remote sensing

MDPI

Article

Hybrid Scene Structuring for Accelerating 3D Radiative Transfer Simulations

Jianbo Qi [1,2,3,*], Tiangang Yin [4], Donghui Xie [3] and Jean-Philippe Gastellu-Etchegorry [2]

1 Key Laboratory for Silviculture and Conservation of Ministry of Education, Beijing Forestry University, Beijing 100083, China
2 Centre d'Etudes Spatiales de la Biosphere, Université de Toulouse, CNRS, CNES, IRD, UT3, Toulouse 31401, France; jean-philippe.gastellu@iut-tlse3.fr
3 Faculty of Geographical Science, Beijing Normal University, Beijing 100875, China; xiedonghui@bnu.edu.cn
4 Earth System Science Interdisciplinary Center, University of Maryland, College Park, MD 20740-3823, USA; tiangang@smart.mit.edu
* Correspondence: jianboqi@bjfu.edu.cn; Tel.: +86-10-6233-8133

Received: 9 October 2019; Accepted: 5 November 2019; Published: 12 November 2019

Abstract: Three-dimensional (3D) radiative transfer models are the most accurate remote sensing models. However, presently the application of 3D models to heterogeneous Earth scenes is a computationally intensive task. A common approach to reduce computation time is abstracting the landscape elements into simpler geometries (e.g., ellipsoid), which, however, may introduce biases. Here, a hybrid scene structuring approach is proposed to accelerate the radiative transfer simulations while keeping the scene as realistic as possible. In a first step, a 3D description of the Earth landscape with equal-sized voxels is optimized to keep only non-empty voxels (i.e., voxels that contain triangles) and managed using a bounding volume hierarchy (BVH). For any voxel that contains triangles, within-voxel BVHs are created to accelerate the ray–triangle intersection tests. The hybrid scheme is implemented in the Discrete Anisotropic Radiative Transfer (DART) model by integrating the Embree ray-tracing kernels developed at Intel. In this paper, the performance of the hybrid algorithm is compared with the original uniform grid approach implemented in DART for a 3D city scene and a forest scene. Results show that the removal of empty voxels can accelerate urban simulation by 1.4×~3.7×, and that the within-voxel BVH can accelerate forest simulations by up to 258.5×.

Keywords: radiative transfer; ray tracing; bounding volume hierarchy

1. Introduction

Understanding and modeling the radiative behavior at Earth's surface is essential for a wide range of scientific domains including agriculture, forestry and urban microclimate. The applications vary from structural [1] and biophysical parameter retrieval [2] of vegetation, to urban radiation flux studies [3]. For vegetation monitoring, much attention has been paid to radiative transfer modeling and numerous physical models have been developed in the past decades to describe the interactions between solar radiation and complex canopies [4]. Compared to one-dimensional (1D) models (e.g., SAIL [5]), which treat the canopy as a horizontally homogeneous medium, three-dimensional (3D) radiative transfer (RT) models can take into account the highly heterogeneous Earth surfaces. Accurate and efficient 3D models are essential tools for deriving accurate parameters from remote sensing measurements [6], as well as for validating satellite-based geo-physical products [7]. This is notably true as 3D information is more and more available with the development of 3D acquisition technology. For example, LiDAR (Light Detection and Ranging) is more and more used in forest

studies for measuring the vegetation 3D attributes [8,9], which are important input parameters for 3D radiative transfer models. However, a drawback of 3D RT models is that they are more difficult to parameterize, especially for some LUT (look up table)-based inversion methods of remote sensing acquisitions. For example, hundreds of thousands of simulations and representatives of different environmental and instrumental conditions may be needed to build an invertible LUT database [10]. Thus, the consideration of computational efficiency becomes crucial in these kinds of applications. In particular, 3D RT models should take advantage of the techniques developed in the computer graphics domain to optimize the computation. The poor efficiency is mainly due to the fact that RT models are designed to simulate remote sensing acquisitions that are very accurate in terms of radiometry, and not simply realistic from the human point of view, conversely to most computer graphics algorithms.

In 3D RT models, the complex landscape elements are described with triangle meshes, geometrical objects and turbid media. Triangle meshes can be used to simulate very precise 3D structures (e.g., position of each leaf), while turbid media can be used to simulate the canopy structure with statistical parameters (e.g., LAD: leaf area density and LOD: leaf orientation distribution). To simulate large and complex scenes, the triangle and turbid elements in 3D space must be organized into efficient computer data structures to avoid unmanageable computer constraints in terms of volume memory and computation time [11], e.g., the algorithm that tests all objects one by one to determine if an intersection occurs is too computationally expensive to simulate complex scenes [12]. Most of the RT models rely on a space subdivision technique to avoid some redundant computations. With space subdivision, objects which are located in a subspace (or voxel) which is not intersected by a ray can be directly excluded from the intersection tests, which can significantly increase the efficiency when handling scenes with a large number of elements. The space subdivision often partitions the space into hierarchical subspaces (or voxels), where the scene elements are located [13]. This very important technique is heavily used in some ray-tracing-based (discrete-ordinate or Monte-Carlo) RT models, for achieving computationally efficient ray-object intersections [14–16].

Up until now, various space subdivision approaches have been implemented, including uniform grid [17], BSP (binary space partitioning) tree [18], octree [13] and BVH (bounding volume hierarchies) [19] etc. Because the performances of these approaches usually depend on the specific scene structures, no single technique can be universally the best [20]. The uniform grid is well suited for homogeneous scenes, i.e., landscape elements are relatively uniformly distributed in space, while other approaches, such as BVH, are better for inhomogeneous scenes. These facts lead us to develop a hybrid scene structure which can combine the advantages of the uniform grid and BVH tree, which is also self-adaptive to different scene structures.

In this paper, we present a hybrid scene structuring scheme that is designed to accelerate the radiative transfer modeling in the Discrete Anisotropic Radiative Transfer (DART) model, because the latter one simulates the radiative budget and remote sensing images of Earth landscapes using the uniform grid. DART is one of the most comprehensive physical-based models [6]. It can simulate the 3D radiative transfer budget and various remote sensing acquisitions (e.g., LiDAR, spectroradiometer and solar-induced fluorescence, etc.) over the whole optical domain, including atmosphere. Compared to other models, DART can work on scenes that are simulated with any combinations of triangles and fluids, including the so-called turbid medium that is often used for giving a statistical representation of vegetation. This newly designed structuring scheme combines the advantages of uniform grid and BVH, which enables DART to simulate larger scenes under higher spatial resolutions with lower computational cost. This paper is structured as follows. Section 2 presents the new hybrid scene structuring algorithm and its implementation in DART. In Sections 3 and 4, the hybrid algorithm is compared to the original DART, in terms of accuracy, computer time and memory usage. Section 5 concludes this paper and gives the outlook for future improvements.

2. Methods

2.1. Uniform Grid in Discrete Anisotropic Radiative Transfer (DART)

DART uses the discrete-ordinate ray-tracing method to simulate the propagation and interaction of radiation in the "Earth-Atmosphere" scene (Figure 1), which is represented as a 3D array of rectangular voxels. The tracked radiation is along N directions that discretize the 4π space. Each direction Ω_i is associated with a solid angle $\Delta\Omega_i$ with $\sum_{i=1}^{N} \Delta\Omega_i = 4\pi$ sr [21]. The 3D scene contains two major parts: the atmosphere and the Earth scene. Its usefulness is illustrated here with its so-called "reflectance mode", where the sun is the only radiation source. At the beginning, the propagation of radiation in the atmosphere is simulated [22], which gives the direct sun irradiance and diffuse atmospheric radiation that reach the voxels of the top of the Earth scene at the bottom of atmosphere (BOA). Then, this so-called BOA radiation illuminates the Earth scene. A typical Earth scene can usually contain landscape elements, such as trees, grass, houses and water. These elements are represented with any combination of triangles, fluids and turbid media.

