remote sensing

MDPI

Article

OGAIS: OpenGL-Driven GPU Acceleration Methodology for 3D Hyperspectral Image Simulation

Xiangyu Li [1,2], Wenjuan Zhang [2], Bowen Wang [2,*], Huaili Qiu [1], Mengnan Jin [2,3] and Peng Qi [2,4]

[1] School of Physics, Hefei University of Technology, Hefei 230601, China; 2021111191@mail.hfut.edu.cn (X.L.); hlqiu@hfut.edu.cn (H.Q.)
[2] Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China; zhangwj@aircas.ac.cn (W.Z.); jinmengnan23@mails.ucas.ac.cn (M.J.); qipeng24@mails.ucas.ac.cn (P.Q.)
[3] School of Electronic Electrical and Communication Engineering, University of the Chinese Academy of Sciences, Beijing 100049, China
[4] College of Resources and Environment, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: wangbowen@aircas.ac.cn

Abstract: Hyperspectral remote sensing, which can acquire data in both spectral and spatial dimensions, has been widely applied in various fields. However, the available data are limited by factors such as revisit time, imaging width, and weather conditions. Three-dimensional (3D) hyperspectral simulation based on ray tracing can overcome these limitations by enabling physics-based modeling of arbitrary imaging geometries, solar conditions, and atmospheric effects. This type of simulation offers advantages in acquiring multi-angle and multi-condition quantitative results. However, the 3D hyperspectral simulation requires substantial computational resources. With the development of hardware, a graphics processing unit (GPU) offers a potential way to accelerate it. This paper proposes a 3D hyperspectral simulation model based on GPU-accelerated ray tracing, which is realized by modifying and using a common graphics API (OpenGL). Through experiments, we demonstrate that this model enables 600-band hyperspectral simulation with a computational time of just 2.4 times that of RGB simulation. Furthermore, we analyzed the balance between calculation efficiency and accuracy, and carried out a correlation analysis between ray count and accuracy. Additionally, we verified the accuracy of this model by using UAV-based data. The results demonstrate over 90% spectral curve similarity between simulated and UAV-acquired images. Finally, based on this model, we conducted additional simulation experiments under different environmental variables and observation conditions to analyze the model's ability to characterize different situations. The results show that the model effectively captures the effects of environmental variables and observation conditions on the hyperspectral characteristics of vehicles.

Keywords: hyperspectral simulation; ray tracing; OpenGL; algorithm acceleration; verification

check for updates

Academic Editor: Giorgio Antonino Licciardi

Received: 7 April 2025
Revised: 16 May 2025
Accepted: 19 May 2025
Published: 25 May 2025

Citation: Li, X.; Zhang, W.; Wang, B.; Qiu, H.; Jin, M.; Qi, P. OGAIS: OpenGL-Driven GPU Acceleration Methodology for 3D Hyperspectral Image Simulation. *Remote Sens.* **2025**, *17*, 1841. https://doi.org/10.3390/rs17111841

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Hyperspectral imaging, an advanced spectral imaging technology that has undergone constant development since its emergence in the 1980s [1], has become an indispensable tool in remote sensing applications. This technology has made substantial contributions to various earth observation studies, including precision agriculture [2,3], environmental monitoring [4], and geosciences research [5,6]. Recent advancements in sensor hardware and data processing techniques have further expanded hyperspectral imaging technology into the military domain [7]. Hyperspectral sensors can acquire data across hundreds of

consecutive narrow spectral bands, demonstrating exceptional capability in military target detection and identification [8,9].

However, hyperspectral remote sensing data are affected by observation geometry, imaging conditions, spatial resolution and atmospheric variations [10], leading to insufficient data quality under certain complex conditions to fully meet practical application requirements.

To overcome these challenges, hyperspectral imaging simulation technology has emerged as a valuable alternative. By leveraging physical modeling and algorithm optimization, hyperspectral imaging simulation technology overcomes the constraints of real sensor's observation conditions and provides a controllable experimental environment for remote sensing research under various conditions. Compared to real data acquisition, simulation technology can reduce economic and time costs, and be used to support algorithm validation, data augmentation, and sensor design optimization. Currently, there are two main approaches to hyperspectral imaging simulation: two-dimensional (2D) image fusion [11,12] and three-dimensional (3D) scene simulation [13,14].

The 2D image fusion method is based on spaceborne/airborne images, inverting the reflectivity of each pixel in the image, then using the atmospheric radiative transmission to calculate the atmospheric parameters at the simulation time, and finally calculating the radiation at the entrance pupil of the sensor to obtain the spaceborne satellite image under certain imaging conditions. While hyperspectral image generation yields accurate results from existing visible/multispectral data, current data are typically limited to a single viewing angle, constraining comprehensive multi-angle and multi-condition spectral analysis.

The 3D scene simulation method first establishes a 3D geometric model of surface features and assigns optical properties to the 3D model. Then, the radiation calculation is performed according to the given imaging observation conditions. Finally, the digital image is output after the sensor simulation. Ray tracing is the core algorithm of 3D scene simulation. By simulating the physical behavior of radiation, it achieves highly accurate results [15]. The 3D method does not depend on the existing image, and once the 3D scene is constructed, it can simulate the scene under any observation geometry, solar conditions and atmospheric conditions. In addition, it is more interpretable for the physical modeling of radiative transmission. Therefore, the 3D method will be more popular in the pursuit of more quantitative radiation calculation.

The ray tracing simulation places extremely high demands on computing resources, particularly for the hyperspectral simulation which processes thousands of spectral bands and hundreds of millions of rays, therefore, it is necessary to accelerate it. On the one hand, acceleration can be achieved by accelerating operations at the hardware and software levels, and on the other hand, it can be achieved by appropriately reducing the accuracy requirements. Table 1 presents the implementation methods and acceleration techniques of three hyperspectral simulation tools: LESS [16], DART [17], and DIRSIG [14].

Most existing hyperspectral simulation tools are implemented based on CPU and lack GPU acceleration methods. Therefore, the core focus of this paper is to realize GPU-accelerated parallel computing for hyperspectral simulation. With the development of computer technology, the latest GPUs, such as the RTX series from NVIDIA, have special ray tracing support. Given the applicability of this technique, existing studies have achieved GPU-accelerated single-band simulation through APIs such as OpenGL and CUDA. Among these, OpenGL is more widely used because of its good method library support and cross-platform characteristics. For instance, Han et al. aim at the visible light imaging detection of complex space targets at close range, propose the complex target's space-based optical imaging simulation method based on the 3D programming tool OpenGL and 3D target

models, and implement imaging simulation for the TG-02 companion satellite observation mission based on OpenGL [18,19]. Yu et al. conducted an infrared imaging simulation of ship targets [20]. Xin et al. used OpenGL to construct a model of the aircraft target in the airport and its ground background [21]. Jiang et al. used OpenGL to simulate the infrared image of the sea surface [22]. Shen et al. proposed an infrared image generation method based on OpenGL [23]. Kramer et al. developed an algorithm based on OpenGL to calculate the virtual test room including tubular radiators [24]. Hu et al. used OpenGL rendering technology to construct an infrared scene simulation system and obtained a synthetic infrared image from any angle of the 3D scene [25]. In terms of hyperspectral simulation, DIRSIG, a simulation tool developed by the Rochester Institute of Technology (RIT) for generating high-fidelity remote sensing images [14], the LESS model proposed by Qi et al. [16], the DART model proposed by Gastellu-Etchegorry [17] have all identified GPU-accelerated ray tracing as one of the future developmental directions for their respective models. In the calculation method proposed by Cao Y et al., there is no discussion on GPU acceleration [13]. At the same time, Bian pointed out that the computational efficiency of the model based on the GPU computing core will be significantly improved, which will be the future development direction of hyperspectral remote sensing imaging simulation [26].

Table 1. Comparison of Some Hyperspectral Simulation Tools.

Hyperspectral Simulation Tools	Spectral Range	Implementation Methods	Acceleration Techniques
LESS	Unlimited	Based on an open source ray-tracing code named Mitsuba (CPU)	(1) CPU parallel computing (2) Monte Carlo algorithm (3) importance sampling (4) Russian roulette technique
DART	From ultraviolet to thermal infrared wavelengths	CPU, no mention of the computing engine	(1) CPU parallel computing (2) Monte Carlo algorithm (3) importance sampling (4) scene subdivision
DIRSIG	Cover the visible through infrared (0.2–20.0 μm)	Based on the Intel Embree ray-tracing engine (CPU)	(1) CPU parallel computing (2) Monte Carlo algorithm (3) importance sampling

In this article, with algorithm optimization, we reduced computing resource requirements while ensuring accuracy. The number of rays traced is positively correlated with the demand for computing resources. Finding the lowest number of rays while ensuring the accuracy of calculation is conducive to the improvement of computational efficiency. Moreover, we use the bounding volume hierarchy (BVH) algorithm and the low-difference sequence sampling method to improve the intersection efficiency and sampling efficiency.

In addition, the accuracy of simulation is an advantage of 3D simulation, and many studies have analyzed the simulation accuracy. For example, Chen compared the simulation results with the radiance images obtained by the GF-2 satellite [27]. Song compared the measured images of the GF-6 satellite panchromatic band with the simulated images under the same imaging conditions [28]. However, the accuracy verification of hyperspectral simulation models is less seen. This is because it is difficult to obtain the three-dimensional model and reflectivity data required for hyperspectral simulation. Therefore, some verifications are only based on theoretical calculations [13]. Compared with the two verification methods, although theoretical verification can directly verify the accuracy of the model calculation in principle, the measured data verification can better verify the model in the

whole process (especially multiple-scattering radiance) when conditions permit. Therefore, it is necessary to carry out verification based on the measured data.

Based on the above status, we carried out hyperspectral remote sensing simulation imaging research, which realized hyperspectral ray tracing simulation based on OpenGL. We focused on modifying the OpenGL rendering pipeline while accelerating the model through algorithm optimization and adjusting the number of rays, so that a high-precision hyperspectral simulation speed of 1.4 million facets and 350 × 350 pixels reaches the minute level. Subsequently, we validated the model obtained from the above research using UAV-based data. The verification results show that the simulation model can correctly simulate the hyperspectral remote sensing imaging process.

2. Simulation Model

2.1. Radiative Transfer

To construct a 3D hyperspectral simulation model, it is necessary to theoretically establish a radiative transfer chain. The focus is to analyze the radiative transmission process from its source to the sensor, and to derive the complete transfer mechanism through the sun-atmosphere-ground-atmosphere-satellite path.

As shown in Figure 1, the radiation received by a single sensor pixel can be categorized into three components: ground-reflected radiation (L_{zero}, radiation path 1 in Figure 1), path radiation (L_p, radiation path 2 in Figure 1) and adjacency effect radiance (L_s, radiation path 3 in Figure 1). Among these, ground-reflected radiation contributes the most, it includes surface-reflected direct solar radiation (radiation path 1.1 in Figure 1), surface-reflected diffuse solar radiation (radiation path 1.2 in Figure 1), and surface-reflected background-reflected radiation from surrounding objects (radiation path 1.3 in Figure 1), and it undergoes the atmospheric transmission process before reaching the sensor. Path radiation refers to solar radiation that is scattered by the atmosphere before reaching the ground and then enters the sensor, also known as path radiance. Adjacency effect radiation originates from background regions outside the instantaneous field of view (IFOV), which is then scattered or refracted into the sensor.

