Journal of *Marine Science and Engineering*

MDPI

Article

Enhanced Multi-Beam Echo Sounder Simulation through Distance-Aided and Height-Aided Sound Ray Marching Algorithms

Jianhua Cheng, Jingyu Ge * and Runze Bai

College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China; chengjianhua@hrbeu.edu.cn (J.C.); brz@hrbeu.edu.cn (R.B.)

* Correspondence: gejingyu@hrbeu.edu.cn; Tel.: +86-1576-554-4275

Abstract: The study proposes two innovative algorithms in the field of multi-beam echo sounder (MBES) simulation: distance-aided sound ray marching (DASRM) and height-aided sound ray marching (HASRM). These algorithms aim to enhance the efficiency and accuracy of MBES simulations, particularly when dealing with long-distance propagation and real-time processing limitations. DASRM addresses issues related to simulation accuracy by efficiently utilizing the KD-tree for spatial indexing and intersection detection instead of the signed distance field (SDF). Building upon the further analysis of DASRM, HASRM is proposed, which improves the search strategy for ray intersections and utilizes a height field pyramid for sampling and retrieval, thereby reducing memory usage while enhancing indexing efficiency. The experimental results demonstrate that both algorithms significantly outperform traditional methods in terms of simulation time, with HASRM exhibiting particular advantages in parallel computing due to its data structure and improved strategies. Additionally, DASRM is well suited for applications requiring complex scene construction, while HASRM proves especially effective in simulating MBES with a focus on underwater terrain due to its effectiveness in handling large incident angles and long-distance propagation.

Keywords: sonar simulation; ray marching; height field; multi-beam echo sounder; sphere tracing

Citation: Cheng, J.; Ge, J.; Bai, R. Enhanced Multi-Beam Echo Sounder Simulation through Distance-Aided and Height-Aided Sound Ray Marching Algorithms. *J. Mar. Sci. Eng.* **2024**, *12*, 913. https://doi.org/10.3390/jmse12060913

Received: 8 April 2024
Revised: 10 May 2024
Accepted: 28 May 2024
Published: 29 May 2024

1. Introduction

The ocean covers approximately 71% of the Earth and significantly influences the climate, the economy, and transportation. Thus, it is crucial for humans to correctly and efficiently understand, study, and develop the ocean. However, due to the different transmission media between the ocean and land regions, signals such as light and electromagnetic waves used on land are rapidly attenuated in water. As a result, underwater tasks typically require the use of acoustic signals, which have a wide range of applications, including underwater communication, location, distance measurement, speed measurement, and imaging.

Acoustic signals have a wide range of applications in underwater localization, acoustic imaging, and topographic surveys [1]. In addition, experiments are necessary for testing and development. However, underwater experiments are limited by financial burdens, time, and manpower costs; unexpected accidents sometimes hinder the progress of projects. The significance of simulation experiments for cost reduction, risk reduction, and development efficiency is self-evident, and more realistic simulations can be applied to monitor system states and reproduce the environment [2].

Simulation is an essential component of the development and improvement of underwater acoustic devices, and algorithms with good simulation capabilities can also be applied to digital twin technology and virtual reality technology, allowing for the effective reduction of experimental costs, improving development efficiency, monitoring the working state of equipment, and simulating and analyzing equipment data.

1.1. Related Works

The primary objective of current research in simulating underwater acoustic equipment is to improve both accuracy and efficiency. The first type of investigation focuses on the physical properties of underwater acoustic signals, such as energy attenuation, reverberation, noise, and refraction by media. By employing computer graphics techniques to simulate underwater acoustic equipment, this line of research aims to improve simulation accuracy and provide more precise experimental methods for corresponding equipment studies. The second type of inquiry concentrates on optimizing algorithm efficiency through computer graphics techniques like rasterization, ray casting, BVH (bounding volume hierarchy), AABB (Axis-Aligned Bounding Box) [3], and ray tracing. This optimization enables a faster simulation of the physical properties of underwater acoustic signals and enhances the overall efficiency of simulating underwater acoustic equipment. Consequently, it establishes a foundation for real-time and efficient simulation algorithm research. The main objective pursued in this article is primarily focused on enhancing simulation efficiency.

