Efficient Real-time Volumetric Cloud Rendering using a Hybrid Sparse Voxel-Noise Cloud Model

JiangTao Zhu*
China Aviation Planning and Design Institute, Beijing, China

ChongWen Wang†
Beijing Institute of Technology, Beijing, China

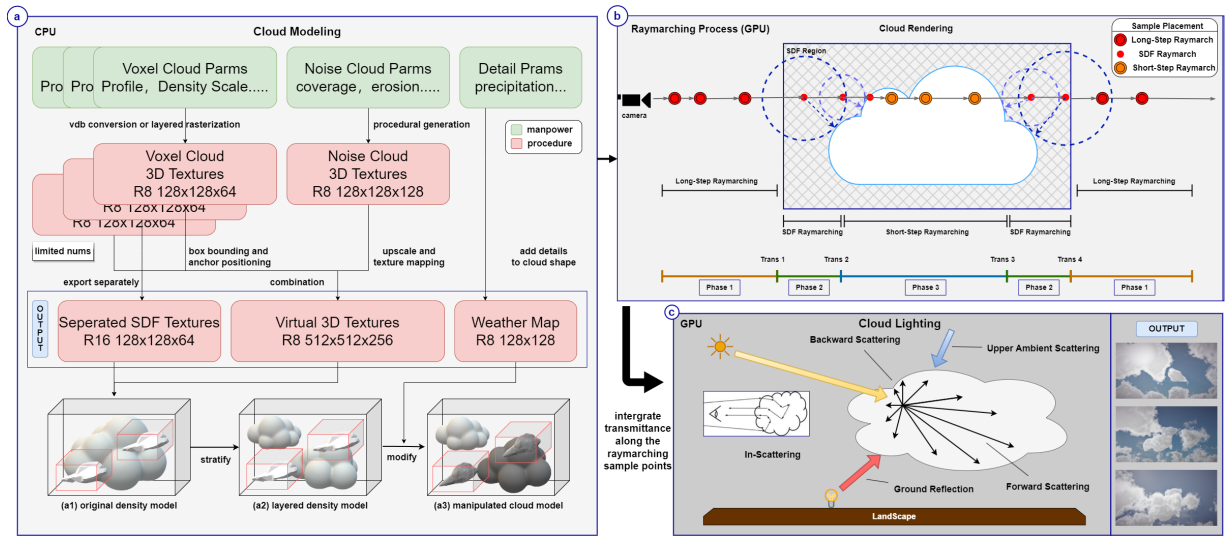

Figure 1: Mixed cloud method overview. (a) The output of cloud modeling is 3D density textures, which are combined as a virtual texture to build the sparse structure. (b) The rendering process incorporates a novel three-phase raymarching algorithm efficiently skipping empty space through the utilization of spatial scalar field. (c) A summation of multiple light effect contributions.

Abstract

Volumetric cloud rendering, crucial for applications such as flight simulators and open-world video games, demands efficient real-time rendering techniques. Current methods either lack accuracy or are computationally intensive. This paper presents a hybrid sparse voxel-noise cloud model that combines high-fidelity voxel clouds in priority regions with less computationally demanding noise clouds elsewhere. We introduce a three-phase raymarching algorithm optimized for this structure, achieving a balance between realism and performance. Experimental results show that our approach reduces rendering time by 30% compared to separate rendering of voxel and noise clouds, while enhancing visual fidelity. This work offers a practical solution for real-time volumetric cloud rendering, advancing the state of the art in this domain. The code we used to evaluate our model is available at https://gitee.com/jett-rot/Project-VolumetricCloudRendering

Keywords: Volumetric Rendering, Raymarching, Noise Clouds, Voxel Clouds, Signed Distance Field

1 Introduction

Real-time rendering of dynamically realistic clouds holds significant value for applications characterized by outdoor environments. For instance, in open-world video games, the lighting condition changes day and night, necessitating a dynamic weather system for the natural evolution of cloudscapes. Players are allowed to ascend from the ground and seamlessly soar through the sky. The large-scale and dynamic features of clouds pose significant challenges in cloud rendering. However, as Goswami and others [6, 12] have surveyed, cloud models based on physical simulations—such as particle-based methods, hybrid physics-driven procedural techniques, and approaches incorporating sounding data—require solving the complex Navier-Stokes fluid dynamics equations. Although these methods can produce highly realistic visual results, achieving real-time rendering with such models on modern consumer-grade GPUs remains infeasible due to their computational intensity, often resulting in frame rates below 30 frames per second. Instead, volumetric rendering, as the most advanced cloud rendering method, is widely adopted in the industry due to its superior performance and feature support compared to other techniques [14].

Currently, noise clouds and voxel clouds stand out as the most prevalent volume rendering techniques, each has their own advantages and challenging limitations. As depicted in Fig. 2(a), In addition to some basic features such as the effects of lighting on clouds throughout the day-night cycle and the projection of shadows on the ground, noise clouds excel in rendering large-scale regions, demonstrating excellent frame rates and low performance overhead. However, they may lack accuracy in close-range rendering from aerial perspectives, limiting their suitability for flight simulations. Since noise cloud models are generated using fractal noise, developers can control their position and size, but it is challenging to present precise cloud shapes.