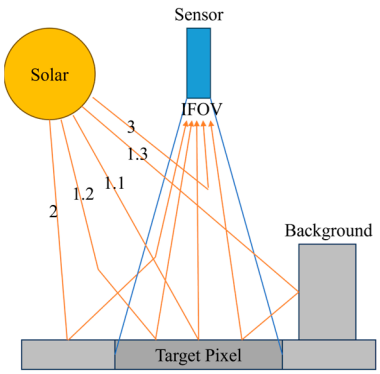


Figure 1. Radiation to the sensor.

With these components, the radiance at the sensor at top of atmosphere (TOA) can be expressed by the following equation.

$$L_o = \tau L_{zero} + L_p + L_s \quad (1)$$

And the core of the 3D hyperspectral simulation model is to calculate these three parameters, and the following section will present the detailed methodology. Among these parameters, L_{zero} is computed using the ray tracing algorithm, while τ, L_p and L_s are derived from the atmospheric radiative transfer equation using the widely adopted MOD-

TRAN5 [29]. In addition to the three atmospheric radiation parameters, two additional atmospheric radiation parameters will be calculated using MODTRAN5 in subsequent sections. When performing MODTRAN5 computations, the input parameters include solar angle, sensor angle, atmospheric model, aerosol model and surface reflectance corresponding to the simulated spectral bands. The following section focuses on the ray tracing algorithm for calculating zero-distance reflected radiance L_{zero}.

2.2. Ray Tracing Algorithm

The backward ray tracing simulations, based on geometric optics principles, simulate radiative transfer processes including ray transmission, reflection, and scattering to precisely quantify radiance contributions. This study uses the backward ray tracing algorithm to trace rays arriving at the sensor. Leveraging the reversibility principle of radiative transfer, rays are traced from the sensor back to their source (direct solar radiation or scattered radiation). The backward ray tracing algorithm comprises three key phases: sensor-originated ray-casting, ray-scene intersection calculation, and radiation calculation, which are described below.

2.2.1. Sensor-Originated Ray Casting

Figure 2 illustrates the ray-casting process. In backward ray tracing simulations, the sensor aperture stop serves as the ray-casting origin, with the IFOV geometry defining the ray propagation vectors. Assuming the observed target center is located at the origin of the global coordinate system, with the sensor zenith angle θ_o, sensor azimuth angle φ_o, and the sensor-to-target distance $\boldsymbol{r_o}$. The global coordinate of the sensor $\boldsymbol{p_o}$ can be mathematically expressed by the following equation:

$$\boldsymbol{p_o} = (|\boldsymbol{r_o}|\sin\theta_o\cos\varphi_o, |\boldsymbol{r_o}|\sin\theta_o\sin\varphi_o, |\boldsymbol{r_o}|\cos\theta_o) \tag{2}$$

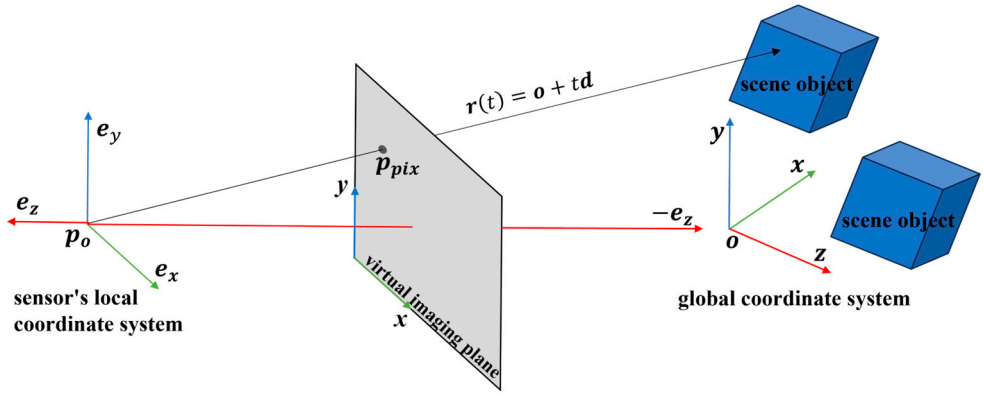


Figure 2. Ray-casting sketch.

This model employs a right-handed Cartesian coordinate system where the simulated image's horizontal and vertical axes align with $\boldsymbol{e_x}$ and $\boldsymbol{e_y}$, respectively. $-\boldsymbol{e_z}$ direction of the sensor's local coordinate system corresponds to the observation direction of the remote sensor (the remote sensor always observes the origin point of the global coordinate system). Thus, the position $\boldsymbol{p_o}$ of the remote sensor can be calculated using the following equation:

$$\boldsymbol{e_z} = \frac{\boldsymbol{p_o}}{|\boldsymbol{p_o}|} \tag{3}$$

For this study, a zero-roll-angle configuration is adopted, so $\boldsymbol{e_y}$ and $\boldsymbol{e_y}$ can be calculated by following equations:

$$\boldsymbol{e_y} = \frac{\boldsymbol{e_z}\times(0,0,1)}{|\boldsymbol{e_z}\times(0,0,1)|} \tag{4}$$

$$\boldsymbol{e_x} = \boldsymbol{e_y} \times \boldsymbol{e_z} \tag{5}$$

In this study, the sensor's internal optical system is simplified as a pinhole camera model incorporating a virtual imaging plane, where incident rays are projected to form an image. The virtual imaging plane is uniformly discretized along $\boldsymbol{x}$ and $\boldsymbol{y}$ directions into pixel units to characterize the actual pixel distribution of the sensor. Consequently, the casted ray $\boldsymbol{r}(t)$ can be mathematically expressed as:

$$\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}(0 \leq t < \infty) \tag{6}$$

In the equation, $\boldsymbol{o}$ denotes the ray origin, corresponding to the sensor's position $\boldsymbol{p_0}$. t represents the path length from $\boldsymbol{p_0}$ to the scene intersection point. $\boldsymbol{d}$ is the unit direction vector of ray propagation. The unit direction vector $\boldsymbol{d}$ for ray propagation is derived from the pixel-to-aperture geometric relationship. Given the projected position $\boldsymbol{p_{pix}}$ of a pixel on the virtual imaging plane, the calculation equation for vector $\boldsymbol{d}$ is:

$$\boldsymbol{d} = \frac{\boldsymbol{p_{pix}} - \boldsymbol{p_o}}{\left|\boldsymbol{p_{pix}} - \boldsymbol{p_o}\right|} \tag{7}$$

2.2.2. Ray-Scene Intersection Calculation

Following ray casting from the sensor aperture, each ray must be geometrically traced through the 3D scene to simulate the complete radiative transfer path. This study utilizes a 3D scene model composed of triangular mesh elements. Therefore, the core of the ray-scene intersection calculation lies in determining which triangular facet in the scene intersects with the ray. The ray-triangle intersection test consists of two sequential steps: First, verify whether the ray intersects the plane containing the triangular facet and compute the intersection point if such intersection exists. Then, determine whether this intersection point lies within the boundaries of the triangular facet. If both conditions are satisfied, the spatial coordinates $\boldsymbol{P}$ of the intersection point, the optical properties of the triangular facet, and the normal vectors (In 3D model files, each vertex is defined with a normal vector.) of its three vertices are returned for subsequent radiative transfer calculations.

As shown in Figure 3, assuming three vertices of a triangular facet are $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$, and the facet's normal vector can be obtained by the following equation:

$$\boldsymbol{N} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \frac{(\boldsymbol{p_2} - \boldsymbol{p_1}) \times (\boldsymbol{p_3} - \boldsymbol{p_1})}{\left|(\boldsymbol{p_2} - \boldsymbol{p_1}) \times (\boldsymbol{p_3} - \boldsymbol{p_1})\right|} \tag{8}$$

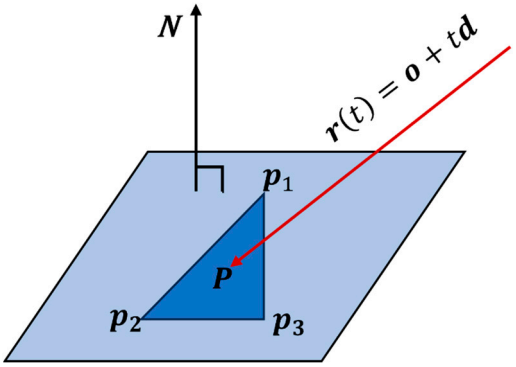


Figure 3. Diagram of ray-triangle intersection.

The normal vector of the plane containing a triangular facet is identical to the facet's normal vector, thus the plane equation can be expressed as:

$$Ax + By + Cz + D = 0 \tag{9}$$

Given that all three vertices ($\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$) of the triangular facet lie on the plane, substituting any one of them into the plane equation yields:

$$D = -\left(Ap_{1x} + Bp_{1y} + Cp_{1z}\right) = -\boldsymbol{N}\cdot\boldsymbol{p_1} \tag{10}$$

The intersection of a ray with a plane then can be transformed into determining whether the ray is perpendicular to $\boldsymbol{N}$. If they are perpendicular, there is no intersection. If not, assume the ray hits a point $\boldsymbol{P}$ on the plane. Based on Equation (6), $\boldsymbol{P}$ can be expressed by the following equation:

$$\boldsymbol{P} = \boldsymbol{o} + t\boldsymbol{d} \tag{11}$$

Since $\boldsymbol{P}$ lies on the plane, the plane can be expressed by the following equation:

$$\boldsymbol{P}\cdot\boldsymbol{N} + D = 0 \tag{12}$$

Based on the above equations, the propagation distance of the ray t can be calculated as:

$$t = \frac{\boldsymbol{N}\cdot\boldsymbol{p_1} - \boldsymbol{o}\cdot\boldsymbol{N}}{\boldsymbol{d}\cdot\boldsymbol{N}} \tag{13}$$

If $t > 0$, it indicates a valid intersection between the ray and the plane. After calculating the intersection point $\boldsymbol{p}$, the barycentric coordinate method is employed to determine whether $\boldsymbol{p}$ lies within the current triangular facet. If the inclusion conditions are satisfied, the ray is confirmed to intersect with that triangular facet. The algorithm iterates through all triangular facets in the scene and selects the intersection point with the smallest distance t as the final valid intersection between the ray and the scene.

2.2.3. Radiation Calculation

As the radiative transfer calculation method remains consistent across all spectral bands, we omit the wavelength dependence notation for all parameters in this section.