Bell and Linnett [4] proposed a ray-tracing-based side-scan sonar simulation algorithm, incorporating acoustic signal propagation. They accounted for the curved propagation path of sound waves influenced by the sound velocity profile and employed a height field ray-casting technique. By employing larger steps outside the bounding box of the terrain height field and conducting fine searching inside it, they effectively reduced the computational load associated with ray tracing due to complex propagation paths. Guériot et al. [5] proposed a tube tracing technique that reduces the number of rays required by utilizing only four rays to achieve tube tracing. Additionally, they separated the acoustic signal propagation and rendering processes, enabling the simulation of multiple sonar devices using a single set of acoustic signal propagation histories. Coiras and Groen [6] investigated the side-looking sonar imaging process under the assumption of constant sound velocity and signal propagation along straight lines. They successfully demonstrated its application through simulation and three-dimensional sonar image reconstruction. Gu et al. [7] used ray-casting to observe the beam emitted by the sonar as a group of straight rays distributed according to the beam shape. By calculating the intersection points with the object's triangles, they simulated the image sonar. Kwak et al. [8] improved Gu et al.'s [7] method and introduced sound attenuation effects to generate grayscale sonar images. Saz et al. [9] introduced an acoustic model based on the assumption that the sound velocity is constant, combining ray tracing and frequency domain methods to generate high-quality sonar images, but with poor real-time performance. Aykin and Negahdaripour [10] used a ray-casting-based method to analyze quadratic surfaces and provide a better ray distribution pattern to address the problem of undersampling and oversampling caused by uniformly distributed rays, but mainly based on the assumption of ray propagation along straight lines. DeMarco et al. [11] combined the Gazebo simulator and ROS to design the FLS simulator [12]. They used ray casting to generate point clouds and then converted them into sonar images, but did not consider the environmental influence on signal propagation paths. Gwon et al. [13] developed the missing SSS module in UWSim, using a simplified Lambertian diffusion model with rayleigh noise and speckle noise, but did not consider the influence of sound velocity on propagation paths. Cerqueira et al. [14,15] combined rasterization and ray tracing to optimize simulated sonar reflection, reducing the ray tracing area required through rasterization. They also used BVH and AABB algorithms to accelerate rendering [16], and employed osgOcean for the construction of underwater scenes, thereby achieving a real-time application of the simulator. However, only sound wave propagation along straight lines was considered. Ding, Rui, and Shiguang Liu et al. [17] proposed a novel method for simulating sound propagation in underwater scenes by incorporating an enhanced ray tracing technique that accounts for the unique characteristics of the underwater environment, along with a threshold-based approach to compute impulse response in the high-frequency domain. This approach efficiently calculates underwater sound propagation while yielding simulation results that closely align with real-world values. The validity of this method was demonstrated through various

experiments conducted in underwater scenes, marking its pioneering contribution as the first tailored sound propagation model specifically designed for virtual reality applications within an aquatic setting.

Comparing the underwater sonar simulation with underwater acoustic communication simulation, it can be observed that due to the necessity of considering the actual propagation path of underwater sound rays, softwares such as WOSS [18] and COMSOL [19] are often used for underwater acoustic channel simulation. Methods like the Bellhop model [20] or finite element analysis are employed to simulate the propagation of underwater acoustic signals. However, since these simulations primarily focus on accuracy rather than real-time performance, they do not meet the requirements for real-time sonar simulations.

