixed particle system based on monte carlo method.," Powder Technology: An International Journal on the Science and Technology of Wet and Dry Particulate Systems, 430.

[9] Kobrtek J, Milet T , Michal T, et al. (2022). "Comparison of modern omnidirectional precise shadowing techniques versus ray tracing," Computer Graphics Forum: Journal of the European Association for Computer Graphics (1), 41.

[10] Hillaire S. (2020) "A Scalable and Production Ready Sky and Atmosphere Rendering Technique," Computer Graphics Forum 39(4), 13-22.

[11] Czerninski I, Schechner Y. Y. (2021). Accelerating Inverse Rendering By Using a GPU and Reuse of Light Paths. ArXiv, abs/2110.00085, 1-31.

[12] Huo Y, Wang R, Hu T, et al. (2016) "Adaptive matrix column sampling and completion for rendering participating media," ACM Transactions on Graphics 35(6), 1-11.

[13] West R, Georgiev I, Gruson A, et al. (2020) "Continuous multiple importance sampling," ACM Transactions on Graphics 39(4), 1-12.

[14] Szirmaykalos L, Magdics M, Sbert M. (2018) "Multiple Scattering in Inhomogeneous Participating Media Using Rao-Blackwellization and Control Variates," Enfermería Intensiva 24(1), 12-22.

[15] Vibert N, Gruson A, Stokholm H, et al. (2019) "Scalable Virtual Ray Lights Rendering for Participating Media," Computer Graphics Forum 38(4), 57-65.

[16] Deng X, Jiao S, Bitterli B, et al. (2019) "Photon surfaces for robust, unbiased volumetric density estimation," ACM Transactions on Graphics 38(4), 1-12.

[17] Wilkie A, Vevoda P, Bashford-Rogers T, et al. (2021) "A fitted radiance and attenuation model for realistic atmospheres," ACM Transactions on Graphics 40(4), 1-14.

[18] Kettunen M, D'Eon E, Pantaleoni J, et al. (2021) "An unbiased ray-marching transmittance estimator," ACM Transactions on Graphics 40(4), 1-20.

[19] Korkin S, Yang E-S, Spurr R, et al. (2022) "Numerical results for polarized light scattering in a spherical atmosphere," Journal of Quantitative Spectroscopy and Radiative Transfer 287,108194.