Given the occurrence of multiple reflection events during ray propagation, a single-ray path may produce a sequence of discrete intersection points with scene geometries. In this context, we define $\boldsymbol{P_1}$ as the first ray-scene intersection point along the propagation path. The reflected radiance at point $\boldsymbol{P_1}$ is calculated based on its spatial coordinates, normal vector (obtained through a weighted average of the normal vectors at the three vertices of the triangular facet), and optical properties using the following equation:

$$L_{zero,j}(\boldsymbol{P_1}, \boldsymbol{\omega_0}) = \int_{\xi^2} f_{\boldsymbol{P_1}}(\boldsymbol{P_1}, \boldsymbol{\omega_i}, \boldsymbol{\omega_0}) L_j(\boldsymbol{P_1}, \boldsymbol{\omega_i}) cos\theta_i d\boldsymbol{\omega_i} \tag{14}$$

In the above equation, $L_{zero,j}$ denotes the zero-distance reflected radiance of a single ray, $\boldsymbol{\omega_0}$ represents the outgoing direction of reflected radiation, $f_{\boldsymbol{P_1}}(\boldsymbol{P}_1, \boldsymbol{\omega_i}, \boldsymbol{\omega_0})$ signifies the bidirectional reflectance distribution function (BRDF) at point $\boldsymbol{P_1}$, which depends on the material's optical properties at the intersection, $\boldsymbol{\omega_i}$ indicates the incident direction of incoming radiation, L_i corresponds to the incident radiance of a single ray, and θ_i designates the angle between the incident direction and the normal vector. Considering the approximately Lambertian characteristics of the simulated surface materials, an isotropic BRDF is implemented for terrain objects. The BRDF at point $\boldsymbol{P_1}$ can be calculated from its measured reflectance $\rho_{\boldsymbol{P_1}}$:

$$f_{\boldsymbol{P_1}}(\boldsymbol{P_1}, \boldsymbol{\omega_i}, \boldsymbol{\omega_0}) = \frac{\rho_{\boldsymbol{P_1}}}{\pi} \tag{15}$$

As demonstrated by Equation (12), the calculation of the outgoing radiance at intersection point $\boldsymbol{P_1}$ requires integrating radiance over the entire hemispherical space, which is computationally intractable in practice. To address this, we employ a Monte Carlo integration method with single random sampling to approximate the integral. The convergence of the Monte Carlo estimator depends solely on the number of samples, indicating that accurate estimates can be achieved when a sufficient number of rays are traced [30]. A detailed discussion of the required number of rays will be presented in Section 3.

Therefore, based on Monte Carlo integration methodology, $L_{zero,j}(\boldsymbol{P_1}, \boldsymbol{\omega_0})$ can be estimated through a combined approach of direct and indirect radiation sampling. The incident radiance L_j at point $\boldsymbol{P_1}$ can be categorized into two components based on their origins: the direct radiance L_d, obtained from direct radiation sampling, and indirect radiance L_{id}, which consists of solar scattered radiation and background-reflected radiation, obtained from indirect radiation sampling, which is randomly sampled in the upper hemisphere. Thus, the zero-distance reflected radiance of a single ray can be calculated through the following equation:

$$L_{zero,j}(\boldsymbol{P_1}, \boldsymbol{\omega_0}) = L_d(\boldsymbol{P_1}, \boldsymbol{\omega_i}) f_{\boldsymbol{P_1}}(\boldsymbol{P_1}, \boldsymbol{\omega_i}, \boldsymbol{\omega_0}) cos(\theta_d) + \frac{1}{n}\sum \frac{L_{id}(\boldsymbol{P_1}, \boldsymbol{\omega_i})(\boldsymbol{P_1}, \boldsymbol{\omega_i}, \boldsymbol{\omega_0}) cos(\theta_{id})}{pdf_{\boldsymbol{P_1}}(\boldsymbol{\omega_i})} \quad (16)$$

In the above equation, θ_d and θ_{id} denote the angles between the normal vector and the sampling ray (the ray cast from the intersection point for radiance sampling) directions. And n denotes the sampling ray count. The Sensor will cast thousands of primary rays per pixel, thus generating only one indirect radiance sampling ray at each intersection point is sufficient to meet the sampling requirements, while avoiding the exponential increase in the number of sampling rays caused by recursive processes. The sampling directions are distributed according to the probability density function $pdf_{\boldsymbol{P_1}}(\boldsymbol{\omega_i})$. Following the principle of hemispherical radiance importance sampling, the relationship between $pdf_{\boldsymbol{P_1}}(\boldsymbol{\omega_i})$ and θ_{id} is given by $pdf_{\boldsymbol{P_1}}(\boldsymbol{\omega_i}) = cos(\theta_{id})/\pi$.

For direct radiance sampling, an indirect radiance sampling ray will be cast from the intersection point $\boldsymbol{P_1}$ toward the solar direction, followed by an intersection test with the scene. If an occlusion is detected, the direct solar irradiance at point $\boldsymbol{P_1}$ equals zero (indicating that $\boldsymbol{P_1}$ lies within a shadow region). Conversely, if no intersection occurs, the direct solar radiance L_d can be computed through the direct solar irradiance E_{sd} (obtained from MODTRAN5): $L_d = E_{sd}$.

The calculation of L_{id} depends on whether the indirect sampling ray intersects with the scene. If the ray hits on the sky, then $L_{id} = L_{ss}$, L_{ss} represents the solar scattered radiance reaching the point $\boldsymbol{P_1}$, obtained from MODTRAN5. If the ray hits surrounding objects (Assuming the new intersection point is $\boldsymbol{P_2}$), then $L_{id} = L_{bs}$, L_{bs} represents the background-reflected radiance received at $\boldsymbol{P_1}$ from $\boldsymbol{P_2}$, This value is calculated using a recursive method: The direct and indirect radiance sampling procedures are recursively performed at point $\boldsymbol{P_2}$ to compute the background-reflected radiance contribution. If the indirect radiance sampling ray from $\boldsymbol{P_2}$ intersects the scene again at $\boldsymbol{P_3}$, the direct and indirect radiance sampling operations are recursively repeated at $\boldsymbol{P_3}$. This recursive process continues until either the indirect sampling ray escapes into the atmosphere, or the preset maximum recursion depth is reached. It is generally accepted that when the recursion depth reaches four bounces, the contribution of background-reflected radiance from the final intersection point to $\boldsymbol{P_1}$ becomes negligible. Thus, the radiance from the terminal intersection could be approximated as zero. At this point, the reflected radiance of one ray $L_{zero,j}(\boldsymbol{P_1}, \boldsymbol{\omega_0})$ can be solved.

During ray casting, each sensor pixel casts multiple rays for Monte Carlo sampling. Consequently, the final zero-distance reflected radiance L_{zero} for the pixel is consequently

obtained by computing the arithmetic mean of zero-distance reflected radiance values from all sampled rays. Incorporating atmospheric effects, radiance at the sensor is then calculated as follows:

$$L_o = \frac{1}{n}\sum_{j=1}^{n} L_{zero,j}(\boldsymbol{P_1}, \boldsymbol{\omega_0})\tau + L_p + L_s \quad (17)$$

3. Implementation and Acceleration

3.1. Implementation of Model Based on OpenGL

3.1.1. Implementation of RGB Ray Tracing

As a graphics rendering API specially designed for RGB rendering, OpenGL's rendering pipeline inherently lacks direct applicability to hyperspectral simulation. Figure 4 illustrates the traditional OpenGL rendering pipeline.

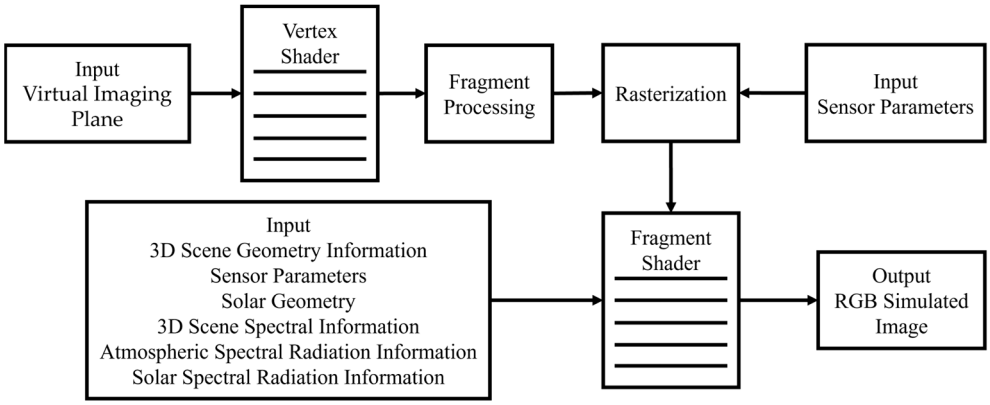


Figure 4. Traditional OpenGL rendering pipeline.

As depicted in Figure 4, the virtual imaging plane is initially input into the vertex shader, which processes and transmits the plane's vertices to subsequent pipeline stages. During the fragment processing and rasterization phases, the virtual imaging plane undergoes meshing segmentation, where each mesh unit corresponds to a sensor pixel element. The meshed virtual imaging plane is then transferred to the fragment shader. The fragment shader constitutes the core component of the entire rendering pipeline. By integrating the input simulation parameters, it performs radiometric calculations using ray tracing algorithms and ultimately outputs the simulated image. Within this rendering pipeline architecture, both the vertex shader and fragment shader are custom-implemented using OpenGL Shading Language (GLSL), while fragment processing and rasterization are fixed phases.

3.1.2. Implementation of Hyperspectral Ray Tracing

The current rendering pipeline can only perform RGB ray tracing simulation, making it difficult to achieve hyperspectral simulation because of two primary limitations: radiometric quantification and hyperspectral representation.

In terms of radiation quantification, OpenGL's computational output is written to the "fragColor" array in the fragment shader. This array is constrained to a 4-channel × 2-byte format, meaning the radiation calculation results for each spectral band can only be represented with 2 bytes. This inevitably leads to unavoidable data compression of the radiation values. To address this limitation, we established a customized mapping relationship that transforms the 4-channel × 2-byte data format into a single-channel × 8-byte format. This approach enables an 8-byte representation of single-band floating-point radiation calculation results, thereby overcoming OpenGL's inherent quantization insufficiency.

For hyperspectral applications, since the current rendering pipeline can only obtain single-ray single-band radiance data for each pixel per complete computational execution, simulating hundreds of spectral bands requires executing a complete computational process for each ray and each band. Each such computational process requires computationally intensive ray-scene intersection operations, resulting in prohibitive efficiency degradation for large-scale hyperspectral simulations. The pseudocode for implementing hyperspectral simulation using this approach is presented in Algorithm 1 below.

Algorithm 1 TraditionalRayTracing

```
InitializeRayTracingRenderer(vertex_shader, fragment_shader)
InputParameters(scene_geometry, sensor_params, solar_geometry,
scene_spectral_data, atmospheric_radiation, solar_radiation)
for each band:
    for each ray:
        Calculate ray path and radiance → single_band_single_ray_image
        Store in single_band_multi_ray_image
    end for
    Store single_band_multi_ray_image in multi_band_image
end for
```

Considering that the ray path remains identical across all bands in hyperspectral simulation, and the computational load of radiation is significantly lower than that of ray-scene intersection, we have modified the OpenGL rendering pipeline modifications as illustrated in Figure 5. For a streamlined figure presentation, the sensor parameter input during rasterization has been omitted.