Only a few existing methods for real-time sonar simulation take into account the curved propagation of sound waves influenced by the velocity of sound. Among them, references [4,5,17] have proposed simulation methods for acoustic signals along complex curved paths; however, they all have certain limitations. In reference [4], a height field bounding box was utilized to reduce the computational cost of ray tracing, which required representing the scene solely with a height field and limited the types and complexity of simulated objects. This approach is more suitable for scenarios involving exclusively terrain height fields. Additionally, if the bounding box of the height field data in the scene is large, rays still need to perform numerous small step movements and corresponding detections, resulting in significant computational costs for ray tracing. Reference [5] adopted a method that separates rendering from acoustic signal propagation by initially recording the propagation history of acoustic signals and subsequently conducting rendering and simulation based on sonar equipment characteristics. However, this method primarily suits scenarios where simulation conditions are infrequently altered and only changes in sonar device model definitions occur; it is unsuitable for situations requiring modifications in acoustic signal propagation within scenes. Reference [17] employed a threshold method and utilized three distinct sound speed models, namely constant gradient within layers, constant sound speed within layers, and fixed sound speed. The selection of acoustic models with varying levels of precision was based on the variation in the incidence angle of the sound wave. A higher-precision acoustic model was employed when significant changes in angle occurred, while a lower-precision model or straight-line propagation was used for minor angle variations. However, certain assumptions regarding mid–high frequency and shallow water areas were still necessary to enhance efficiency. Moreover, the lack of addressing the fundamental issue of intersection detection for curved rays is evident in the approximations made for different acoustic models. Therefore, in order to enhance the practicality of simulation algorithms and accurately simulate scenarios involving long propagation distances (whether the distance is long or not depends on the actual distribution of sound velocity profiles, typically referring to situations where the actual path of sound rays deviates significantly from a straight line), particularly for MBES devices, it is crucial to explore new simulation methods.

1.2. Contribution

The paper addresses the simulation artifacts and real-time issues associated with sound ray simulation for the MBES over long propagation distances. Initially, widely-used underwater sonar simulation frameworks are presented. Based on the framework and leveraging the distance-aided ray marching algorithm, a novel approach called distance-aided sound ray marching (DASRM) is introduced. This method utilizes the KD-tree instead of the signed distance field (SDF) to mitigate issues related to simulation accuracy and real-time performance. Following this, the paper explores unresolved challenges within DASRM and introduces a height-aided sound ray marching (HASRM) algorithm. HASRM employs a height field pyramid for terrain profile sampling and spatial indexing, effectively reducing memory usage and enhancing indexing efficiency. By focusing on calculating target position depth instead of propagation distance, this algorithm minimizes required iterations, thus improving simulation efficiency. Ultimately, experiments are presented in

this paper comparing traditional intersection test algorithms based on BVH and AABB with DASRM and HASRM, providing a critical analysis of their respective strengths and weaknesses while suggesting potential areas for future improvement.

2. Materials and Methods

2.1. Basic Sonar Simulation Framework

Underwater sonar simulation commonly employs ray theory, which precisely models the acoustic beam as an array of rays with various incident angles. This approach facilitates the accurate simulation of the sonar beam's measurement process. Initially, following standard sonar measurement principles, the local coordinate system (n-frame) is designated as the geographic coordinate system, while the body coordinate system (b-frame) is configured with respect to the vehicle and oriented forward-right-down. Based on these coordinate system definitions, the transformation matrix $\mathbf{C}_b^n$ is calculated to facilitate converting the ray direction $\mathbf{u}_b$ and ray coordinates $\mathbf{w}_b$ from the b-frame to n-frame within the local reference.

$$\begin{aligned} \mathbf{u}_n &= \mathbf{C}_b^n \mathbf{u}_b \\ \mathbf{w}_n &= \mathbf{C}_b^n \mathbf{w}_b \end{aligned} \tag{1}$$