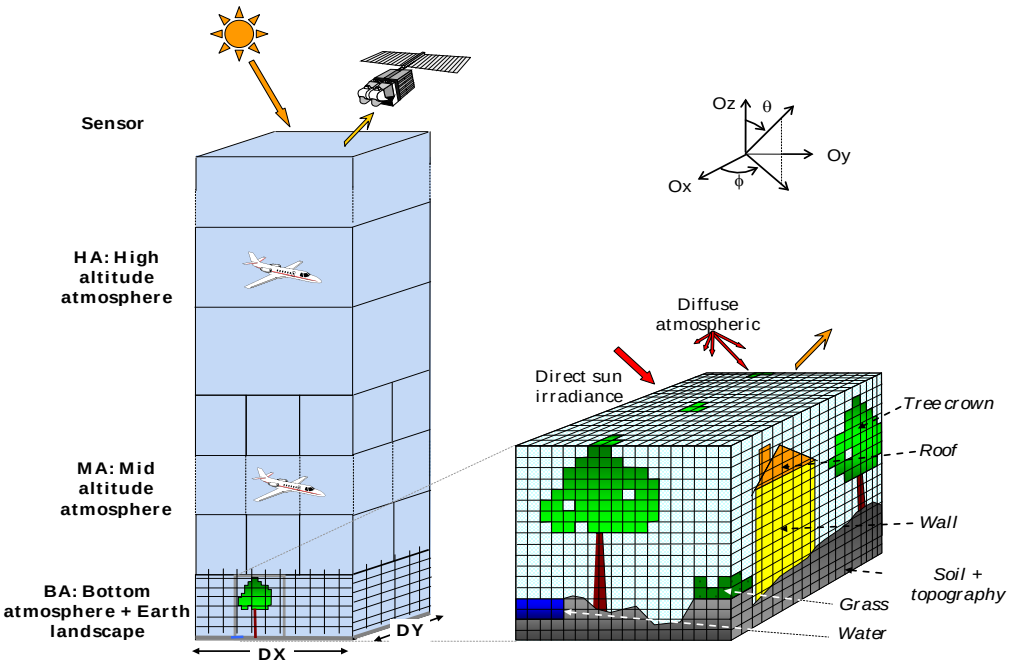


Figure 1. Radiative transfer modeling in the "Earth-Atmosphere" scene of the Discrete Anisotropic Radiative Transfer (DART) model. The scene of DART is divided into two parts: Earth and Atmosphere. Earth is represented with regular voxels while Atmosphere is represented with different horizontal layers.

As presented in Figure 1, DART partitions the Earth scene into equal-sized voxels, which are used to organize the landscape elements (triangles, fluids and turbid media). Equal-sized voxels are mandatory for simulating the 3D radiative budget (e.g., absorbed radiation) of Earth's landscapes. For each triangle, a triangle-voxel intersection test [23] is performed to determine the voxel that contains it. The turbid medium is just a set of descriptive statistical parameters (e.g., LAD and LOD for vegetation), and it usually fills the whole voxel. To store this information in memory for radiation tracking, a list of voxels is first created. The length of this list is equal to the number of voxels in the scene. For each voxel, DART stores a list of references of all the triangles which intersect this voxel and a list of turbid media that are in the voxel (Figure 2).

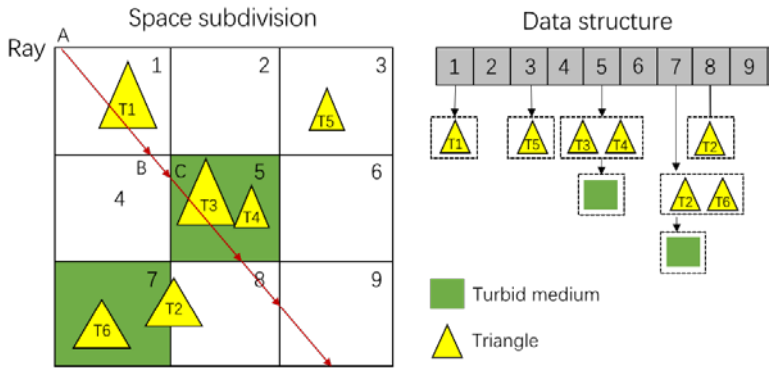


Figure 2. Space subdivision and data structure.

The radiative transfer with the Earth scene (landscape) starts by tracing rays from the BOA illumination plane to the scene. Figure 2 illustrates the case of a ray that enters a voxel at the entry point A and then crosses five voxels. When a ray enters a voxel, ray-triangle tests are performed sequentially for all the triangles inside the voxel. Intersection occurs for the closest triangle that intersects the incident ray. Usually, the order of the intersections is not known, thus a sorting algorithm is performed, which may require large computation time if too many triangles are inside the voxel. If the ray intersects a triangle, scattering is computed using the optical properties of this triangle. If present, turbid media is taken into account, both for transmittance and scattering, including within voxel single and multiple scattering. If no triangle intersection is found, the ray crosses the voxel, and an exit point (e.g., point B in Figure 2) will be calculated by a ray-plane intersection algorithm [24]. Then, it is very straightforward to find the next voxel that the ray will enter, because the exit face of the current voxel is also the entry face of the next voxel. It is a major advantage of the uniform grid partitioning technique. However, this step-by-step algorithm cannot skip empty spaces (e.g., voxel 4, 8 and 9 in Figure 2), which may lead to many redundant computations in highly heterogeneous scenes with large empty gaps. This uniform voxel approach may also consume a lot of memory, since all the voxels are identical and need to be created, except that some of them store a link to a list of triangles or turbid media.

2.2. *Hybrid Scene Structuring*

2.2.1. Voxel-Level Ray Tracking

DART uses the voxel as the basic unit to calculate the transferred energy in the Earth scene. This approach is also adopted in our new hybrid scene approach, since voxels can also support turbid medium, which makes it possible to simulate triangle and turbid medium simultaneously in a single scene. The voxels in the 3D space are categorized into three groups: non-empty voxel, empty voxel, and null voxel (Figure 3). The non-empty voxel is the voxel that contains triangles or fluids/turbid media. The empty voxel contains nothing, but they are needed to set up the source rays and store radiation that exits the Earth scene during the ray-tracing process. Therefore, they only exist on the side of the scene. For null voxels, they neither contain any triangles or turbid media nor store any energy, thus, they do not need to be created in the scene, which can save computer memory, especially when the 3D scene contains many empty spaces (e.g., landscape with topography). Null voxels only exist inside the scene.

To skip the empty spaces (null voxels) during the ray-tracing method, the empty and non-empty voxels are organized using a BVH tree, which is well adapted to non-uniform geometry distributions. The BVH used in this study is a binary tree that structures the bounding volumes of all the voxels into a hierarchy structure. As shown in Figure 3b, it first creates a root node that represents the bounding volume of the whole scene. Then, all the voxels are divided into two sides around the midpoint along the longest scene axis (e.g., the x axis). For each side, a bounding volume containing the voxels inside it

is created. In the BVH tree, these two bounding volumes are the child nodes of the root. This partition process is recursively repeated until the child node only contains a small number of voxels (e.g., 1 or 2).