On the other hand, the underlying scalar field of voxel clouds is responsible for efficient empty space skipping in GPU volume rendering, ensuring model accuracy to obtain better performance. As illustrated in Fig. 2(b)(c), voxel clouds, with their fixed shape and size, lack the flexibility required to realistically reproduce the

*e-mail: zhujt016@avic.com
†e-mail: wcwzzw@bit.edu.cn

complex morphological changes and evolutions of actual cloud formations, making it unsuitable for representing thin, transparent cloud types such as stratus or cirrus. This limitation becomes particularly pronounced in dynamic weather systems and meteorological simulations that demand high levels of realism, thus restricting the ability of voxel clouds to support a full range of cloud types. Furthermore, the substantial memory requirements associated with voxel clouds further constrain their applicability, particularly on low-end devices or mobile terminals.

To address these challenges, this paper presents a novel solution named Mixed Cloud, designed to create an efficient, dynamic, and user-friendly real-time cloud rendering model. The key contributions are as follows:

Integration of Sparse Structures: First, the paper introduces a sparse structure that seamlessly integrates voxel clouds and noise clouds. This integration leverages the strengths of both voxel and noise clouds while mitigating their respective weaknesses, striking a favorable balance between performance and visual quality.

Optimized Lighting Rendering Paradigm: Second, this paper optimizes the traditional lighting rendering paradigm [9, 19] by refining the calculation methods for ambient lighting. Specifically, it simulates the effects of two distinct light sources: the ground reflection and the atmosphere, thus increasing the realism of cloud representation.

Three-Phase Raymarching Algorithm: Third, this paper introduces a three-phase raymarching algorithm specifically designed for sparse structures. The algorithm effectively avoids sampling empty space during raymarching and ensures smooth transitions between different phases. By optimizing the rendering process of model sampling, the algorithm effectively reduces unnecessary sampling and achieves significant performance improvements by optimizing raymarching configurations across different phases.

Implementation and Validation: Lastly, this paper employs a custom Vulkan renderer to replicate traditional methods and implement the newly proposed Mixed Cloud solution. Through extensive experimentation, we conducted performance evaluations and feature comparisons against advanced cloud rendering methods. The results demonstrate the feasibility and superiority of the Mixed Cloud approach in practical applications.

2 Related Works

This section provides an overview of existing tools and previous work in the field of volumetric cloud generation. While these methods have significantly advanced the rendering of realistic cloud formations or performance of implementation, they still exhibit certain drawbacks.

Research related to volumetric cloud rendering can be categorized into three primary domains: cloud modeling, cloud rendering, and cloud lighting. The relationship among them is crucial, with cloud modeling serving as the foundation of cloud rendering, significantly influencing the technological trajectory of cloud rendering. Cloud lighting typically works independently, utilizing economical empirical formulas to approximate complicated lighting processes.

Noise Cloud. This model employs fractal noise to simulate the amorphous characteristics of clouds, and cloud shapes are procedurally generated using volumetric implicit functions [4, 5, 17]. In recent years, one of the most successful methods has been the Nubis model proposed by Guerrilla Games, developers of the game "Horizon Zero Dawn". [19–22]. In addition, researchers have proposed some optimization methods. For instance, Rockstar Games introduced An intricate cloud and fog structure that utilizes aligned frustum voxel and Look-Up Table (LUT) to construct cloud shapes [18]. Tencent Magic Cube Studio utilizes a semi-octahedral LUT parameterization of the sky and stores raymarching results in a texture for chessboard updates, bringing volumetric cloud to mobile platforms [30]. In summary, noise clouds leverage high-level knowledge

	(a) Noise Cloud	(b) Low-Res Voxel Cloud	(c) Nubis Cloud
Evolution	Yes	Not Really	Not Really
Time of Day	Yes	Yes	Yes
Lighting	Yes	Yes	Yes
Terrain-Cast Shadows	Yes	Yes	Yes
High Frame-rates	Yes	Yes	Yes
Flight-Capable	NO	Not Good	Yes
Freedom Modeling	NO	Yes	Yes
Memory-Friendly	Yes	Not Good	NO
All Kinds of Cloud	Yes	NO	NO
Realism	Not Good	Not Good	Yes

Figure 2: The feature matrix of modern volumetric rendering methods (a) Traditional noise clouds offer cost-effective solutions, but fail in aerial perspectives and freeform modeling. (b) Low-resolution voxel clouds make a compromise in performance, resulting in subpar visual representation. (c) *Nubis Cloud* stands out as a high-resolution voxel cloud [24], albeit with significant memory overhead.

available in various fields to achieve artistic simulations of fluid clouds, sacrificing quality for improved performance. In our work, a similar noise cloud model is also employed.

Voxel Cloud. The basic idea behind this technique is to employ a complex cloud mesh composed of volumetric particles to depict the shape of clouds. For instance, Bouthors et al. utilized surface mesh volumes and super-textures to enrich cloud shape and contour [1, 29]. A recent significant advancement is the integration of voxels, point clouds, and meshes through fluid simulation for cloud generation by Schneider et al. [24]. Compared to the well-established noise clouds, voxel clouds demonstrate increased realism and excellent in raymarching efficiency. However, memory constraints pose an inevitable challenge for voxel clouds. Current voxel cloud models are almost static, and not suitable for thin stratus or wispy cirrus cloud.