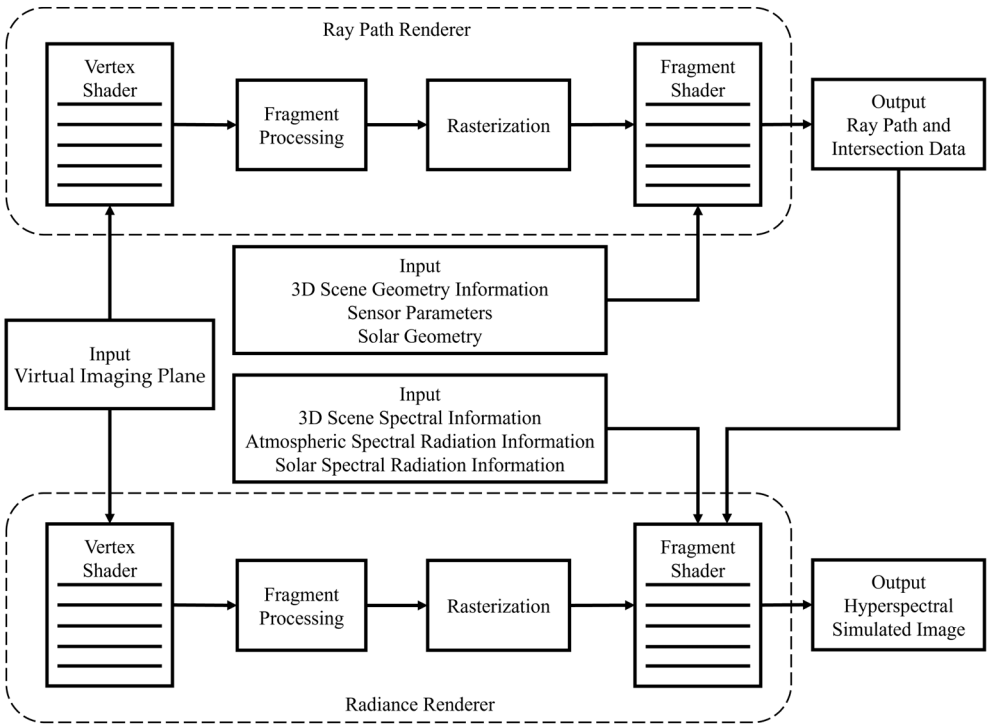


Figure 5. OpenGL rendering pipeline after modification.

The modified rendering pipeline divides the original ray tracing renderer into two serially executed renderers: the Ray Path Renderer and the Radiance Renderer.

The Ray Path Renderer exclusively computes ray transport paths; thus, its input only contains geometry-related parameters. Since the calculation results (material index at ray-scene intersection, cosine between surface normal and incident ray, and cosine between surface normal and outgoing ray) do not require high-precision representation, the "fragColor" array in the Ray Path Renderer maintains a 4-channel × 2-byte format (RBGA), where 3 channels can record information for a single-ray-scene intersection. However, due to ray reflections in the scene, a single radiance transport path typically involves multiple intersections, making one "fragColor" array insufficient for recording all intersection data. OpenGL provides an off-screen rendering technique called Multiple Render Targets (MRT), which supports simultaneous output of shader computation results to up to 8 texture maps. This key technology is implemented through Framebuffer Objects (FBO), which overcome the functional limitations of traditional default framebuffers by creating customizable virtual framebuffers. By creating up to 8 "fragColor" arrays and binding them to 8 FBOs, respectively, we can obtain up to 64-byte output channels, sufficient to record all intersection information along a radiance transport path. Since the radiance transport paths are identical across all spectral bands, the Ray Path Renderer only needs to execute once per ray, eliminating the need for per-band computation. The recorded intersection data are stored in texture maps and serve as one of the inputs for the Radiance Renderer to participate in radiance calculations.

The Radiance Renderer performs only radiance computations. By taking precomputed radiance transport paths from the Ray Path Renderer as input, it only requires one processing pass per spectral band to generate a hyperspectral remote sensing image at the sensor's entrance pupil. The pseudocode after modifying the rendering pipeline is in Algorithm 2 as follows.

Algorithm 2 ModifiedRayTracing

```
InitializeRayPathRenderer(vertex_shader, path_calculation_shader)
InputParameters(scene_geometry, sensor_params, solar_geometry)
for each ray:
    Calculate ray path → single_ray_data
    Store in multi_ray_data
end for
InitializeRadianceRenderer(vertex_shader, radiance_calculation_shader)
InputParameters(multi_ray_data, scene_spectral_data,
    atmospheric_radiation, solar_radiation)
for each band:
    Calculate radiance → single_band_multi_ray_image
    Store in multi_band_image
end for
```

The modified hyperspectral rendering pipeline no longer requires ray-scene intersection calculations for each spectral band, thereby effectively improving the efficiency of hyperspectral simulation.

3.1.3. Efficiency Verification of the Simulation

Through an example, we compare the simulation time of different numbers of bands after OpenGL rendering pipeline transformation. The simulation conditions of the example are as follows: the number of 3D scene's facets is 1.4 million, the number of pixels is 350 × 350, the number of rays tracked by each pixel is 2000, and the number of simulated

bands is 3, 100, 200, 300, 400, 500 and 600, respectively. The facet partitioning of the 3D scene is illustrated in Figure 6 and the simulation results are shown in Table 2.

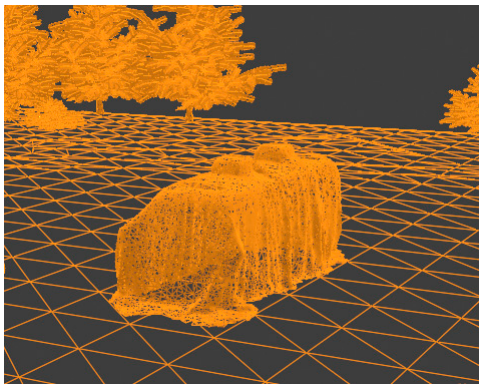


Figure 6. The facet partitioning of the 3D scene.

Table 2. Simulation time of different number of bands.

Number of bands	3	100	200	300	400	500	600
Simulation time (GPU: RTX 4080)	166.42 s	203.75 s	255.73 s	294.54 s	333.93 s	373.06 s	395.58 s

By comparing and analyzing the simulation time of different band numbers, while the RGB 3-band simulation requires 166.42 s, the 600-band hyperspectral simulation completes in 395.58 s, which is merely 2.4 times the RGB computation time. These experimental results demonstrate that the efficiency of hyperspectral simulation after modification has been improved, which has reached the same order of magnitude as the RGB simulation time. The term 'RGB simulation' here refers to the modified rendering pipeline implementation. In the original rendering pipeline, since no data transfer between renderers is required, it is about 25 s faster than the modified RGB simulation.

3.2. Model Acceleration

Even if the pipeline modification reduces intersection calculations to a single operation per simulation, the ray tracing algorithm still requires considerable computational resources. Especially for the 3D hyperspectral simulation model, the model itself has a pursuit of radiation calculation accuracy. While the number of rays directly determines simulation accuracy, the resultant computational burden—excessive runtime and resource consumption—hinders the practical application of the hyperspectral simulation model. We therefore developed an acceleration framework that preserves the required precision standards. Therefore, we have implemented the following acceleration optimization operations for the model while maintaining accuracy requirements:

3.2.1. Accelerate at the Algorithmic Level

At the algorithmic level, we use the bounding volume hierarchy algorithm and the low-difference sequence sampling method to improve the intersection efficiency and sampling efficiency.

Bounding volume hierarchy is a classic design idea of space acceleration structure. By recursively grouping scene objects and constructing bounding boxes for each group, a tree-like hierarchical relationship is formed, which reduces the number of invalid intersection tests of triangular facets, and improves the efficiency of algorithms such as ray tracing and collision detection [31].

In practical applications, the bounding volume hierarchy establishes a progressive filtering mechanism from global to local scales through its hierarchical bounding box structure. Mathematically, this constructs a mapping relationship that transforms continuous 3D space into a discrete tree-based representation. The standard BVH construction initiates from a root node encompassing all facets and employs a recursive algorithm with the following procedure:

(1) Compute the bounding box of the current node
(2) Select the optimal splitting axis (typically via SAH* evaluation)
(3) Determine the splitting plane position
(4) Generate left/right child nodes

This recursive process continues iteratively until the node's facet count falls below a specified threshold (such as 8 facets). The last node is called the leaf node.

In ray tracing, the process starts from the root node and sequentially checks the intersection between the ray and the bounding boxes of its child nodes. For leaf nodes that intersect with the ray, all contained triangular facets are further examined; nodes that do not intersect, along with their child nodes, are skipped directly.

During radiative sampling, the integration error stems from the random clustering and voids of sample points inherent in independent random sampling. In contrast, low-discrepancy sequences position each newly generated sample point to "proactively compensate" for previously uncovered regions, thereby avoiding the clustering and voids characteristic of random sampling. By generating highly uniform and non-repetitive sample points in the integral domain, the low-difference sequence makes the sampling ray evenly distributed, and the Monte Carlo integral of the radiation sampling can converge quickly, thereby improving the sampling efficiency [32].

3.2.2. Explore the Balance Between Accuracy and Efficiency

Regarding the balance between accuracy and efficiency, we optimized the number of traced rays. Although as mentioned above, the increase in the number of rays determines the calculation accuracy, the boundary benefit of increasing rays will gradually decrease with the increase of its number. At the same time, the increase in the number of rays will also lead to a decrease in computational efficiency. The complexity of ray tracing is $O(n)$, meaning that it is linearly positively correlated with the number of rays n.

Therefore, from the perspective of balancing the calculation efficiency and calculation accuracy, we discuss the relationship between the number of rays and the calculation accuracy and try to find an optimal number of rays, so that the balance between calculation efficiency and calculation accuracy can be achieved at this number. We carried out this research by increasing the number of rays for the same example. We take the simulation result with 10,000 rays as the reference standard and statistically analyze the difference between each simulation result under a different number of rays and the reference standard, and the comparative parameters consist of the mean value and variance of pixel-wise radiance relative differences. We also compared the simulation time across varying ray counts to validate their linear correlation. The mean value, variance and simulation time are shown in Figure 7.