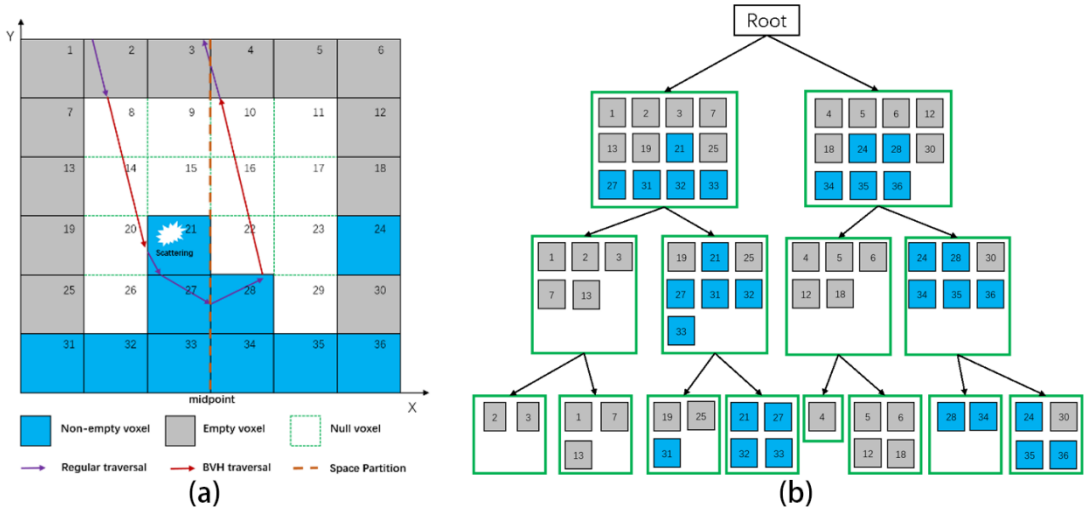


Figure 3. Hybrid scene structuring scheme: (**a**) The structure of a typical scene with non-uniform geometry distribution, (**b**) The corresponding bounding volume hierarchy (BVH) tree which organizes all the non-null voxels.

To track rays in this hybrid scene, the algorithms of uniform grid and BVH are combined. This is illustrated here with a ray that enters a voxel (i.e., voxel 2) on the side of the simulated earth scene (Figure 3a). Then, the computation of the exit point gives the next possible voxel (i.e., voxel 8). If the next voxel is a null voxel, the BVH algorithm is used to determine the next non-null voxel (i.e., voxel 21). For that, the BVH algorithm tests if the ray intersects the bounding volume of the root node. If there is an intersection, the algorithm tests the occurrence of the intersection with child nodes. At the leaf node, the ray will test the ray-voxel intersection with all the voxels, one by one, inside the node using a ray-box intersection algorithm [25]. This binary search scheme is very efficient for finding voxels that are far from the current voxel. However, it is less efficient if the next cell is very close, since the BVH top-down algorithm searches from the root node to leaf node for ray. Thus, as long as a ray meets non-null voxels, the grid tracking method is used. This mechanism makes the model to choose the tracking scheme automatically, and it can adapt to different scene structures without user intervention.

2.2.2. Within-Voxel Ray Tracking

In the hybrid structuring scheme, the triangles are not directly managed by BVH but organized by voxels. When a ray hits a voxel, the algorithm tests the intersection with all the triangles one by one. For some applications (e.g., large voxel size), each voxel may contain many triangles (e.g., 100 triangles). Hence, testing triangle intersections one by one may also slow down the radiative transfer process. Thus, if the number of triangles in a voxel exceeds a threshold, a BVH tree (so called within-voxel BVH) is created inside this voxel to organize all the triangles directly. In that case, during ray tracking, the test of the triangle intersection is calculated with the BVH tree instead of the plain list of triangles. Figure 4 shows a simple 1 m × 1 m scene, which contains only one voxel with N randomly distributed triangles. The number N takes the value 1, 5, 10, 50, 100, 200, 400 or 800. The illumination density is set to 100 rays per square meter, which gives a total number of 100 rays. The total intersection time is measured, which is shown in Figure 4b. It can be seen that the total time increases linearly with the increase of N if the within-voxel BVH algorithm is not used, while it is nearly constant if the within-voxel BVH is used. When N is as low as 10, the time is nearly the same. It explains that the number 10 is used as a threshold to indicate whether a BVH tree should be created in a voxel, which avoids creating BVH trees for voxels that contain only a few triangles (i.e., less than 10). This approach is useful

because the BVH tree also consumes memory. Another advantage of the within-voxel BVH is that it can overcome the floating-point limitation of Embree by offsetting triangle coordinates with voxel coordinates, especially when handling very large scenes, which is very important for the application of the 3D radiative transfer model in satellite products' validation.

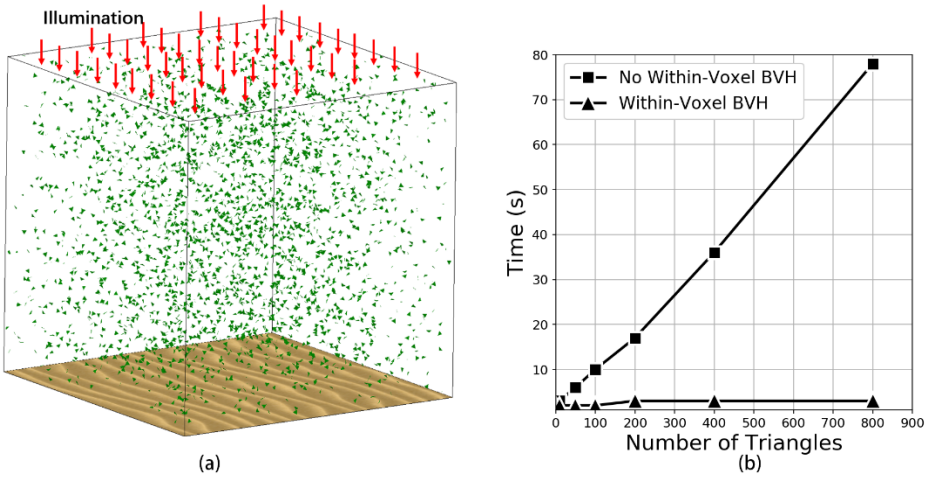


Figure 4. Simulation time varies with the number of triangles in one voxel: (**a**) 1 m × 1 m scene with randomly distributed triangles, (**b**) simulation time under different number of triangles.

2.2.3. Implementation

The hybrid structuring scheme was implemented in DART using a ray-tracing library named Embree [26], which is a collection of high-performance ray-tracing kernels and is optimized for Intel® processors with support for streaming SIMD extensions (SSE) and Intel advanced vector extensions (AVX, AVX2, and AVX-512) instructions. To implement the voxel-level ray tracing, a user-defined geometry that represents the voxel needs to be defined in Embree. A voxel can be described by its central position and size and the ray-voxel intersection algorithm should be provided by users to make the geometry able to be recognized by Embree. The implementation details are illustrated in Appendix A.

When the number of triangles within a voxel exceeds a threshold (e.g., 10), a BVH tree which manages all the triangles inside the voxel is built. In DART, a triangle may belong to different voxels. To avoid copying a triangle several times, we store all the triangles from the whole scene in a vertex array, as illustrated in Figure 5. Each voxel only stores a list of references to the triangles contained. Thus, all the within-voxel BVHs can share the same vertex array in memory.