Raymarching. A well-known method for rendering volumetric clouds is raymarching based on density sampling [9, 10, 18, 24, 30]. Each ray samples and integrates opacity and color information for every unit it traverses. To further enhance visualization quality and perception, we trace arbitrary rays to apply visual effects such as ambient occlusion, shadows, and shading. These effects often necessitate secondary raymarching, which incurs significant computational overhead. Additionally, Signed Distance Fields (SDFs) can be employed as acceleration kernels to improve the efficiency of raymarching [13, 23]. SDFs can also be used to compute soft ground projection shadow [27], making voxel clouds a cost-effective choice.

Artificial Intelligence. Compared to traditional rasterization and ray-tracing methods, artificial intelligence has been explored to generate photorealistic cloud images. Kallweit's algorithm [15], utilizing the Radiation Prediction Neural Network (RPNN), effectively simulates realistic cloud color accumulation. However, the model's training demands significant computational resources, including an NVIDIA Titan (Pascal) GPU and two Intel Xeon CPUs, taking approximately 12 hours—rendering it impractical for real-time applications. Similarly, the latest video-based generative models, namely *Sora* [2], can produce highly realistic cloud animations. Despite their impressive results, these models still encounter challenges beyond rendering time and computational cost. According to the technical

report [3], *Sora* struggles with spatial accuracy and simulating complex physical phenomena, such as cloud lightning and the natural evolution of clouds under wind dynamics.

Next section describes our approach and is divided into three areas (modeling, rendering, lighting) where we present methods used or considered by us.

3 Methods Overview

In this section, a novel volumetric rendering approach for clouds named as "Mixed Cloud" is introduced, aiming to create an efficient, dynamic, and user-friendly real-time cloud model. By leveraging a sparse structure formed by both noise clouds and voxel clouds, it seeks to merge the strengths of these two methods while striking an optimal balance between realism and performance. The diagram in Figure 1 delineates the three primary components of "Mixed Cloud".

In cloud modeling, the origin density model of the clouds is generated using fractal noise [21, 22], and the voxel texture can be obtained from layered rasterization or fluid simulation [24,25]. Since the resolutions of those output textures differ, normalization using standard bounding boxes is necessary. They are then merged into a high-resolution virtual texture, followed by procedural modifications using implicit mathematical expressions (see Section 4).

In cloud rendering, it primarily deals with how cloud model is transformed into pixels output to the screen. In this paper, a new raymarching algorithm, optimized for the sparse structure, is employed to correctly and effectively render the sky (see Section 5).

In cloud lighting, besides the traditional transmittance attenuation prototype [8, 21], we have also developed a new method for ambient scattering, representing the color variation between upper and lower layers of the sky (see Section 6).

4 Cloud Modeling

This section describes how cloud data is generated and stored in computers and how it is organized to form the structure of clouds.

4.1 Constructing Sparse Structure

As depicted in Fig. 1(a), the density model of clouds origin from 3D textures generated by both noise clouds and voxel clouds. Noise clouds offer a cost-effective solution for generating expansive cloudscapes. On the other hand, voxel clouds provide a more precise representation, ideal for depicting clouds with defined shapes and volumes. The scalar field data from the voxel cloud texture is separated because it will be frequently used in raymarching.

Noise Generation. Randomized fractal clouds can be simulated using Fractional Brownian Motion (FBM) noise, as proposed by Voss [28]. Among the various noise, Perlin-Worley noise stands out as a prevalent choice [5, 17]. Perlin noise is adept at reproducing the fog-like effects observed in cloud layers, while Worley noise effectively simulates the cauliflower-like billowing patterns typical of cumulus clouds. Perlin-Worley noise amalgamates the strengths of both types, showing smooth continuity alongside distinct cellularity.

Model Voxelization. The most challenging aspect of voxel cloud creation is the construction and voxelization of the model. A straightforward approach involves layer-by-layer rasterization scanning of the model for voxelization [25], followed by the use of alligator noise erosion to add cloud layer details. Another method involves utilizing Houdini to perform fluid simulations using voxel models and point cloud data to develop cloud models, which results in more vivid and natural shapes.

Then, the output of noise clouds and voxel clouds will be merged into a virtual texture using a hierarchical spares structure. Scene updates dynamically as the camera moves, with the GPU only updating the regions corresponding to objects that need refreshing. The high-resolution Virtual Texture does not physically exist in memory but is logically constructed from multiple small size textures. In this way, noise clouds and voxel clouds can be uniformly handled.

4.2 Cloud Stratification

Clouds can be roughly categorized into three types based on basic physical characteristics: Stratus, Cumulus, and Cirrus clouds [11]. However, this classification differs significantly from realistic clouds, which exhibits a variety of transitional forms due to differences in density distribution with altitude (see Fig. 3). Our primary targets are stratus and cumulus, typically found between 1.5 to 4 kilometers above ground level, as well as cumulonimbus spanning the atmosphere. Given their inherently thin nature, high-altitude cirrus clouds lend themselves to a more cost-effective method, notably through the integration of 2.5D textures into our system, thereby achieving a more natural representation of cloud layers.

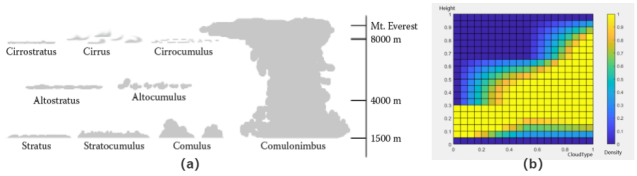

Figure 3: (a) illustrates various types of clouds in the sky along with their height ranges. (b) gives a more intuitive presentation about how the density-height gradient field remaps the height and cloud type sampled from the weather map to the cloud density.