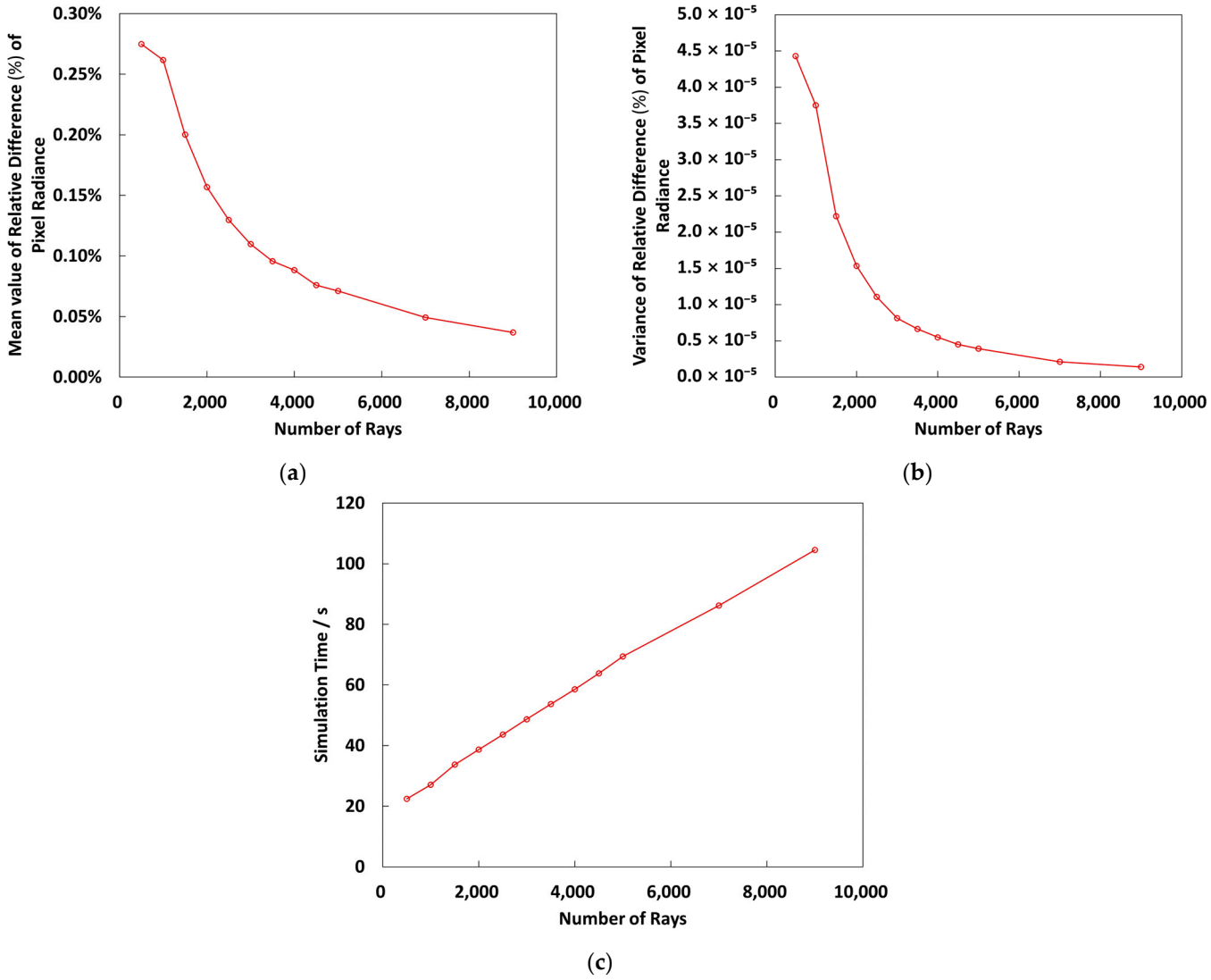


Figure 7. (**a**) Mean value of relative difference of pixel radiance under different number of rays; (**b**) Variance of relative difference of pixel radiance under different number of rays; (**c**) Simulation time under different number of rays.

The simulation with 2000 rays achieves a runtime of 38.7 s, representing a 70.4% reduction compared to 130.6 s of 10,000 rays, while maintaining excellent simulation accuracy as evidenced by a 0.16% mean relative radiance difference and a 1.54×10^{-5} variance. This optimized configuration delivers a 70.4% computational efficiency enhancement with only a 0.16% acceptable accuracy compromise. Therefore, we choose 2000 rays as the best balance between efficiency and calculation accuracy.

To verify the generalizability of this conclusion, we conducted a further simulation experiment on a complex urban scene. The scene features a relatively complex geometric structure and dozens of material types. The facet partitioning of the 3D scene is illustrated in Figure 8.

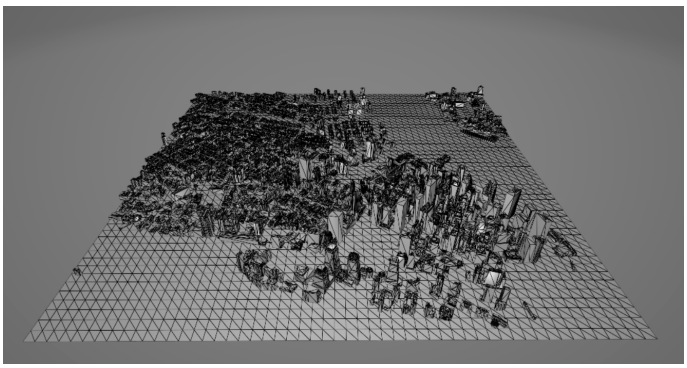


Figure 8. The facet partitioning of the city 3D scene.

The results of the simulation experiment are shown in Figure 9.

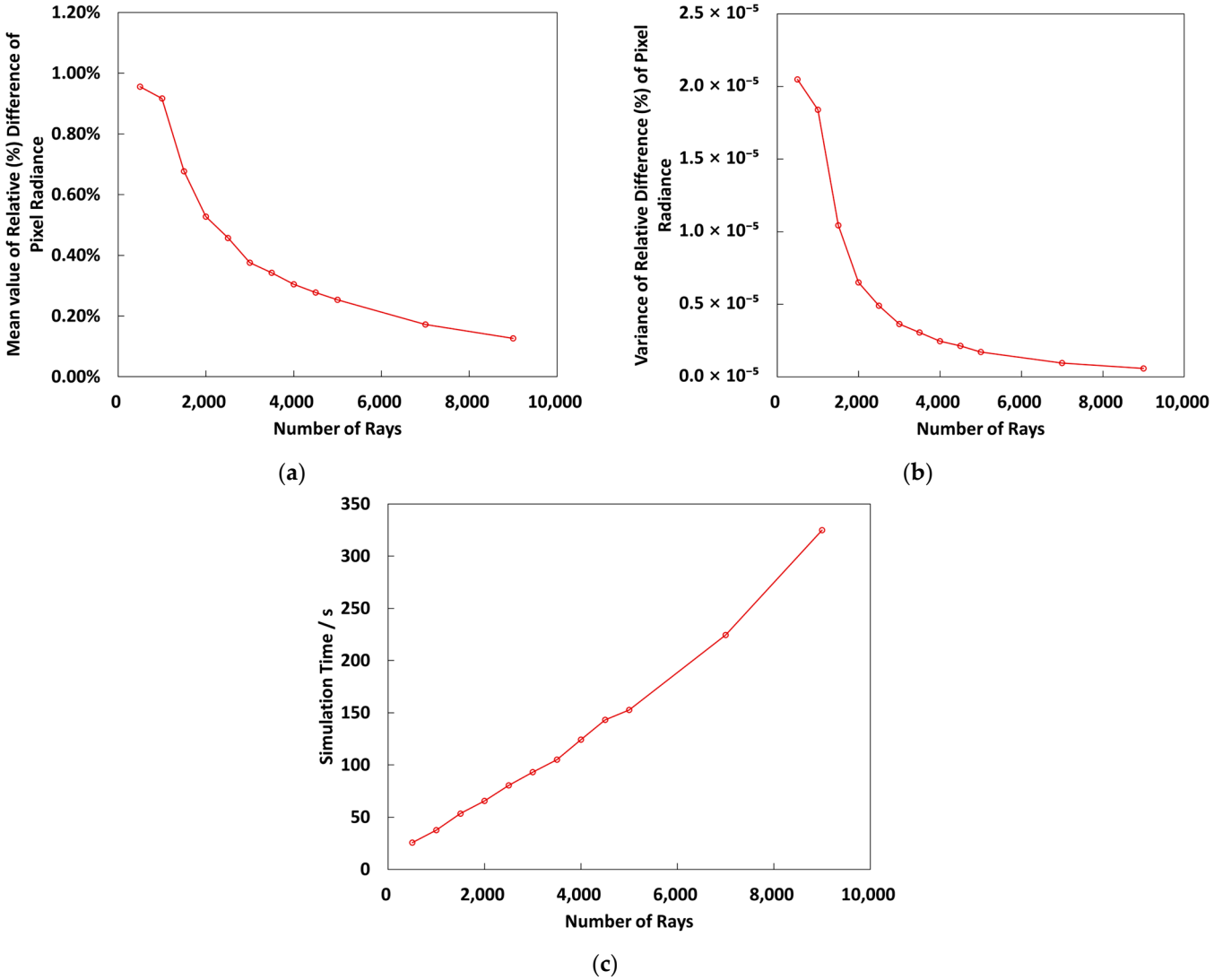


Figure 9. (**a**) Mean value of relative difference of pixel radiance under different number of rays; (**b**) Variance of relative difference of pixel radiance under different number of rays; (**c**) Simulation time under different number of rays.

The results show that when the number of rays is 2000, the simulation time is 65.5 s, representing an 83.2% reduction compared to 390.1 s of 10,000 rays, while maintaining excellent simulation accuracy as evidenced by a 0.53% mean relative radiance difference

and a 6.51 $\times$ 10^{-5} variance. It can be seen that as the geometric and material complexity of the scene increases, the simulation error with 2000 rays, while larger, remains within an acceptable range. However, if the scene complexity further increases (such as in dense vegetation), 2000 rays may no longer meet the required simulation accuracy.

4. UAV-Based Data Validation

Following the establishment of our hyperspectral remote sensing imaging simulation model, we further conducted simulation validation using UAV-based data. The data collection was conducted in Weichang County, Chengde City, Hebei Province where a flat terrain was selected as the test site. The site included dry grassland, grassland, bare soil and trees (spruce). A net-covered Toyota Coaster vehicle served as the primary validation target at the site center. The experiment mainly includes four aspects: measured data acquisition, 3D model construction, reflectivity data acquisition and simulation comparison.

4.1. Acquisition of UAV Data

The X20P airborne hyperspectral imager manufactured by AUZP Scientific (Beijing, China) is a frame-based imaging system designed for UAV platforms, featuring a 20MP CMOS sensor as its core component. This device acquires hyperspectral data across more than 160 spectral channels in frame imaging mode, with continuous spectral coverage from 350 to 1000 nm. Incorporating stability-enhancing design features, the system demonstrates exceptional compatibility with UAV operations [33]. Detailed technical specifications are provided in Table 3.

Table 3. Specific parameters of X20P.

spectral range	350~1000 nm
number of pixels	1886×1886 pixels/frame
number of spectral channels	164 (extensible)
detector	20 MP hyperspectral imagery CMOS
imaging modes	synchronized imaging of all channels of the full array, global shutter
optics array/FOV	66/35°
angle jitter amount	±0.015°
stabilization range	pitch direction: ±40°, Rolling direction: ±45°

4.2. 3D Model Construction

The study first constructed a 70 m $\times$ 70 m 3D model of the test site centered on the target vehicle as the input for the hyperspectral simulation. The model provides a detailed division of vehicle materials, including air-conditioning metal, silver metal, yellow metal, bare metal, white metal, glass and rubber. And we divided the differentiation of background surface feature material, including dry grassland, road, grassland, vehicle and leaf. All 3D models were stored in OBJ format.

The 3D models of the actual vehicle and the model vehicle are shown in Figure 10.

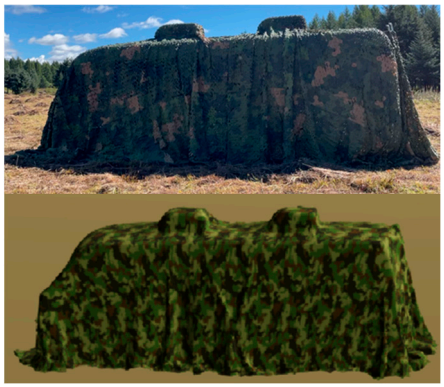


Figure 10. Actual vehicle and model vehicle.