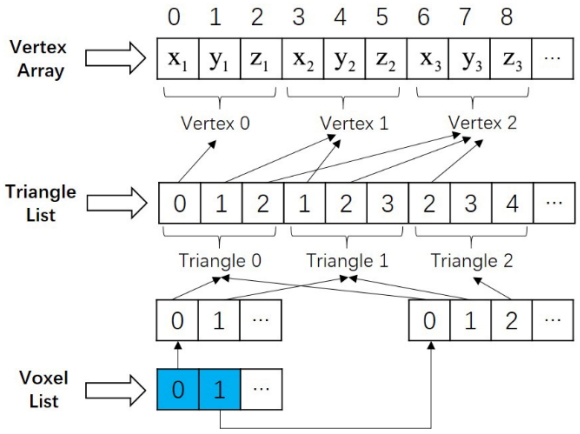


Figure 5. Triangle management in hybrid scene structuring scheme.

Due to floating-point precision problems, a ray scattered by a triangle can be intercepted again by the same triangle. This self-intersection can significantly decrease radiation scattering if it is not well handled. Here, an intersection filtering mechanism is used to avoid this self-intersection problem. Each triangle has a unique ID (Identity Document), if the intersection occurs twice, the second successive intersection will be ignored. This mechanism can be achieved through the intersection filter function provided by Embree and the implementation details can be found in Appendix B.

2.3. Radiation Tracking

When a ray enters a non-empty voxel (i.e., a voxel containing triangles or turbid media), scattering may occur. Figure 6 illustrates the scattering process inside non-empty voxels. For a turbid voxel, suppose an incident ray with radiant power $W_{inc}(\Omega_i)$ enters the cell from direction Ω_i. Then the transmitted radiation which exits the voxel is:

$$W_{trans}(\Delta l, \Omega_i) = W_{inc}(\Omega_i) \cdot T(\Delta l, \Omega_i), \tag{1}$$

where, $T(\Delta l, \Omega_i)$ is the transmittance along segment Δl within the cell, it is calculated as:

$$T(\Delta l, \Omega_i) = e^{-G(\Omega_i) \cdot u_f \cdot \Delta l}, \tag{2}$$

where $G(\Omega_i) = \int_{2\pi} \frac{g_f(\Omega_f)}{2\pi} \left|\Omega_f \cdot \Omega_i\right| d\Omega_f$, which is the leaf projection coefficient for direction Ω_i, Ω_f is the leaf normal orientation vector, $\frac{g_f(\Omega_f)}{2\pi}$ is the normalized leaf angle distribution function, and u_f is the foliage volume density. Thus, the total intercepted radiation within the voxel is $W_{inc}(\Omega_i) \cdot [1 - T(\Delta l, \Omega_i)]$. Scattering events may happen anywhere along the interval $[0, \Delta l]$. To make it simpler, two scattering points (SP) are defined along Δl: one is for upward scattering, and the other one is for downward scattering. The angular distribution of the scattered radiation is determined by the phase function of medium defined in this cell, which is given by:

$$\frac{P(\Omega_i, \Omega_s)}{4\pi} = \frac{\int_{2\pi} \frac{g_f(\Omega_f)}{2\pi} \left|\Omega_f \cdot \Omega_i\right| f\left(\Omega_f, \Omega_i \to \Omega_s\right) d\Omega_f}{\int_{2\pi} \frac{g_f(\Omega_f)}{2\pi} \left|\Omega_f \cdot \Omega_i\right| d\Omega_f}, \tag{3}$$

where, $f\left(\Omega_f, \Omega_i \to \Omega_s\right)$ denotes the leaf scattering phase function. For a bi-Lambertian leaf facet, it is determined by the leaf diffuse reflectance (ρ_{front} and ρ_{back} for front and back reflectance, respectively) and transmittance (τ for both sides).

$$f\left(\Omega_f, \Omega_i \to \Omega_s\right) = \begin{cases} \frac{\rho_{front}}{\pi} s\left(-\Omega_f \cdot \Omega_i\right) + \frac{\tau}{\pi} s\left(\Omega_f \cdot \Omega_i\right), \text{ if } \Omega_f \cdot \Omega_s \geq 0 \\ \frac{\rho_{back}}{\pi} s\left(-\Omega_f \cdot \Omega_i\right) + \frac{\tau}{\pi} s\left(\Omega_f \cdot \Omega_i\right), \text{ if } \Omega_f \cdot \Omega_s < 0 \end{cases}, \quad s(x) = \begin{cases} 1, \text{ if } x > 0 \\ 0, \text{ else} \end{cases}, \tag{4}$$

Thus, the scattered radiation in direction Ω_s is $W_{sca}(\Omega_s) = W_{inc}(\Omega_i) \cdot [1 - T(\Delta l, \Omega_i)] \cdot \frac{P(\Omega_i, \Omega_s)}{4\pi}$. This radiation will be attenuated again during its way from the SP to the border of the cell, which gives the first-order scattering radiation outside the cell $W_{first}(\Omega_s) = W_{sca}(\Omega_s) \cdot T(\Delta l_i, \Omega_s)$. When the radiation transfers from the SP to the outside of the cell, it will be scattered again, the total intercepted radiation for each direction Ω_s is $W_{Int-first}(\Omega_s) = W_{sca}(\Omega_s) \cdot [1 - T(\Delta l_i, \Omega_s)]$. Since each scattering event will produce N new rays, the number of rays becomes unmanageable for higher order scattering events. Therefore, we approximate all higher order (≥2) scatterings with only one scattering point, which is defined as the center of the cell, and then a phase function $\frac{P_h(\Omega_i, \Omega_s)}{4\pi}$, which can describe the angular distribution of all the higher scattering is defined. The detailed derivation of this phase function can be found in the Appendix of Reference [27].

After the calculation of first-order and multi-order within-cell scattering, the scattered radiation may exit the cell from any position of the six faces, and they will enter the next non-empty voxel or

exit the scene. To track the rays which exit the current voxel, each voxel is divided into n^3 sub-voxels, and each voxel face is divided into n^2 sub-faces. All the J rays that exit the same sub-face along the same direction are grouped as an unique exit point (P_E), which is calculated from all the exit points (P_1 , . . . , P_J) weighted by their respective energy (W_j):

$$P_E = \frac{\sum_{j=1}^{J} P_j \cdot W_j}{\sum_{j=1}^{J} W_j}, \tag{5}$$

where, W_j is the flux of the ray exit from P_j. This approach greatly reduces the number of rays.

For any voxel that contains triangles or parts of triangles, if a ray–triangle intersection point occurs, the scattered radiation is determined by the bidirectional reflectance distribution function (BRDF) of the triangle, which is given by $f(\Omega_i \rightarrow \Omega_s) = \frac{\rho}{\pi}$, if the triangle is assumed as a Lambertian surface.

Any ray $W_{i,j}(\Omega_v)$ that exits the top scene through the voxel $V(i, j, k_{top})$ along the direction (Ω_v, $\Delta\Omega_v$) is stored at the top level of the top cells, in order to create a 2D power distribution (i.e., image creation) per DART upward discrete direction. If the sun is the unique source of radiation, the total power per voxel $V(i, j, k_{top})$ is transformed into radiance $L_{i,j}(\Omega_v)$ along direction Ω_v:

$$L_{i,j}(\Omega_v) = \frac{W_{i,j}(\Omega_v)}{\Delta x \cdot \Delta y \cdot \cos\theta_v \cdot \Delta\Omega_v}, \tag{6}$$

where, Δx and Δy are the x and y dimensions of the cell, respectively. θ_v is the zenith angle of observation. $\Delta\Omega_v$ is the associated solid angle along direction Ω_v.