4.3 Adjusting Weather Parameters

To manipulate the desired cloudscapes, several parameters are encapsulated in a texture called the weather map, where *cloud coverage* controls the expected cloud density at the very point; *precipitation* describes the probability of rainfall, consequently impeding light propagation; *cloud type* ranges from 0 to 1, with their physical meanings interpretable as cloud shape. For instance, in mountainous or lake/river regions, it is advisable to increase the value of *cloud coverage* and *precipitation* and change *cloud type* to alto clouds, which better adhere to natural patterns.

5 Cloud Rendering

Given the adoption of a novel sparse structure mentioned in sec.4.1, corresponding adaptations are necessary in the raymarching algorithm. In this section, a new rendering algorithm termed three-phase raymarching is utilized to address it.

As depicted in Figure. 4 and algorithm 1, the core of this algorithm is a state machine. According to the cloud model described in Section 4.1, the rendering region is divided into three regions. The algorithm determines the current phase based on the scene data of sampling points, choosing a suitable raymarching algorithm to decide the step length, quickly skipping through blank areas in the sky.

Another key component is the transition processing during state switching. For example, when the ray reaches *Trans 1*, the visual fidelity and dynamic transition between noise and voxel clouds are crucial to prevent visual artifacts caused by low-resolution noise clouds. Near the SDF region and when the field of view includes the SDF region, cloud density is gradually blended based on distance from camera. Simultaneously, the frequency of high-frequency detail noise in voxel clouds is increased to sharpen the surface and highlight the shape of clouds. When the ray enters the SDF region, the raymarching algorithm needs to be switched to SDF Raymarching, and it is important to note that when the sampling point is adjacent to the SDF bounding box, edge-stopping may occur, requiring manual adjustment of the step length to prevent it. Likewise, the opposite *Trans 4* applies.

Besides, when the ray reaches *Trans 2* and *Trans 3*, there may be two situations: the sampling point may be located in a cavity

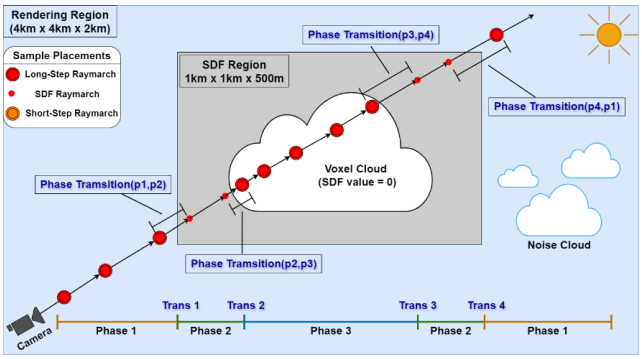

Figure 4: The structural details of the three-phase raymarching algorithm for ground perspectives are outlined below. All potential sample points are marked, with phase transitions occurring at varying distances along the ray path. This enables a smooth transition between different phases based on the spatial information, ensuring optimized sampling and accurate rendering.

within the cloud or just inside the cloud. This may lead to frequent algorithm switching, a phenomenon we called *phase jitter*. To address this issue, a hit counter was introduced to cache missed counts. If the sampling point fails to enter the interior of the voxel cloud multiple times, phase transition occurs. If the sampling point re-enters the voxel cloud frequently, it counts the miss instead of switching.

Finally, it has been observed that rays take longer to exit the SDF region than to enter it during phase transitions. To improve algorithm efficiency, it is recommended to position the SDF region within low-altitude cloud layer, which presents higher cloud density, thereby increasing the effective coverage of the SDF region.

Algorithm 1 three-phase Raymarching Algorithm

procedure RAYMARCH(ray r, scene s, phase p)
  $t \leftarrow 0, misses \leftarrow 0$
  while $t < t_{max}$ **do**
    $raypos \leftarrow r.origin + t \cdot r.direction$
    $density \leftarrow CloudTest(raypos)$
    if $distance(p,s) < CAMERANEAR$ **then**
      BlendAndUpscale $(density, dist)$
    end if
    if $density > \varepsilon$ **then** ▷ ε is a small threshold
      $R \leftarrow R + Lighting(density, dist)$
    else ▷ count the miss and adjust stepsize
      if $misses > 6$ **then**
        $stepSize \leftarrow PhaseTransistion(p,s)$
      end if
      $t \leftarrow t + stepSize$
    end if
  end while ▷ accumulate the color and alpha
  return (R, α)
end procedure

6 Cloud Lighting

Cloud lighting is a well-researched field that describes how light propagates through clouds, affecting their appearance. In this paper, we extend the traditional raymarching rendering equation [9, 19] to calculate both direct and indirect scattering from sunlight using Equation 1. Additionally, we incorporate the ambient lighting term using the Preetham sky model.

$$L(x,w) = \int_0^D e^{-\tau(x,x')} \sigma_{is}(x') [p_{sun}(w,w_{sun}) L_{sun}(x', w_{sun}) + p_{amb} L_{amb}] dx' \quad (1)$$

Where $\sigma_i s$ represents the scattering coefficient, p_{sun} is the phase function, and L_{sun} denotes the radiance of sunlight coming from the direction w_{sun}. We utilize a traditional energy attenuation method to prevent overexposure caused by sunlight in the sky. Energy starts at 1 and is multiplied by various attenuation factors approaching 0. The scattering of sunlight into clouds is primarily influenced by several factors below:

Extinction $e^{-\tau}$. Beer's law is often employed to simulate the attenuation of light passing through clouds, representing a crucial determinant of cloud color and directional illumination. Beer-Lambert is a modification of Beer's law that takes multiple scattering into consideration, setting a minimum brightness at the bottom of clouds.