4.3. Reflectance Data Acquisition

Material reflectivity, as an intrinsic property of materials, serves as a critical parameter for radiative transfer calculations in simulation processes. Precise measurement of material reflectivity constitutes an essential prerequisite for simulation validation. In this study, field measurements were conducted to simultaneously acquire real-scene data while systematically collecting reflectance characteristics of both target vehicles and the surrounding environments. These field measurements were performed using the SR3501 spectrometer manufactured by AUZP Scientific (Beijing, China).

Based on the actual material distribution of the vehicle and comprehensive field environmental investigation, the on-site materials were systematically classified into distinct categories according to their spectral characteristics and representativeness. The classification scheme includes grassland, dry grassland, spruce, bare soil, wood, net-light green, net-deep green, net-medium green and net-brown, with each category assigned a unique identification code.

For reflectance characterization, each material specimen was subjected to 30 replicate measurements to ensure measurement reliability. The raw reflectance data subsequently underwent standardized processing comprising data format normalization, outlier rejection using statistical criteria, multi-scan averaging to improve signal-to-noise ratio, spectral absorption feature processing and spectral smoothing using appropriate digital filters. The resulting processed reflectance spectra are presented in Figure 11.

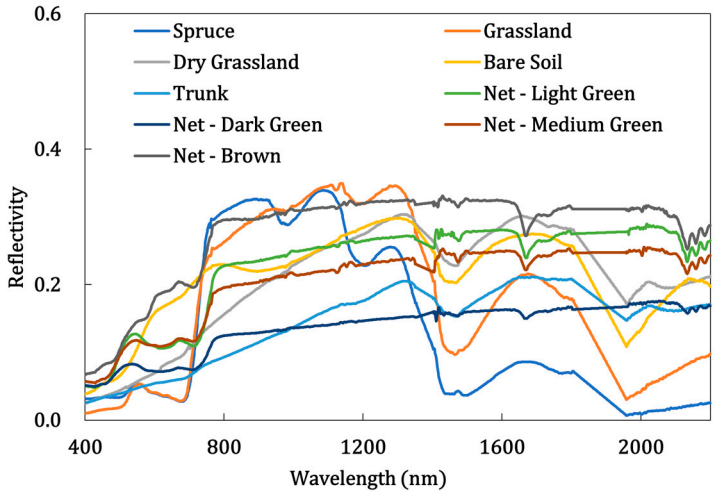


Figure 11. Reflectance curves of main materials in test site.

4.4. Validation Results

For 3D hyperspectral simulation, the superior quantification capability was the primary criterion for selecting this methodology. Therefore, we choose to compare the spectral radiance obtained by simulation with the spectral radiance obtained by UAV. The area selected for comparative verification is the average pure pixel radiance of the vehicle roof, dry grassland, green grassland and canopy. The parameters used in the comparison include spectral angle and similarity, which are defined as follows.

Spectral Angle (SA):

$$SA = arccos\left(\frac{\sum_{i=1}^{n} x_i \hat{x}_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} \hat{x}_i^2}}\right) \tag{18}$$

Similarity:

$$SIM = \frac{Cov(x,\hat{x})}{\sigma_x \cdot \sigma_{\hat{x}}} \%100 \tag{19}$$

Here, x and $\hat{x}$ denote the spectral curves of the simulated image and real image, x_i and $\hat{x}_i$ denote the radiance values of the spectral curves from the simulated image and real image at band i. These images and the comparison verification areas selected are shown in Figure 12.

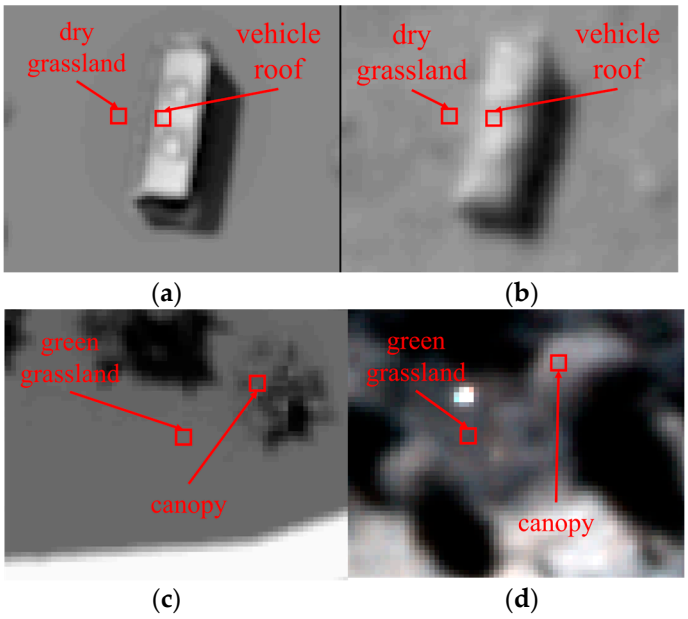


Figure 12. (**a**) Simulation image (band: 530 nm) of vehicle and dry grassland; (**b**) Real image (band: 530 nm) of vehicle and dry grassland; (**c**) Simulation image (band: 530 nm) of green grassland and canopy. (**d**) Real image (band: 530 nm) of comparison verification area.

Since no high-fidelity 3D modeling was performed for the green grassland and canopy, their 3D models differ from reality. Here we focus solely on their reflected radiance spectra. The bright spot in the real image is the reflectance standard panel the spectral radiance curves are shown in Figure 13.

The similarity of the vehicle roof's spectral radiance curve obtained by simulation and UAV is 93.02% with a spectral angle of 5.5804°. And for dry grassland, the spectral radiance similarity is 92.76% and the spectral angle is 7.284°. For both the green grassland and canopy, their simulated spectral radiance curves achieved over 97% similarity to the measured data, with spectral angles all within 8°. The spectral response function (SRF) of a sensor alters the original radiance signal through band integration effects, resulting in the smoothing of narrowband absorption features in observed data—a phenomenon

referred to as "spectral blurring", so the atmospheric absorption of simulation image is more obvious than that of the real image. These results show a strong correlation, indicating a high accuracy of the simulation model.

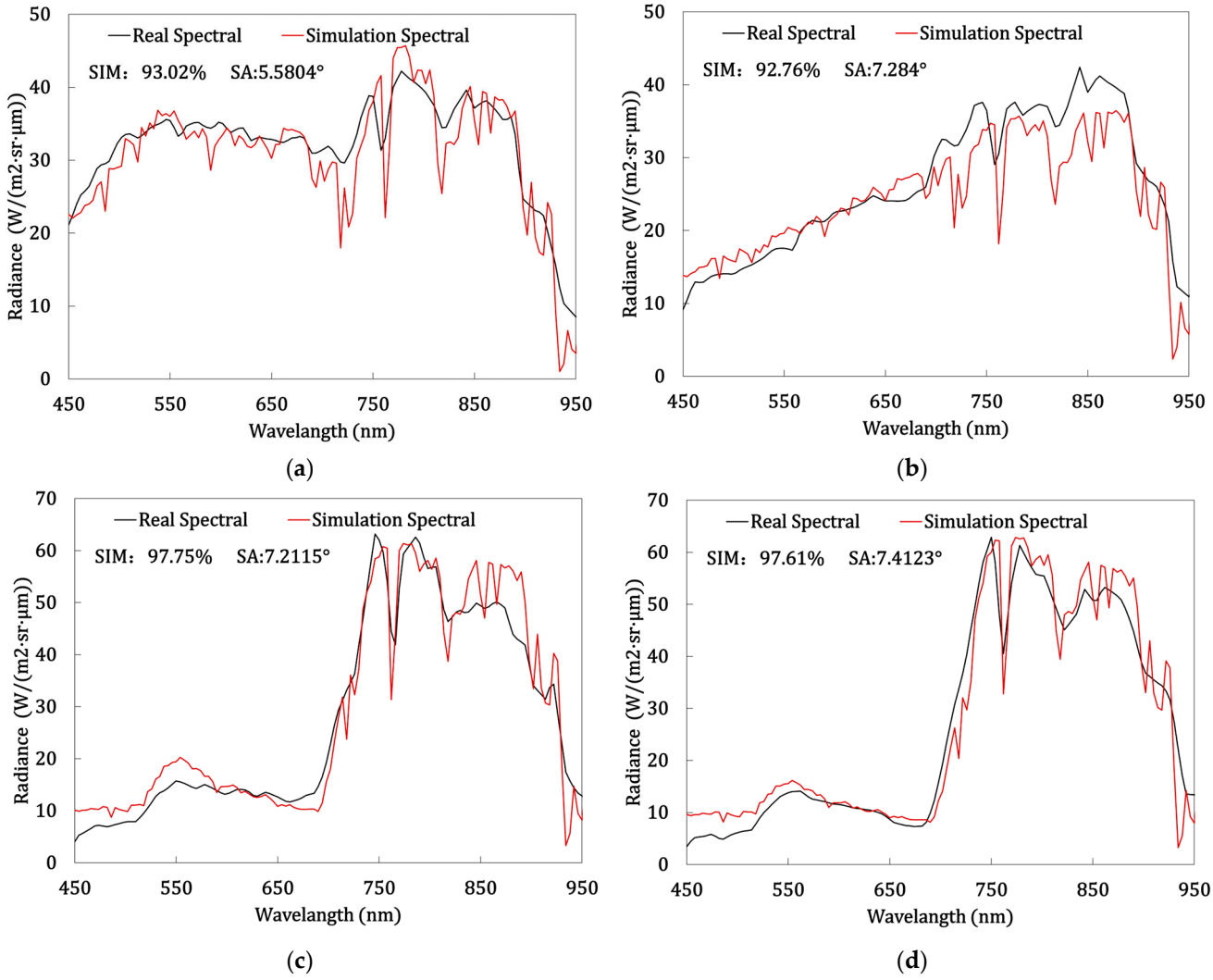


Figure 13. (**a**) Comparison of measured and simulated vehicle roof spectral radiance; (**b**) Comparison of measured and simulated dry grassland spectral radiance; (**c**) Comparison of measured and simulated green grassland spectral radiance; (**d**) Comparison of measured and simulated canopy spectral radiance.

5. Discussion

Building on this model, additional simulation experiments under varying environmental variables and observational conditions were conducted to analyze the model's capability in representing different situations. The experimental results demonstrate that the model effectively characterizes the effects of background conditions, solar altitude angle and sensor altitude angle on vehicle hyperspectral signatures.

5.1. Effect of Background on Spectral Radiance

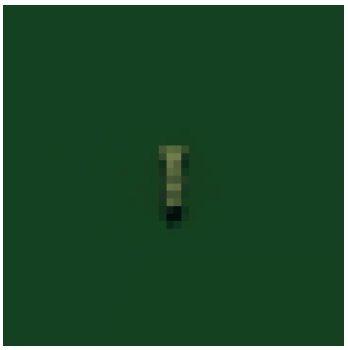
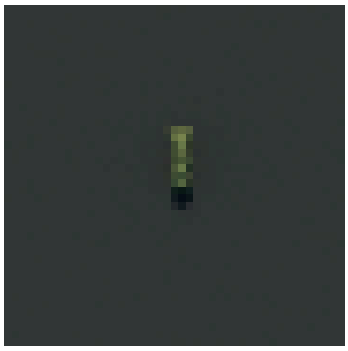
With all other conditions being rigorously held constant, the vehicle is selected to be placed under different background conditions (grassland, woodland, desert, concrete, asphalt road) for simulation. The other simulation conditions are shown in Table 4.