As shown in Figure 1, the basic simulation framework begins by generating multiple rays in the b-frame to simulate the acoustic beam, based on the MBES's beam shape, opening angle, and configured sonar settings. An error is introduced to account for sensor calibration, represented by σ_{sensor}, as well as the installation position of the MBES at p_{mbe} and its associated installation error denoted by $\sigma_{install}$. Both σ_{sensor} and $\sigma_{install}$ primarily affect the beam pointing error, which is essentially similar to the attitude error. Therefore, these two types of errors are combined with the attitude error to calculate the coordinate transformation matrix $\mathbf{C}_b^n$ in order to incorporate corresponding errors. The form of errors can be a constant or a time-varying function with random noise, depending on the characteristics of the simulated device. Subsequently, both ray origin $\mathbf{w}_b$ and direction $\mathbf{u}_b$ are transformed into the local coordinate system while considering the actual attitude $\mathbf{C}_b^n$ and position $p_{vehicle}$ of the vehicle.

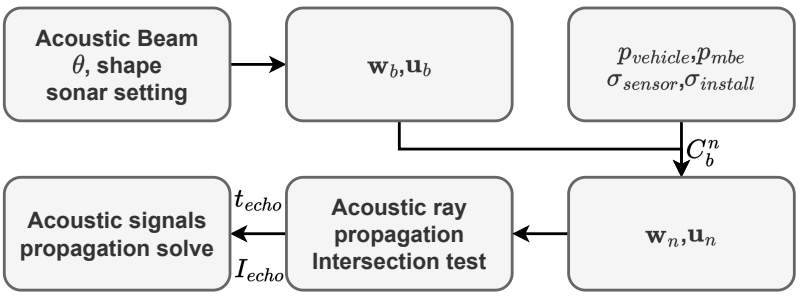

Figure 1. Sonar simulation framework.

In the n-frame, the sound ray tracing method uses the actual sound velocity profile (SVP) svp_{true} to calculate the actual trajectory of a sound ray. Then, by using intersection detection techniques such as ray tracing or ray casting, the intersection points w_{inter} between the ray and the object being tested, along with the propagation time t_{prop} and echo intensity I_{echo}, are calculated.

Measurement outcomes are obtained by performing calculations based on simulated propagation time t_{prop} and echo intensity I_{echo}, using the measurement principles of the simulated sonar equipment.

MBES commonly employs the constant gradient sound ray tracing algorithm to obtain measurement results, which assumes an underwater environment with multiple layers characterized by varying sound velocity and a constant gradient, as shown in Figure 2.

Within these layers, rays propagate along arcs with constant curvature determined by Equation (2).

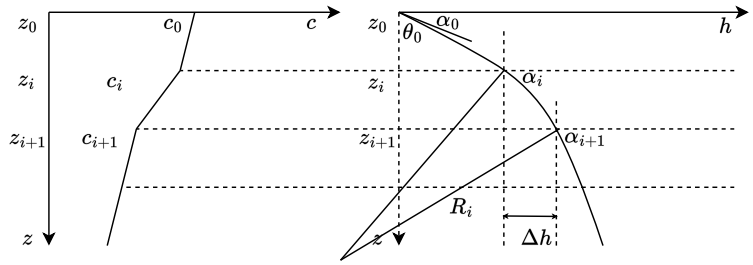

Figure 2. Constant gradient sound ray tracing.

$$\rho = \frac{d\theta}{ds} = \frac{\sin\theta}{c} \cdot \frac{dc}{dz} = \frac{\cos\alpha}{c} \cdot g \tag{2}$$

In this equation, θ represents the incident angle of the sound ray element ds, α denotes its glancing angle, and c signifies the sound velocity at location ds. Equation (3) expresses the curvature radius of the arc.

$$R_i = |(\frac{\cos\alpha_i}{c_i} \cdot g_i)^{-1}| = |\frac{c_i}{g_i \cos\alpha_i}| \tag{3}$$

Within the i-th layer, the horizontal distance increment Δh is given by Equation (4).

$$\Delta h = R_i \mid \sin\alpha_i - \sin\alpha_{i+1} \mid = \frac{c_i}{g_i \cos\alpha_i} \cdot (\sin\alpha_i - \sin\alpha_{i+1}) \tag{4}$$