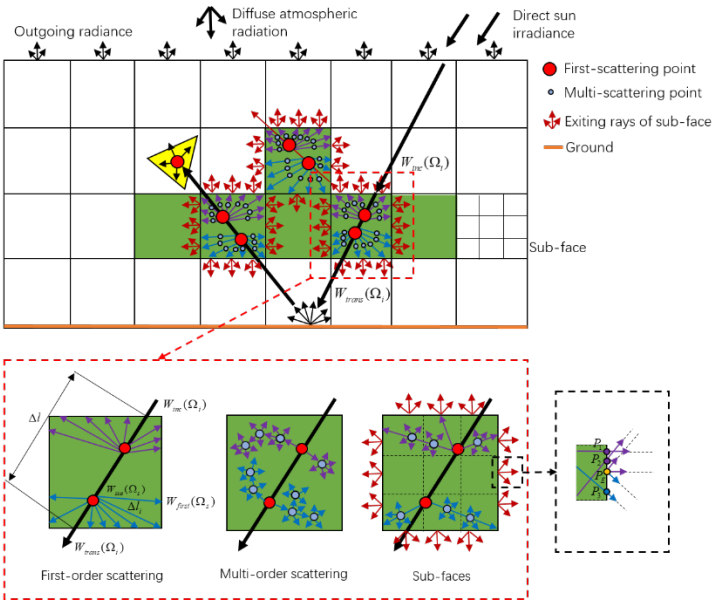


Figure 6. Radiation tracking in non-empty voxels.

2.4. Performance Evaluation Using Three-Dimensional (3D) Realistic Scenes

2.4.1. BASEL City Scene

BASEL is a full 3D city model of the Basel city (Switzerland), which was adapted in the frame of the European Union-funded H2020 project UrbanFluxes (http://urbanfluxes.eu/). It has a size of 7 km × 6.5 km with 1,593,457 triangles. One of the goals of this project was to address the challenge of combating heat in cities by using Earth observation (EO) to identify urban energy budget (UEB) spatial patterns. For that, DART was used to simulate a time series of maps of radiative budget with the help

of satellite images and urban geometric databases [3]. The approach required running thousands of DART simulations with changing parameters (e.g., sun direction), which stressed the usefulness of the hybrid algorithm to reduce computation time and memory. Since the city scene mainly consists of houses, which are represented with triangles, there are large empty spaces between houses and inside the houses (Figure 7). This is ideal for testing the acceleration performance of the new scene structuring scheme. The trees included in this scene are represented as a pure turbid medium, which is described with a set of statistical parameters (LAD, LOD, etc.). Thus, the dimensions of voxels will influence the shapes of the simulated trees. For example, if voxels are too large (e.g., 10 m), trees may be only represented by a single voxel or totally removed from the scene if this tree only occupies less than 20% of a voxel. To get a reasonable representation of trees, the dimensions of a voxel should be relatively small (e.g., 0.5 m). However, the realistic description of scene elements (i.e., very small voxels) and computation time are to some extent contradictory. As illustrated in Figure 7c–d, a 100 m × 100 m subzone, which is composed of 3236 triangles, was selected to quantify the computational performance of the hybrid algorithm under different resolutions.

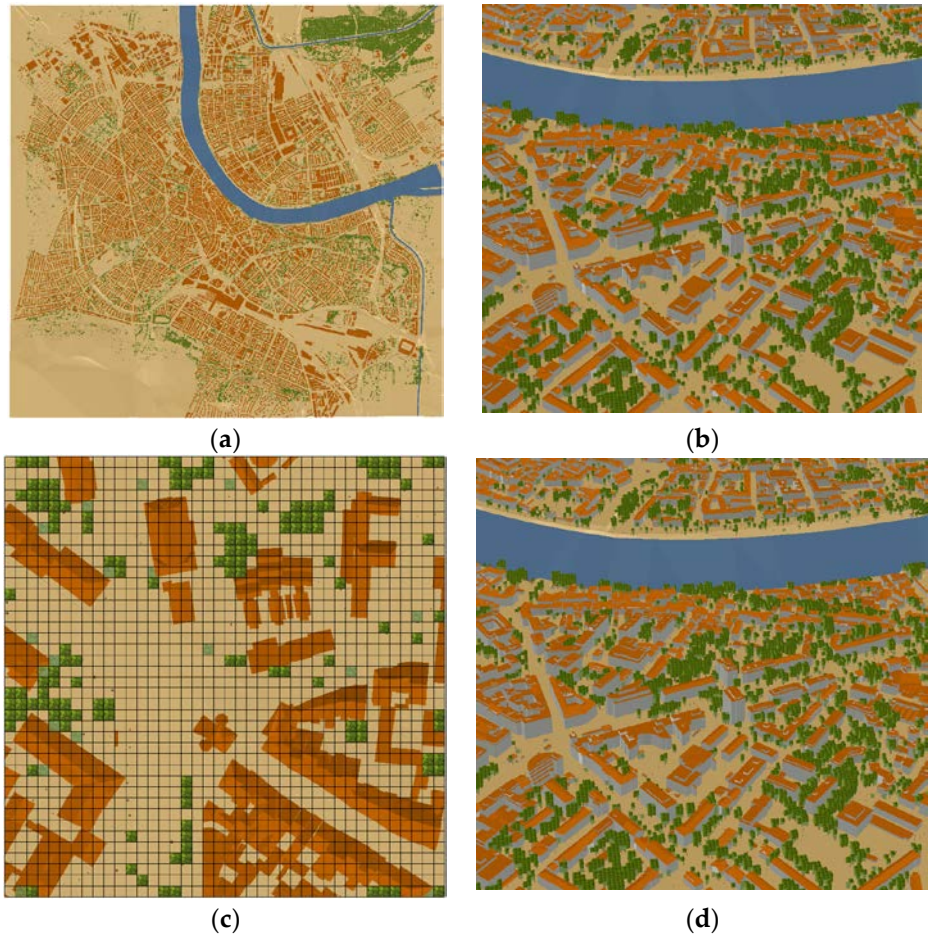
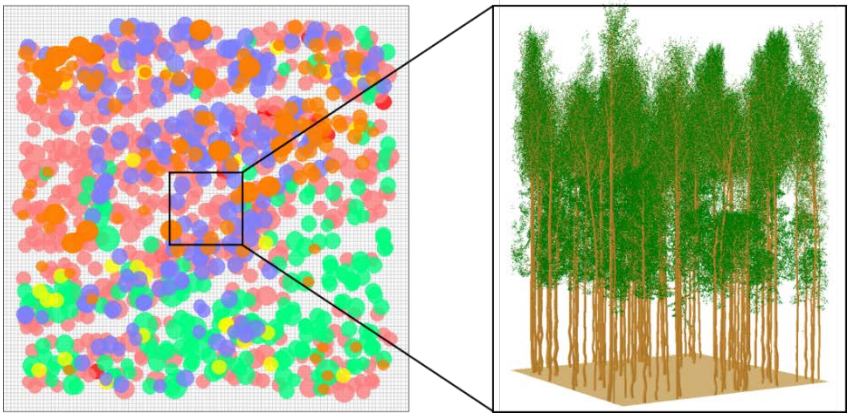


Figure 7. The BASEL scene: (**a**) Overview of the BASEL scene, (**b**) Three-dimensional (3D) view of a part of the scene, (**c**) Voxelization of the scene in 5 m resolution and (**d**) Voxelization of the scene in 10 m resolution.