Phase Function p_{sun}. The Henyey-Greenstein phase function is often employed to compute the amount of scattering of sunlight entering and exiting a medium. Besides, a mixture of Henyey-Greenstein and an approximate backward scattering function is typically used to achieve better sliver edge effects.

Inscattering σ_{In}. Inscattering is a sight-dependent effect used to simulate light refraction within clouds, with a probability of ultimately rotating 180 degrees and emitting from the gaps in the opposite direction. This effect is less pronounced at higher altitudes due to plenty of sunlight.

The newly introduced ambient scattering is height-dependent, influenced by both ground-reflected light and upper sky scattering.

$$L_+(x) = \int_{\Omega_{2\pi+}} p_{iso} L_{amb_+} e^{-\sigma_t \frac{H_+}{n_+ \omega}} d\omega \quad (2)$$

Here, $L_+(x)$ is the top radiance, $\Omega_{2\pi+}$ denotes all possible directions of the top hemisphere, $p_{iso} = 1/4\pi$ is our assumed isotropic phase function, n+ is the local normal to the volumes. Similarly, for the bottom ground reflection term, we can use the same equation with '-' instead of '+'. It must be clarified that this lighting model relies on high-level empirical paradigms rather than precise physical calculations, thereby significantly reducing computational costs and ensuring real-time performance.

7 Results and Evaluation

All experiments were performed on a Windows 10 system with an NVIDIA GTX 3060 (laptop) graphics card. A custom renderer developed by Vulkan was employed for the work presented in this paper. The shading language used is GLSL. The rendering resolution for all experiments is 1920x1080, with each frame rendered at one-fourth resolution after reprojection. Nvidia's official graphics analysis tool, Nsight, is used for testing and analyzing GPU rendering times and texture memory usage submitted to the GPU. Rendering time was measured as the average time taken during the raymarching process in computer shader.

7.1 Render Target and Experiment Parameters

To accurately test the performance of volumetric cloud rendering methods and eliminate interference from unrelated variables, this study selected a clear summer noon as the atmospheric simulation scene. This choice provides a typical atmospheric environment and removes additional atmospheric factors, ensuring the validity and comparability of the experiment's results.

Once the experimental scene was determined, cumulus clouds in the lower atmosphere were chosen as the rendering target for all experiments. Compared to thin, sheet-like stratus clouds or mixed-state stratocumulus clouds, cumulus clouds have distinct volume and shape. This makes it easier to control the size of the rendering target in both noise cloud and voxel cloud models, ensuring consistency of variables in the experiment. Furthermore, cumulus clouds, being one

of the most common cloud types, have general representativeness. Unlike other higher-altitude cloud types, lower-altitude cumulus clouds are not influenced by other clouds during the raymarching process. This allows for more accurate measurement of the time required to render a single cloud, providing a more reliable data foundation for performance evaluation.

For all experiments, the parameters of the raymarching algorithm will remain consistent, following recommended values from the literature [7] to ensure comparability and representativeness. The long-step size is set to 0.05 times the atmospheric thickness, with the short-step size being one-third of the long-step. The maximum marching intervals are fixed at 128 steps, and the number of sun-sampling iterations during sub-raymarching is kept constant at six.

7.2 Data Sets

All noise textures and voxel data used in the experiments were generated using the Houdini asset creation tool provided by Schneider et al. [21, 24].

Noise Cloud	Low-Res Voxel Cloud	High-Res Voxel Cloud	Mixed Cloud
L×W×H (km): 4x4x2 Precision (m): ~32	L×W×H (km): 4x4x2 Precision (m): ~16	L×W×H (km): 1x1x0.5 Precision (m): ~8	L×W×H (km): 4x4x2 Precision (m): 8~32
Low-Freq Noise Texture 128x128x128 4 channels RGBA8	N/A	N/A	Low-Freq Noise Texture 128x128x128 4 channels RGBA8
High-Feq Noise Texture 32x32x32 3 channels RGB8	N/A	N/A	High-Feq Noise Texture 32x32x32 3 channels RGB8
N/A	Voxel Modeling Texture 256x256x128 3 channels RGB8	Voxel Modeling Texture 128x128x64 ^2 3 channels RGB8	Voxel Modeling Texture 128x128x64 ^2 3 channels RGB8
N/A	SDF Texture 256x256x128 1 channels R16	SDF Texture 128x128x64 ^2 1 channels R16	SDF Texture 128x128x64 ^2 1 channels R16
Other 2D Textures	Other 2D Textures	Other 2D Textures	Other 2D Textures

Figure 5: Different datasets used in the comparsion experiments and ablation experiments below. Specifically, two voxel textures are used in the hybrid spares structure of Mixed Cloud.