Table 4. Simulation conditions without backgrounds.

solar altitude angle	45°
solar azimuth angle	180°
sensor altitude angle	90°
sensor azimuth angle	180°
weather conditions	clear conditions
spectral resolution	5 nm
spatial resolution	1.0 m

The simulation images under different background conditions are shown in Table 5.

Table 5. Simulation images (true color) under different background conditions.

Background Condition	**Grassland**	**Woodland**	**Dessert**
simulated image			
Background condition	Concrete	Asphalt Road	
simulated image			

The spectral radiance information of the vehicle under different backgrounds is extracted, respectively, and the curves are shown in Figure 14.

According to the analysis of the vehicle spectral radiance curves, the holistic variation trend of the curves under different backgrounds is similar, but the amplitudes are different. This discrepancy arises because background materials with distinct reflective properties produce varying reflected radiation, which subsequently alters the background-reflected radiation received by the target vehicle.

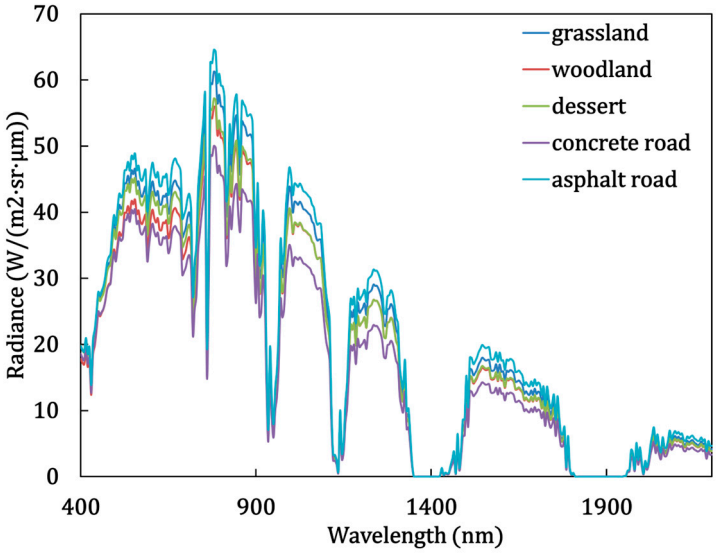


Figure 14. Vehicle spectral radiance under different backgrounds.

5.2. Effect of Solar Altitude Angle on Spectral Radiances

With all other conditions being rigorously held constant, the simulation is carried out under different solar altitude angles (15°, 30°, 45°, 60°, 75°, 90°). The other simulation conditions are shown in Table 6.

Table 6. Simulation conditions without solar altitude angle.

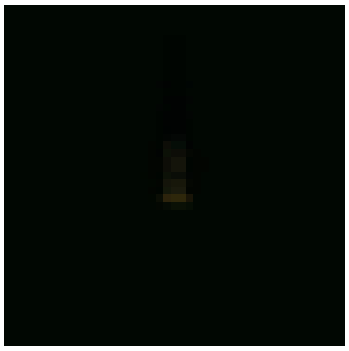
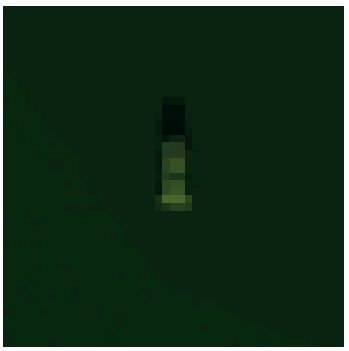
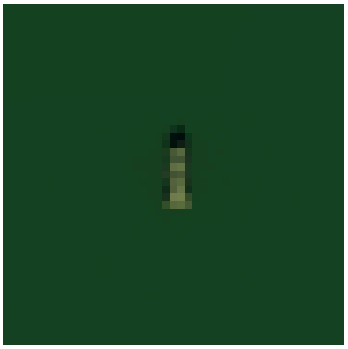
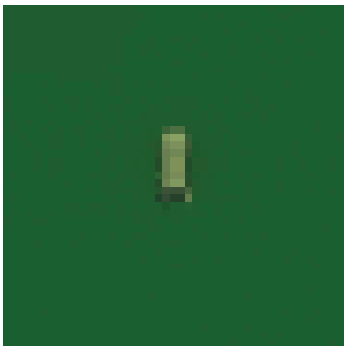
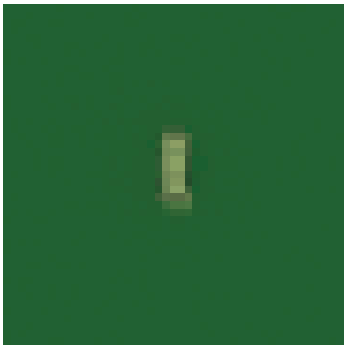
solar azimuth angle	180°
sensor altitude angle	90°
sensor azimuth angle	180°
background conditions	grassland
weather conditions	clear conditions
spectral resolution	5 nm
spatial resolution	1.0 m

The simulation images under different solar altitude angles are shown in Table 7.

The spectral radiance information of the vehicle at different solar altitude angles is extracted, respectively, and the curves are shown in Figure 15.

According to the analysis of the vehicle spectral radiance curves, a positive correlation exists between solar altitude angle and vehicle radiance. This phenomenon is primarily attributed to two governing factors. From the perspective of surface reflectance, an increase in solar altitude angle leads to a greater inclination of solar rays relative to the ground surface. This results in higher solar irradiance per unit area, consequently enhancing the surface-reflected radiation. From the perspective of atmospheric radiative transfer, an increase in solar altitude angle shortens the atmospheric path length traversed by solar radiation. This reduction in path length decreases atmospheric scattering and absorption effects, leading to higher atmospheric transmittance. Consequently, the radiant energy ultimately reaching the ground surface is significantly enhanced.

Table 7. Simulation images (true color) under different solar altitude angles.

Solar Altitude Angle	**15°**	**30°**	**45°**
simulated image			
Solar altitude angle	60°	75°	90°
simulated image			

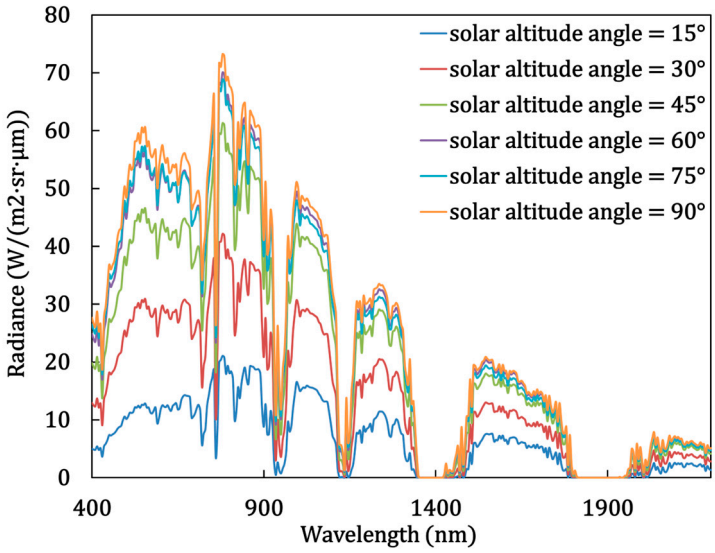


Figure 15. Vehicle spectral radiance under different solar altitude angles.

5.3. Effect of Sensor Altitude Angle on Spectral Radiance

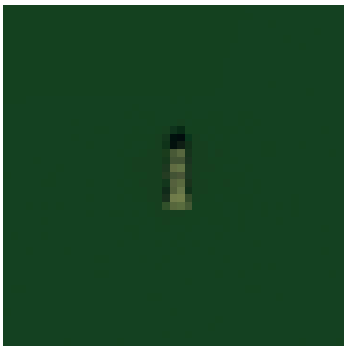
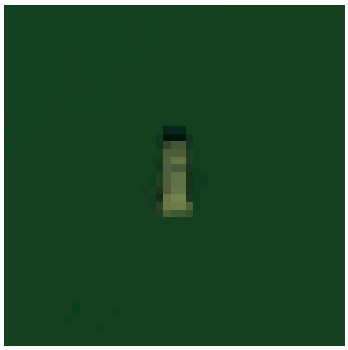
With all other conditions being rigorously held constant, the simulation is carried out under different sensor tail-down angles (90°, 80°, 70°). The other simulation conditions are shown in Table 8.

Table 8. Simulation conditions without sensor altitude angles.

solar altitude angle	45°
solar azimuth angle	180°
sensor azimuth angle	180°
background conditions	grassland
weather conditions	clear conditions
spectral resolution	5 nm
spatial resolution	1.0 m

The simulation images under different sensor altitude angles are shown in Table 9.

Table 9. Vehicle spectral radiance under different backgrounds.

Sensor Altitude Angle	**90°**	**80°**	**70°**
simulated image			

The spectral radiance information of the vehicle at different ground angles of the remote sensor is extracted, respectively, and the curves are shown in Figure 16.

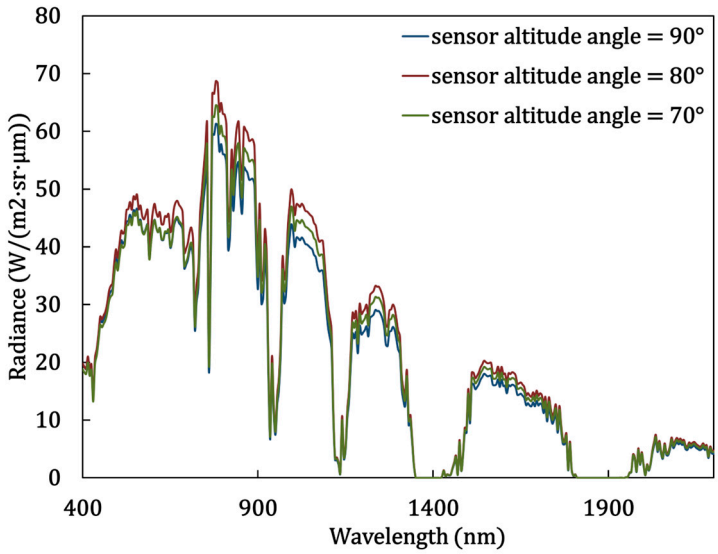


Figure 16. Vehicle spectral radiance in different sensor altitude angles.

According to the analysis of the vehicle spectral radiance curves, the holistic variation trends and the amplitudes of the curves under different backgrounds are similar. It is because the net covering the vehicle exhibits a near-Lambertian reflectance property, demonstrating consistent spectral radiance across all reflection angles.

5.4. Section Summary

Based on the above analysis, the 3D hyperspectral simulation model shows promising multi-condition simulation capabilities, accurately characterizing the effects of different background conditions, solar angles, and sensor viewing geometries on hyperspectral imaging, with results suggesting general agreement with theoretical predictions.