2.4.2. RAMI Forest Scene

We consider a heterogeneous forest scene, named Järvselja Birch Stand (Summer), that was used by the RAMI experiment (http://rami-benchmark.jrc.ec.europa.eu/HTML/RAMI-IV/RAMI-IV.php), in order to inter-compare radiation transfer models [28]. This scene contains 7 species of different trees. 1059 individual trees of these species are distributed in a 100 m × 100 m plot. Because these trees are

extremely well simulated and due to the limitation of computational resources, a 20 m × 20 m subplot was chosen for testing. Figure 8 shows the 2D and 3D views of this subplot. Since all the elements in this scene (branches and leaves) are converted into triangle mesh, this subplot has more than 51,000,000 triangles. The bidirectional reflectance factor (BRF) image at nadir was simulated under different resolutions (0.1 m, 0.25 m and 0.5 m). The corresponding computation time and memory (RAM) were also recorded for hybrid and uniform structuring schemes.



(**a**) (**b**)

Figure 8. The RAMI scene: (**a**) Two-dimensional (2D) view of the 20 m × 20m subplot, (**b**) 3D view of this subplot.

3. Results

Here, simulation accuracy, computation time and memory usage of the hybrid scene structuring approach and the original uniform approach are compared for the city of BASEL (Figure 9) and the RAMI forest scene (Figure 10). In both cases, there is no bias in accuracy at all spatial resolutions. The averaged root mean square error (RMSE) is 0.0001 and 0.0056 for the Basel city scene and the forest scene, respectively. Conversely to the uniform approach, the accuracy of the hybrid approach is nearly independent of the voxel resolution. For Basel city, simulations are accelerated by 1.4×, 1.7× and 3.7× for 1 m, 0.5 m and 0.25 m voxel resolutions, respectively (Table 1).

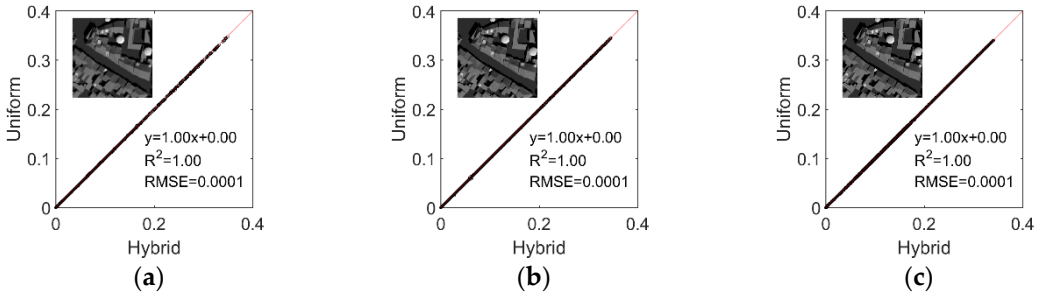


Figure 9. Accuracy comparison of the hybrid and uniform approaches when simulating remote sensing images of Basel at three spatial resolutions: (**a**) 1 m, (**b**) 0.5 m and (**c**) 0.25 m.

For RAMI (Table 2), the hybrid approach reduces the computation time from 67 h to 15 minutes (258.5× speedup) when the resolution is set to 0.5 m. When voxel resolution changes from 0.5 m to 0.1 m, the speedup deceases from 258.5× to 4.9×. As for the memory, the reduction due to the removal of null voxels does not overcome the memory increase caused by within-BVH data structures. Actually, 1,181,824, 193,580 and 36,081 within-voxel BVHs are built for resolution at 0.1 m, 0.25 m and 0.5 m, respectively. It can be seen that using less within-BVHs is better, in terms of memory usage,

than using more BVHs to manage the same number of triangles. It stresses that for triangle-based scenes, the best improvements are obtained for coarser voxel resolutions.

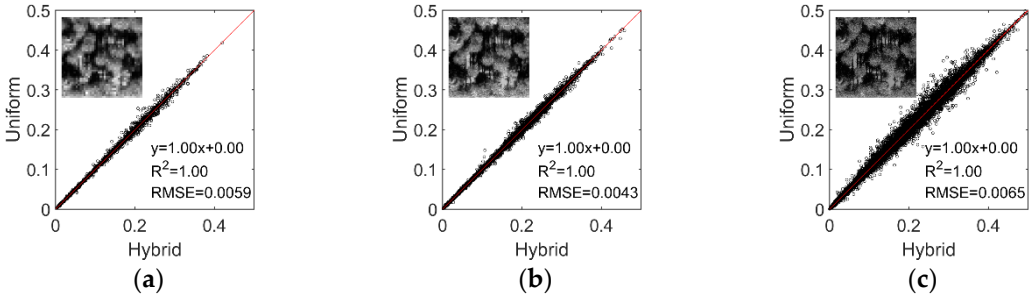


Figure 10. Accuracy comparison of the hybrid and uniform approaches when simulating remote sensing images of the RAMI forest scene at three spatial resolutions: (**a**) 0.5m, (**b**) 0.25m and (**c**) 0.1.

Table 1. Computational resources used by simulating the BASEL scene with different approaches.

BASEL	Null Voxel (%)	Time (s)			Memory (GB)		
Resolution		Uniform	Hybrid	Speedup	Uniform	Hybrid	Decrease
1 m	85.6%	21	15	1.4×	0.18	0.17	6%
0.5 m	91.3%	161	94	1.7×	0.86	0.56	35%
0.25 m	95.2%	2000	540	3.7×	5.4	2.7	50%

Table 2. Computational resources used by simulating the RAMI forest scene.

Forest Scene	Null Voxel (%)	Time (s)			Memory (GB)		
Resolution		Uniform	Hybrid	Speedup	Uniform	Hybrid	Increase
0.5 m	57.0	244,311	945	258.5×	23.5	28.1	16%
0.25 m	69.7	31,622	1221	25.9×	25.6	32.4	21%
0.1 m	84.9	14,277	2911	4.9×	41.2	57.4	28%

4. Discussion

4.1. Accuracy

The accuracy of the proposed method was evaluated by a pixel-wise comparison between images simulated by the hybrid approach and the uniform approach. For the BASEL city scene, these two approaches give the same results for all pixels, while some differences existed for the RAMI forest scene. This incompatible result is mainly caused by the floating-point precision of Embree. In the BASEL city scene, the number of triangles was small, and the size of each triangle was very large, while the RAMI forest scene contained a lot of small triangles (e.g., leaves). Therefore, the floating-point representation of the vertex may cause many differences, compared with the double representation in original DART, which has a significant impact on the ray–triangle intersection, especially when rays traverse through the border of a triangle. For example, some rays may intersect with a triangle in double representation mode, while they may miss the triangle in floating-point mode. This uncertainty becomes larger when many small triangles exist, such as in the RAMI forest scene.