Since the accuracy of the cloud model is determined by the rendering range and texture size, the length ratio represented by each pixel can be approximated by dividing the side length of the rendering distance by the texture size. Thus, with the consistent rendering range, using larger textures will result in higher precision. As shown in Fig. 5, the noise cloud model primarily uses a low-resolution Perlin-Worley noise 3D texture to shape the basic shape of the clouds, and a low-resolution high-frequency curl noise texture to add cloud details. The cloud shapes are defined using a 128x128x128 low-resolution 3D texture, yielding a texture accuracy of approximately 32 meters.

The voxel cloud model utilizes datasets categorized into high-resolution and low-resolution voxel clouds based on different rendering ranges and texture sizes. The low-resolution voxel cloud employs a 256x256x128 3D texture to render a rectangular area with a resolution of about 16 meters within the same range. In contrast, the high-resolution voxel cloud uses two 128x128x64 3D textures to render two original quarter-sized regions, achieving a resolution of approximately 8 meters. Both voxel cloud models store the SDF (Signed Distance Field) in separate 3D textures with a higher 16-bit precision to facilitate multiple sampling during raymarching.

The mixed cloud model's dataset consists of both noise clouds and high-resolution voxel clouds, as described in Section 4.1. It represents rectangular areas with dynamic precision within the same rendering range using a sparse structure. Additionally, all datasets include various 2D textures such as weather maps, flow maps, and cirrus layer shape maps to control and adjust the volumetric cloud system.

7.3 Comparison with Different Algorithms

By comparing the performance of different algorithms applied to the same data set as shown in Table 1 (a)(b) and (c)(d), it can be observed that when applied to the same rendering target using different datasets, in all cases, voxel clouds consistently exhibit higher performance advantages over noise clouds. This disparity can be attributed to the SDF acceleration structure within voxel clouds, which effectively enhances the efficiency of the raymarching algorithm and facilitates the implementation of soft shadow effects. This optimization strategy exemplifies the trade-off between space and time.

On the other hand, by comparing (a) and (c), as well as (b) and (d) in Table 1, no significant performance differences were observed when using the same dataset. This indicates that the novel three-phase raymarching algorithm does not introduce additional performance overhead and maintains the same level of efficiency as the traditional raymarching algorithm.

Additionally, the three-phase raymarching algorithm offers several extra advantages. It is capable of being applied to both noise cloud models and voxel cloud models while utilizing sparse structures to accelerate the rendering process. By incrementally increasing the number of voxel textures, the algorithm's efficiency can be balanced. Furthermore, in terms of visual effects, the three-phase raymarching algorithm performs distance-based frequency sharpening on the cloud surface noise, resulting in a more three-dimensional and enriched appearance compared to traditional cloud models.

Table 1: The experiments compared various raymarching algorithms under different datasets, with consistent algorithmic parameters.

NO.	Algorithm	Data Set (Resolution)	Time (ms)	Output
(a)	Normal Raymarch	Noise Cloud 128×128×128	2.47	Fig. 6(a)
(b)	SDF Raymarch	Low-Res Voxel Cloud 256×256×128	1.51	Fig. 6(b)
(c)	three-phase Raymarch	Noise Cloud 128×128×128	**2.36**	Fig. 6(c)
(d)	three-phase Raymarch	Low-Res Voxel Cloud 256×256×128	**1.61**	Fig. 6(d)

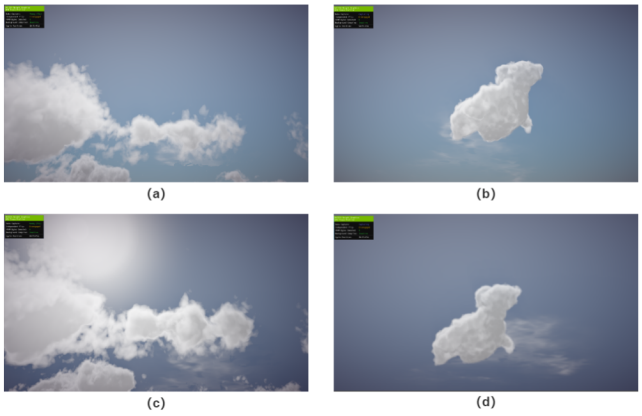

Figure 6: The output images of the raymarching algorithm comparison experiments.

7.4 Ablation Experiment with Cloud Models

By comparing (a), (b), and (c) in Table 2, we can observe the performance of the three-phase raymarching algorithm on the same rendering target when using different datasets. Notably, when using separate noise cloud and high-resolution voxel cloud datasets,

the mixed cloud method integrates them into a virtual texture for cloud modeling. As a result, the corresponding remaining parts of the sparse structure are stored as zero-density blank areas in the same-sized virtual texture to ensure dataset consistency.

For the same rendering target, the three-phase raymarching algorithm demonstrates better performance with the voxel cloud model compared to the noise cloud model. This highlights the importance of introducing voxel clouds to reduce computational overhead. Although the rendering time increases compared to the full-range low-resolution voxel cloud used in Table.2 (d), this is because the sparse structure uses a small-range high-resolution voxel area to balance performance consumption and realistic representation.

Furthermore, it is noteworthy that the rendering time with the mixed cloud dataset is significantly lower than the total time consumed by rendering the two cloud types separately in two passes, as illustrated in Figure.7 (c) and (d), resulting in approximately a 30% performance improvement. This performance profit becomes even more pronounced in cloud-heavy scenes, such as cloudy weather, where it can save at least 10ms in practical scenario. This demonstrates that the mixed cloud solution effectively integrates both voxel and noise cloud models, enabling a seamless transition between the two. This integration significantly reduces the overdraw phenomenon that typically occurs with multi-pass rendering.