6. Conclusions

This paper proposed a 3D hyperspectral simulation model based on ray tracing and OpenGL. A radiative transfer model is first discussed, and based on it, a hyperspectral calculation model based on ray tracing is brought out in this research. OpenGL Rendering Pipeline is modified to implement it. Two main modifications are performed to realize hyperspectral ray tracing simulation: ray tracing simulation is achieved through customization of vertex and fragment shaders in the rendering pipeline, and building upon this, hyperspectral ray tracing simulation is realized by sequentially executing two custom renderers dedicated to geometric and radiometric computations, respectively. Algorithm acceleration is then researched for better simulation speed, hierarchical bounding box and low-difference sequence are used to accelerate the model itself, and the relationship between the number of rays and the calculation accuracy is discussed to find out the balance of simulation efficiency and accuracy. Results show a ray count of 2000 is appropriate. On this basis, this model achieved a 6-minute simulation with 600 bands and a scene with 1.4 million facets.

Furthermore, the simulation accuracy of this model is validated based on UVA data in aspects of spectral radiance, both UAV spectral radiance data and ground-reflectance data are acquired in this research, the former is used for validation and the latter provides the input for the simulation. Results show that the spectral similarity between real data and simulation results is more than 90%. Moreover, the simulation under different conditions is carried out based on the model. The results show that the model can well reproduce the influence of these factors on radiance.

This simulation model has the following limitations: (1) High-fidelity 3D models are required for simulating specific real-world scenarios. (2) Spatial and spectral response characteristics of sensors are not accounted for in the imaging process. (3) Self-emission radiation is not considered, currently preventing simulations in the infrared band. (4) The use of a simplified Lambertian BRDF model introduces certain computational inaccuracies.

For future research, the influence of the imaging effect of the sensor will be considered for remote sense imaging simulation, and the band can be further extended to the infrared in the spectral range. Moreover, we plan to conduct additional field measurements to acquire the reflectance data needed for BRDF modeling.

Author Contributions: Conceptualization, X.L., W.Z. and B.W.; methodology, X.L. and B.W.; software, X.L.; validation, X.L., W.Z., B.W. and M.J.; formal analysis, X.L. and B.W.; investigation, X.L.; resources, W.Z. and B.W.; data curation, X.L. and B.W.; writing-original draft preparation, X.L., W.Z. and B.W.; writing-review and editing, X.L., W.Z., B.W., H.Q., M.J. and P.Q.; visualization, X.L. and B.W.; supervision, W.Z.; project administration, W.Z.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Goetz, A.F.; Vane, G.; Solomon, J.E.; Rock, B.N. Imaging spectrometry for Earth remote sensing. *Science* **1985**, *228*, 1147–1153. [CrossRef]
2. Lu, B.; Dao, P.D.; Liu, J.; He, Y.; Shang, J. Recent Advances of Hyperspectral Imaging Technology and Applications in Agriculture. *Remote Sens.* **2020**, *12*, 2659. [CrossRef]
3. Liu, B.; Tong, Q.X.; Zhang, L.F.; Zhang, X.; Yue, Y.M.; Zhang, B. Monitoring Spatio-Temporal Spectral Characteristics of Leaves of Karst Plant during Dehydration Using a Field Imaging Spectrometer System. *Spectrosc. Spectr. Anal.* **2012**, *32*, 1460–1465. [CrossRef]
4. Shi, T.; Chen, Y.; Liu, Y.; Wu, G. Visible and near-infrared reflectance spectroscopy-An alternative for monitoring soil contamination by heavy metals. *J. Hazard. Mater.* **2014**, *265*, 166–176. [CrossRef]
5. Liang, L.; Di, L.; Zhang, L.; Deng, M.; Qin, Z.; Zhao, S.; Lin, H. Estimation of crop LAI using hyperspectral vegetation indices and a hybrid inversion method. *Remote Sens. Environ.* **2015**, *165*, 123–134. [CrossRef]
6. Goetz, A.F.H. Three decades of hyperspectral remote sensing of the Earth: A personal view. *Remote Sens. Environ.* **2009**, *113*, S5–S16. [CrossRef]
7. Shimoni, M.; Haelterman, R.; Perneel, C. Hyperspectral Imaging for Military and Security Applications Combining myriad processing and sensing techniques. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 101–117. [CrossRef]
8. Tang, G. Research on Visible Hyperspectral Imaging of Ground Sea Target (Chinese). Master's Thesis, Xidian University, Xi'an, China, 2021.
9. Zhang, B.; Chen, Z.C.; Zheng, L.F.; Tong, Q.X.; Liu, Y.N.; Yang, Y.D.; Xue, Y.Q. Object detection based on feature extraction from hyperspectral imagery and convex cone projection transform. *J. Infrared Millim. Waves* **2004**, *23*, 441–445+450.
10. Lapadatu, M.; Bakken, S.; Grotte, M.E.; Alver, M.; Johansen, T.A. Simulation Tool for Hyper-Spectral Imaging From a Satellite. In Proceedings of the 2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, The Netherlands, 24–26 September 2019; p. 5. [CrossRef]
11. Zhao, S.; Zhu, X.; Tan, X.; Tian, J. Spectrotemporal fusion: Generation of frequent hyperspectral satellite imagery. *Remote Sens. Environ.* **2025**, *319*, 114639. [CrossRef]
12. Li, J.; Zheng, K.; Gao, L.; Ni, L.; Huang, M.; Chanussot, J. Model-Informed Multistage Unsupervised Network for Hyperspectral Image Super-Resolution. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 3391014. [CrossRef]
13. Cao, Y.; Cao, Y.; Wu, Z.; Yang, K. A Calculation Method for the Hyperspectral Imaging of Targets Utilizing a Ray-Tracing Algorithm. *Remote Sens.* **2024**, *16*, 1779. [CrossRef]
14. Goodenough, A.A.; Brown, S.D. DIRSIG5: Next-Generation Remote Sensing Data and Image Simulation Framework. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 4818–4833. [CrossRef]
15. Wang, Y.; Kallel, A.; Zhen, Z.; Lauret, N.; Guilleux, J.; Chavanon, E.; Gastellu-Etchegorry, J.-P. 3D Monte Carlo differentiable radiative transfer with DART. *Remote Sens. Environ.* **2024**, *308*, 114201. [CrossRef]
16. Qi, J.; Xie, D.; Yin, T.; Yan, G.; Gastellu-Etchegorry, J.-P.; Li, L.; Zhang, W.; Mu, X.; Norford, L.K. LESS: LargE-Scale remote sensing data and image simulation framework over heterogeneous 3D scenes. *Remote Sens. Environ.* **2019**, *221*, 695–706. [CrossRef]
17. Gastellu-Etchegorry, J.-P.; Lauret, N.; Yin, T.; Landier, L.; Kallel, A.; Malenovsky, Z.; Al Bitar, A.; Aval, J.; Benhmida, S.; Qi, J.; et al. DART: Recent Advances in Remote Sensing Data Modeling With Atmosphere, Polarization, and Chlorophyll Fluorescence. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 2640–2649. [CrossRef]
18. Han, Y.; Lin, L.; Sun, H.; Jiang, J.; He, X. Modeling the space-based optical imaging of complex space target based on the pixel method. *Optik* **2015**, *126*, 1474–1478. [CrossRef]
19. Han, Y.; Chen, M.; Sun, H.; Zhang, Y.; Kong, J. Imaging simulation method of TG-02 accompanying satellite's visible camera. *Infrared Laser Eng.* **2017**, *46*, 1218002. [CrossRef]
20. Chenfei, Y.; Hao, Z.; Guofeng, Z. Research on infrared imaging simulation technology of ocean scene. *Proc. SPIE* **2021**, *12065*, 1206513. [CrossRef]
21. Xin, Y.; Yan, Z.; Xiao-Tian, C.; Feng, Z.; Jun-Jun, Z. An Infrared Radiation Simulation Method for Aircraft Target and Typical Ground Objects Based on RadThermIR and OpenGL. In Proceedings of the 2016 International Conference on Information Systems and Artificial Intelligence (ISAI), Hong Kong, China, 24–26 June 2016; pp. 236–240. [CrossRef]
22. Jiang, W.; Zhao, Y.; Yuan, S. Modeling and OpenGL simulation of sea surface infrared images. *Electron. Opt. Control* **2009**, *16*, 19.
23. Shen, T.; Guo, M.; Wang, C. IR image generation of space target based on OpenGL. *Proc. SPIE—Int. Soc. Opt. Eng.* **2007**, *6786*, 1383–1388. [CrossRef]
24. Kramer, S.; Gritzki, R.; Perschk, A.; Roesler, M.; Felsmann, C. Numerical simulation of radiative heat transfer in indoor environments on programmable graphics hardware. *Int. J. Therm. Sci.* **2015**, *96*, 345–354. [CrossRef]
25. Hu, H.-h.; Feng, C.-y.; Guo, C.-g.; Zheng, H.-j.; Han, Q.; Hu, H.-y. Research on infrared imaging illumination model based on materials. *Proc. SPIE—Int. Soc. Opt. Eng.* **2013**, *8907*, 89070T. [CrossRef]

26. Bian, Z.J.; Qi, J.B.; Wu, S.B.; Wang, Y.S.; Liu, S.Y.; Xu, B.D.; Du, Y.M.; Cao, B.; Li, H.; Huang, H.G.; et al. A review on the development and application of three dimensional computer simulation mode of optical remote sensing. *Natl. Remote Sens. Bull.* **2021**, *25*, 559–576. [CrossRef]
27. Chen, C. Key Techniques Studying on High-resolution Remote Sensing Imaging Simulation (Chinese). Ph.D. Thesis, University of Science and Technology of China, Hefei, China, 2018.
28. Song, B.; Fang, W.; Du, L.; Cui, W.; Wang, T.; Yi, W. Simulation method of high resolution satellite imaging for sea surface target. *Infrared Laser Eng.* **2021**, *50*, 20210127. [CrossRef]
29. Ma, L. Study of the Properties of the Atmospheric Radiative Transfer Software MODTRAN5 (Chinese). Master's Thesis, University of Science and Technology of China, Hefei, China, 2016.
30. Doidge, I.C.; Jones, M.; Mora, B. Mixing Monte Carlo and progressive rendering for improved global illumination. *Vis. Comput.* **2012**, *28*, 603–612. [CrossRef]
31. Vitsas, N.; Evangelou, I.; Papaioannou, G.; Gkaravelis, A. Parallel Transformation of Bounding Volume Hierarchies into Oriented Bounding Box Trees. *Comput. Graph. Forum* **2023**, *42*, 245–254. [CrossRef]
32. Doi, A. Applications of low discrepancy sequences for computer graphics. *J. Jpn. Soc. Simul. Technol.* **2003**, *22*, 232–238.
33. Xiao, Y.; Chen, J.; Xu, Y.; Guo, S.; Nie, X.; Guo, Y.; Li, X.; Hao, F.; Fu, Y.H. Monitoring of chlorophyll-a and suspended sediment concentrations in optically complex inland rivers using multisource remote sensing measurements. *Ecol. Indic.* **2023**, *155*, 111041. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.