4.2. Memory Usage

One of the advantages of the proposed hybrid scene structuring is to save computation memory by removing null voxels from the 3D scene. This can be confirmed by the BASEL city scene (Table 1), where memory usage reduction due to the removal of the null voxels was observed for all voxel resolutions. When resolution increased, the degree of reduction became larger because of the increased

percentage of null voxels. For the RAMI forest scene (Table 2), the memory increased with the increase of voxel resolution, which can be explained by the percentage of null voxels and the presence of within-BVHs. Compared with the BASEL city scene, the RAMI forest scene had more scene elements (e.g., leaves) and a lower percentage of null voxels, which was 57.0% for 0.5 m resolution, while the BASEL city scene had a percentage of 91.3% for the same resolution. On the other hand, the sizes of facets in the RAMI forest scene were much smaller, and each voxel may contain many triangles, which induced the creation of within-voxel BVHs. The increase of memory used by within-voxel BVHs may overcome the reduction of memory caused by the removal of null voxels.

4.3. Computation Time

In DART, there are mainly two factors that influence the computation time: voxel-level ray tracking and within-voxel ray–triangle intersection. For the BASEL city scene, the number of triangles is not numerous for each voxel, thus the major factor that influenced computation time was the voxel-level ray tracking. Because of the high percentage of null voxels, the hybrid approach can skip the empty space between non-empty voxels during the ray tracing, thus the improvement of computation time was observed (Table 1).

As for the RAMI forest scene, a significant improvement of computation time can be noticed (Figure 10). In this case, the improvement due to removal of null voxels was much weaker than the improvement due to within-voxel BVH. Under coarser resolution (e.g., 0.5 m), there are many triangles in a voxel. The uniform approach tests the intersection for all triangles within a voxel for each incident ray, and then sorts the order according to the distance to the origin of the ray. In hybrid mode, a ray can find the closest intersected triangle directly without sorting, which can significantly accelerate the simulation. With better spatial resolutions, the average number of triangles in each voxel decreases, and the time difference between uniform approach and hybrid approach also decreases. This can be confirmed by the results in Table 2, where the speedup decreased to 4.9× when voxel resolution was equal to 0.1 m.

5. Conclusions

A hybrid scene structuring scheme has been proposed to accelerate the radiative transfer process in complex landscapes. This approach utilizes the BVH to skip large empty areas between voxels and to accelerate ray–triangle intersection inside a voxel. Compared to the uniform approach, the hybrid approach can usually reduce the computation time by 2× ~ 3× in city scenes, and more than 100× in forest scenes. In terms of accuracy, it is worth noting that a hybrid algorithm allows to avoid an important limitation of the Embree ray-tracing kernels library: all geometrical variables are stored with floating-point numbers which implies important limitations in terms of accuracy for remote sensing applications over large landscapes. The hybrid algorithm overcomes these limitations by using a double BVH algorithm for voxels and for triangles within the voxels.

The hybrid algorithm opens new opportunities for using DART to derive very accurate remote sensing products for many scientific domains such as forestry, agriculture, urbanism, etc. For example, it allows one to use DART to simulate a time series of maps for an urban radiative budget. The inversion of satellite images is another major domain of application, because it usually requires the simulation of hundreds of thousands of simulations.

Finally, it is worth noting that a recent DART improvement allows to simulate three crowns that are filled with triangles, which could be more accurate than turbid medium. With the original uniform approach, this kind of simulation could be very slow due to the ray–triangle intersection within each voxel, while the hybrid approach accelerates the process by removing the null voxels and optimizing the ray–triangle intersection, which are noted for the city and forest scene, respectively.

Author Contributions: Conceptualization, J.Q. and J.-P.G.-E.; methodology, J.Q.; software, J.Q.; validation, J.Q.; formal analysis, J.Q.; investigation, J.-P.G.-E. and J.Q.; resources, J.-P.G.-E., D.X.; data curation, J.Q.;

writing—original draft preparation, J.Q.; writing—review and editing, J.Q., T.Y. J.-P.G.-E. and D.X.; visualization, J.Q.; supervision, T.Y. J.-P.G.-E. and D.X.; project administration, X.X.; funding acquisition, D.X. and J.Q.

Funding: This work was supported by "The Fundamental Research Funds for the Central Universities (No. BLX201905)". It was also supported by the NSFC programs (No. 41571341). This research was partially supported by the National Research Foundation (NRF) Singapore through the Singapore-MIT Alliance for Research and Technology's Centre for Environmental Sensing and Modeling (SMART-CENSAM) interdisciplinary research program. It is also partially supported by the Center National d'Etudes Spatiales (CNES) in the frame of TOSCA projects 'Stem-Leaf' and 'Hypertropik'.

Acknowledgments: We would like to thank the anonymous reviewers for their valuable comments. The authors would like to thank Nicolas Lauret, Jordan Guilleux and Eric Chavanon for their great help in compiling the code and the improvement of the DART model.

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

Appendix A

A user-defined geometry that represents the voxel needs to be defined to build the voxel-level BVH. The shape of this voxel geometry is specified through two callback functions: RtcVoxelBoundsFunc and RtcVoxelIntersectionFun. They are registered to Embree by rtcSetGeometryBoundsFunction and rtcSetGeometryIntersectFunction, respectively. The bounds function RtcVoxelBoundsFunc calculates the bounding box of the voxel geometry, and its code is illustrated in Listing A1. The geometryUserPtr, which is a user pointer set by rtcSetGeometryUserData, provides the accessibility to member functions of the geometry (e.g., getLowestCornerX() gets the X coordinate of the lowest corner). The intersection function RtcVoxelIntersectionFun is called to determine the real intersection point when a ray intersects the bounding box of the geometry. This function is implemented with a ray-box intersection algorithm [25], since voxels in DART are simply boxes with the same dimensions.

Listing A1. Sample bounds function code of the voxel geometry.

```
void RtcVoxelBoundsFunc(const struct RTCBoundsFunctionArguments *args){
    const Voxel * voxel = (const Voxel *) args->geometryUserPtr;
    RTCBounds *bounds_o = args->bounds_o;
    bounds_o->lower_x = voxel->getLowestCornerX();
    bounds_o->lower_y = voxel->getLowestCornerY();
    bounds_o->lower_z = voxel->getLowestCornerZ();
    bounds_o->upper_x = voxel->getHighestCornerX();
    bounds_o->upper_y = voxel->getHighestCornerY();
    bounds_o->upper_z = voxel->getHighestCornerZ();
}
```

Appendix B

By registering a call back function RTCTriangleFilterFunc (Listing A2), through the function rtcSetGeometryIntersectFilterFunction provided by Embree, an intersection is first set to "valid" if intersection occurs. Then, if self-intersection occurs, the tag "geomID" is set to "invalid", which implies that the ray will ignore this intersection and continue to find other potential intersections. The geometryUserPtr here is still set by rtcSetGeometryUserData, but it represents a list of triangles. When a ray is initialized, the previous intersected triangle preItsTriangle will be provided, which is used in the RTCTriangleFilterFunc function to determine whether the ray will intersect it again.