Table 2: The experiments compared different datasets under three-phase raymarching algorithm, with consistent algorithmic parameters.

NO.	Algorithm	Data Set (Resolution)	Time (ms)	Output
(a)	three-phase Raymarch	Noise Cloud 128×128×128	2.36	Fig. 7(a)
(b)	three-phase Raymarch	High-Res Voxel Cloud 128×128×64 *2	1.85	Fig. 7(b)
(c)	Normal Raymarch	High-Res Voxel Cloud + Noise Cloud (Two Pass)	**4.12**	Fig. 7(c)
(d)	three-phase Raymarch	Mixed Cloud 128×128×128	**2.89**	Fig. 7(d)

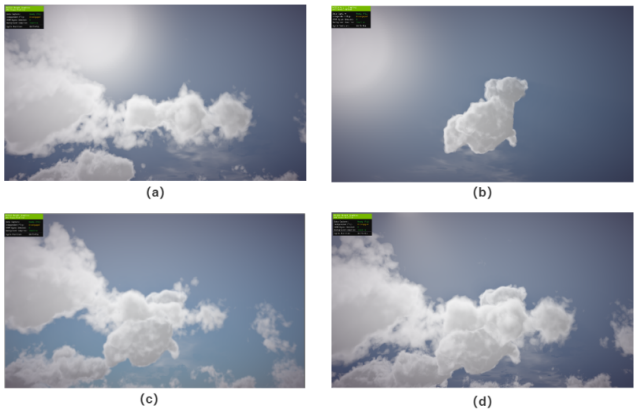

Figure 7: The output images of the ablation experiments of different cloud models.

7.5 Comparison to the State of the Art

We evaluate our novel method by comparing it to the current methods. Variations in texture resolutions employed in cloud modeling significantly influence rendering range and accuracy. Typically, to support aerial perspectives, voxel clouds require 8 meters per pixel to prevent linear artifacts. As shown in Table 3, even without considering uncompressed texture size released in GPU, the compressed texture size in CPU used by low-precision voxel cloud methods is more than five times that of normal noise methods within the same range. Moreover, high-precision voxel clouds, Nubis, exhibit even greater disparity, as the rendering range is reduced by a factor of four. High memory usage poses a critical challenge for voxel clouds in the industry, and the misuse of voxel clouds will lead to severe performance bottlenecks in medium and low-end equipment.

Higher compression rates imply lower precision, making more efficient compression algorithms than BC4 (compression ratio of 3:1) unusable. For voxel clouds employing SDF Raymarching, the reduction in values uncompressed from SDF textures leads to more steps during raymarching, while higher values may cause cloud shape distortion. Confronted with these challenges, the mixed cloud method allocates limited voxel regions within the sparse structure. Consequently, it manages to keep texture memory within 10Mb without sacrificing precision.

Table 3: The comparative analysis of advanced methods includes the Nubis Cloud data from Schneider et al.'s conference reports [21, 24].

Method	Nubis Noise Cloud	Low-Res Voxel Cloud	Nubis Voxel Cloud	**Mixed Cloud**
Noise cloud Textures (Mb)	8.12 128x128x128	N/A	N/A	**8.12 128x128x128**
Voxel cloud Textures (Mb)	N/A	41.1 256x256x128	79.3 512x512x64	**5 *2 128x128x64 *2**
Render Range (Km)	4×4×2	4×4×2	*4×4×0.5*	**4×4×2**
Precision (m/pixel)	32	16	8	**8 ~ 32**
Compressed Size (Mb)	2.67	13.7	25.116	**2.67+1.67*2**

8 Conclusion

we introduced a novel sparse structure cloud model that integrates noise clouds and voxel clouds, and proposed a three-phase raymarching algorithm corresponding to this structure. Our initial exploration focused on low-precision voxel cloud solutions, where the key challenge was the visual artifacts and lighting discrepancies caused by inaccurate lighting calculations. Besides, high memory usage makes voxel clouds impractical for low-end devices, which is why voxel clouds are rarely used in industry.

However, our method also has its limitations. Overlapping voxel cloud regions not only increase memory costs but also causes raymarching to step over areas incorrectly, resulting in wrong shapes. Although adding conditional checks and iterative fallbacks can address the issues, the computational cost is significantly increased. Therefore, it is highly recommend to avoid overlapping voxel regions.

In future work, we aim to further partition the sparse space with intricate structures (such asK-D trees, Octrees [16, 26]) for inside-outside intersection testing and global illumination. By discarding rays that never hit the boundaries of the rectangular prism containing the cloud, and avoiding iterations over sdfbox outside the camera view frustum, empty space can be skipped efficiently. However, dynamic transform in noise clouds may lead to variations in empty and non-empty areas, making it challenging to implement precomputed spatial partitions.

Declarations

Conflict of interest The authors declare no conflict of interest.
Funding This work was supported by National Key Research and Development Program under 2022YFB3305400.

References

[1] A. Bouthors, F. Neyret, N. Max, E. Bruneton, and C. Crassin. Interactive multiple anisotropic scattering in clouds. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, I3D '08, p. 173–182. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1342250.1342277

[2] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh. Video generation models as world simulators. 2024.
[3] J. Cho, F. D. Puspitasari, S. Zheng, J. Zheng, L.-H. Lee, T.-H. Kim, C. S. Hong, and C. Zhang. Sora as an agi world model? a complete survey on text-to-video generation. *arXiv preprint arXiv:2403.05131*, 2024.
[4] D. S. Ebert. Volumetric modeling with implicit functions: a cloud is born. In *ACM SIGGRAPH 97 Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH '97*, SIGGRAPH '97, p. 147. Association for Computing Machinery, New York, NY, USA, 1997. doi: 10.1145/259081.259233
[5] D. S. Ebert. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.
[6] P. Goswami. A survey of modeling, rendering and animation of clouds in computer graphics. *The Visual Computer*, 37:1931–1948, 2021. doi: 10.1007/s00371-020-01953-y
[7] F. Häggström. Real-time rendering of volumetric clouds, 2018.
[8] S. Hill and E. Heitz. Real-time area lighting: A journey from research to production. *ACM SIGGRAPH Courses*, 2016.
[9] S. Hill, S. McAuley, L. Belcour, W. Earl, N. Harrysson, S. Hillaire, N. Hoffman, L. Kerley, J. Patry, R. Pieké, et al. Physically based shading in theory and practice. In *ACM SIGGRAPH 2020 Courses*, pp. 1–12. Association for Computing Machinery, 2020.
[10] S. Hillaire. Physically based sky, atmosphere and cloud rendering in frostbite. In *ACM SIGGRAPH*, pp. 1–62. Association for Computing Machinery, New York, NY, USA, 2016.
[11] L. Howard. *On the modifications of clouds*. Number 3. Asher, 1894.
[12] B. Huang, J. Chen, W. Wan, C. Bin, and J. Yao. Study and implement about rendering of clouds in flight simulation. In *2008 International Conference on Audio, Language and Image Processing*, pp. 1250–1254, 2008. doi: 10.1109/ICALIP.2008.4590228
[13] Y. Jiang, D. Ji, Z. Han, and M. Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization, 2022.
[14] C. Jiménez de Parga and S. R. Gómez Palomo. Efficient algorithms for real-time gpu volumetric cloud rendering with enhanced geometry. *Symmetry*, 10(4), 2018. doi: 10.3390/sym10040125
[15] S. Kallweit, T. Müller, B. Mcwilliams, M. Gross, and J. Novák. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.
[16] N. Max. Global illumination in sparse voxel octrees. *The Visual Computer*, 38:1443–1456, 2022. doi: 10.1007/s00371-021-02078-6
[17] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.
[18] F. B. (Rockstar). Creating the atmospheric world of red dead redemption 2: A complete and integrated solution. In *ACM SIGGRAPH 2019 Courses*, SIGGRAPH '19. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3305366.3335036
[19] A. Schneider. The real-time volumetric cloudscapes of horizon: Zero dawn. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2776880.2787701
[20] A. Schneider. Real-time volumetric cloudscapes. In *GPU Pro 7:Real Time Volumetric Cloudscapes*, pp. 97–128. AK Peters/CRC Press, 2016.
[21] A. Schneider. Nubis: Authoring real-time volumetric cloudscapes with the decima engine. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3084873.3096476
[22] A. Schneider. Nubis: Real-time volumetric cloudscapes in a nutshell. In *Eurographics*. Eurographics Association, The Netherlands, Delft, 2018.
[23] A. Schneider. Nubis, evolved: Real-time volumetric clouds for skies, environments, and vfx. In *ACM SIGGRAPH 2022 Courses*, SIGGRAPH '22. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3532720.3546903
[24] A. Schneider. Nubis3: Methods (and madness) to model and render immersive real-time voxel-based clouds. In *ACM SIGGRAPH 2023 Courses*, SIGGRAPH '23. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3587423
[25] T. Strutz. The distance transform and its computation, 2023.
[26] Ströter, M.-R. Daniel, S. Johannes S., and D. W. ré, Fellner. Olbvh: octree linear bounding volume hierarchy for volumetric meshes. *The Visual Computer*, 36:2327–2340, 2020. doi: 10.1007/s00371-020-01886-6
[27] Y. W. Tan, N. Chua, C. Koh, and A. Bhojan. Rtsdf: Real-time signed distance fields for soft shadow approximation in games. *arXiv preprint arXiv:2210.06160*, 2022.
[28] R. F. Voss. Random fractal forgeries. In *Fundamental Algorithms for Computer Graphics: NATO Advanced Study Institute directed by JE Bresenham, RA Earnshaw, MLV Pitteway*, pp. 805–835. Springer, 1985.
[29] J. Wither, A. Bouthors, and M.-P. Cani. Rapid sketch modeling of clouds. In C. Alvarado and M.-P. Cani, eds., *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*, pp. 113–118. Eurographics Association, Annecy, France, June 2008. doi: 10.2312/SBM/SBM08/113-118
[30] S. Yiming. Tencent Games Developer Summit: Arena Breakout: Creating a Next-gen FPS Game on Mobile, mar 23 2023. [Online; accessed 2024-02-13].

JiangTao Zhu is an industrial design engineer in China Aviation Planning and Design Institute. Hereceived the BS, MS degrees from Beijing Institute of Technology. His research interests include Volumetric Rendering and Digital Twin.

ChongWen Wang is an Associate Professor at the University of Beijing Institute of Technology, China. He was the deputy director of Institute of Digital Performance and Simulation. His research interests include Digital Media Technology, Software Engineering, Artificial Intelligence.