Listing A2. Intersection filtering function code.

```
void RTCTriangleFilterFunc(const RTCFilterFunctionNArguments * args){
    vector<Triangle*> * triList = (vector<Triangle *> *) (args->geometryUserPtr);
    RTCRay *ray = (RTCRay*) args->ray;
    int* valid = args->valid;
    RTCHit* hit = (RTCHit*) args->hit;
    IntersectContext* context = (IntersectContext*)args->context;
    Triangle * preItsTriangle = (Triangle *)context->userPtr;
    if((*preItsTriangle) == (*(triList->at(hit->primID)))){
        hit->geomID = RTC_INVALID_GEOMETRY_ID;
        ray->tfar = embree::inf;
        valid[0] = 0;
    }
}
```

References

1. Myneni, R.B.; Ramakrishna, R.; Nemani, R.; Running, S.W. Estimation of global leaf area index and absorbed PAR using radiative transfer models. *IEEE Trans. Geosci. Remote Sens.* **1997**, *35*, 1380–1393. [CrossRef]
2. Sehgal, V.K.; Chakraborty, D.; Sahoo, R.N. Inversion of radiative transfer model for retrieval of wheat biophysical parameters from broadband reflectance measurements. *Inf. Process. Agric.* **2016**, *3*, 107–118. [CrossRef]
3. Landier, L.; Lauret, N.; Yin, T.; Al Bitar, A.; Gastellu-Etchegorry, J.; Feigenwinter, C.; Parlow, E.; Mitraka, Z.; Chrysoulakis, N. Remote Sensing Studies of Urban Canopies: 3D Radiative Transfer Modeling. In *Sustainable Urbanization*; InTech: London, UK, 2016.
4. Myneni, R.B.; Ross, J. *Photon—Vegetation Interactions: Applications in Optical Remote Sensing and Plant Ecology*; Springer Science & Business Media: Berlin, Germany, 2012.
5. Verhoef, W. Light scattering by leaf layers with application to canopy reflectance modeling: The SAIL model. *Remote Sens. Environ.* **1984**, *16*, 125–141. [CrossRef]
6. Gastellu-Etchegorry, J.P.; Martin, E.; Gascon, F. DART: A 3D model for simulating satellite images and studying surface radiation budget. *Int. J. Remote Sens.* **2004**, *25*, 73–96. [CrossRef]
7. Chen, S.; Liu, L.; Zhang, X.; Liu, X.; Chen, X.; Qian, X.; Xu, Y.; Xie, D. Retrieval of the Fraction of Radiation Absorbed by Photosynthetic Components (FAPARgreen) for Forest using a Triple-Source Leaf-Wood-Soil Layer Approach. *Remote Sens.* **2019**, *11*, 2471. [CrossRef]
8. Ben-Arie, J.R.; Hay, G.J.; Powers, R.P.; Castilla, G.; St-Onge, B. Development of a pit filling algorithm for LiDAR canopy height models. *Comput. Geosci.* **2009**, *35*, 1940–1949. [CrossRef]
9. Lefsky, M.A.; Cohen, W.B.; Parker, G.G.; Harding, D.J. Lidar remote sensing for ecosystem studies: Lidar, an emerging remote sensing technology that directly measures the three-dimensional distribution of plant canopies, can accurately estimate vegetation structural attributes and should be of particular interest to forest, landscape, and global ecologists. *AIBS Bull.* **2002**, *52*, 19–30.
10. Banskota, A.; Serbin, S.P.; Wynne, R.H.; Thomas, V.A.; Falkowski, M.J.; Kayastha, N.; Gastellu-Etchegorry, J.P.; Townsend, P.A. An LUT-Based Inversion of DART Model to Estimate Forest LAI from Hyperspectral Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 3147–3160. [CrossRef]
11. Disney, M.I.; Lewis, P.; North, P. Monte Carlo ray tracing in optical canopy reflectance modelling. *Remote Sens. Rev.* **2000**, *18*, 163–196. [CrossRef]
12. McNeill, M.; Shah, B.; Hebert, M.-P.; Lister, P.F.; Grimsdale, R.L. Performance of space subdivision techniques in ray tracing. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 1992; Volume 11, pp. 213–220.
13. Glassner, A.S. Space subdivision for fast ray tracing. *IEEE Comput. Graph. Appl.* **1984**, *4*, 15–24. [CrossRef]

14. Govaerts, Y.M.; Verstraete, M.M. Raytran: A Monte Carlo ray-tracing model to compute light scattering in three-dimensional heterogeneous media. *IEEE Trans. Geosci. Remote Sens.* **1998**, *36*, 493–505. [CrossRef]
15. Gastellu-Etchegorry, J.-P.; Yin, T.; Lauret, N.; Cajgfinger, T.; Gregoire, T.; Grau, E.; Feret, J.-B.; Lopes, M.; Guilleux, J.; Dedieu, G.; et al. Discrete Anisotropic Radiative Transfer (DART 5) for Modeling Airborne and Satellite Spectroradiometer and LIDAR Acquisitions of Natural and Urban Landscapes. *Remote Sens.* **2015**, *7*, 1667–1701. [CrossRef]
16. Qi, J.; Xie, D.; Guo, D.; Yan, G. A Large-Scale Emulation System for Realistic Three-Dimensional (3-D) Forest Simulation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 4834–4843. [CrossRef]
17. Cleary, J.G.; Wyvill, G. Analysis of an algorithm for fast ray tracing using uniform space subdivision. *Vis. Comput.* **1988**, *4*, 65–83. [CrossRef]
18. Sung, K.; Shirley, P. Ray tracing with the BSP tree. In *Graphics Gems III*; Academic Press Professional, Inc.: Cambridge, MA, USA, 1992; pp. 271–274.
19. Lauterbach, C.; Garland, M.; Sengupta, S.; Luebke, D.; Manocha, D. Fast BVH construction on GPUs. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2009; Volume 28, pp. 375–384.
20. Müller, G.; Fellner, D.W. Hybrid scene structuring with application to ray tracing. In *Proceedings of the Proceedings of the International Conference on Visual Computing (ICVC'99)*; 1999; pp. 19–26.
21. Yin, T.; Gastellu-Etchegorry, J.-P.; Lauret, N.; Grau, E.; Rubio, J. A new approach of direction discretization and oversampling for 3D anisotropic radiative transfer modeling. *Remote Sens. Environ.* **2013**, *135*, 213–223. [CrossRef]
22. Grau, E.; Gastellu-Etchegorry, J.-P. Radiative transfer modeling in the Earth–Atmosphere system with DART model. *Remote Sens. Environ.* **2013**, *139*, 149–170. [CrossRef]
23. Akenine-Möller, T. Fast 3D triangle-box overlap testing. In Proceedings of the ACM SIGGRAPH 2005 Courses, Los Angeles, CA, USA, 31 July–4 August 2005; ACM: New York, NY, USA, 2005; p. 8.
24. Hughes, J.F.; Foley, J.D. *Computer Graphics: Principles and Practice*; Pearson Education: London, UK, 2014.
25. Williams, A.; Barrus, S.; Morley, R.K.; Shirley, P. An efficient and robust ray-box intersection algorithm. In Proceedings of the ACM SIGGRAPH 2005 Courses, Los Angeles, CA, USA, 31 July–4 August 2005; ACM: New York, NY, USA, 2005; p. 9.
26. Wald, I.; Woop, S.; Benthin, C.; Johnson, G.S.; Ernst, M. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Trans. Graph.* **2014**, *33*, 143. [CrossRef]
27. Gastellu-Etchegorry, J.-P.; Demarez, V.; Pinel, V.; Zagolski, F. Modeling radiative transfer in heterogeneous 3-D vegetation canopies. *Remote Sens. Environ.* **1996**, *58*, 131–156. [CrossRef]
28. Widlowski, J.-L.; Pinty, B.; Lopatka, M.; Atzberger, C.; Buzica, D.; Chelle, M.; Disney, M.; Gastellu-Etchegorry, J.-P.; Gerboles, M.; Gobron, N.; et al. The fourth radiation transfer model intercomparison (RAMI-IV): Proficiency testing of canopy reflectance models with ISO-13528. *J. Geophys. Res. Atmos.* **2013**, *118*, 6869–6890. [CrossRef]