Finally, the propagation time in the i-th layer can be calculated using Equation (5).

$$t_i = \left| \frac{1}{g_i} \int_{a_i}^{a_{i+1}} \frac{d\alpha}{\cos\alpha} \right| = \left| \frac{1}{g_i} \ln\left(\frac{\tan(\frac{\alpha_{i+1}}{2} + \frac{\pi}{4})}{\tan(\frac{\alpha_i}{2} + \frac{\pi}{4})} \right) \right| \tag{5}$$

According to Equations (4) and (5), the time of propagation t_i and trajectory (Δh, z_i) of the sound wave in each layer can be calculated. By subtracting t_i step by step from the measured propagation time until reaching the N-th layer, the total propagation time of will be exhausted. Finally, based on the remaining propagation time in the N-th layer, we can obtain the measurement.

The simulation of sound rays is reversed compared to the process of the aforementioned sound ray tracing algorithms, which require obtaining the propagation time of sound rays based on their propagation trajectory for simulating sonar measurements.

Existing sonar simulation algorithms, which are commonly developed based on 3D rendering engines, assume the linear propagation of rays and utilize BVH and AABB algorithms to accelerate intersection detection. To accurately simulate sound ray propagation, it is necessary to employ ray tracing algorithms that follow curved paths instead of straight ones. This requires approximating a curved path with short straight segments, resulting in low efficiency and high computational costs. Therefore, developing a novel algorithm for MBES simulation is necessary in order to enhance the real-time performance and accuracy of the simulation.

2.2. Distance-Aided Sound Ray Marching

Distance-aided ray marching algorithms are extensively used for rendering various phenomena such as liquids, deformations, mixtures, and volumetric clouds [21,22]. The algorithm iteratively moves for each ray based on the signed distance to the nearest object surface provided by the SDF [23], repeating the process until an object intersection is encountered. The distance can also be interpreted as the radius of a sphere centered at the current position that is free of other objects, which is why the method is also referred to

as sphere tracing. This technique ensures that rays move safely and guarantees that no potential intersections with other objects are missed, as shown in Figure 3.

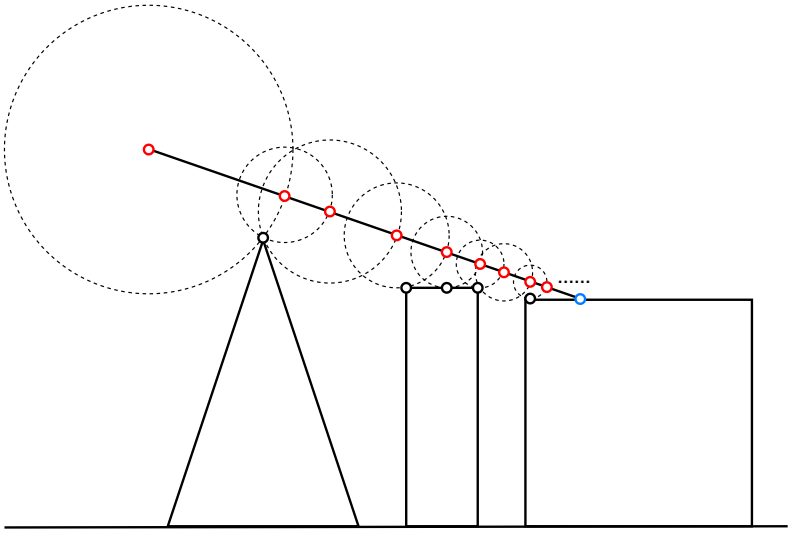

Figure 3. Distance-aided ray marching. The solid black lines in the figure represent objects in the scene, while the dashed black circles represent non-intersecting spherical regions corresponding to distances given by SDF. The red dots represent the positions after each step forward along the ray, and the black dots represent points on the objects closest to the red dots. The blue dots indicate the final intersection point location.

The propagation paths of sound rays are curved due to the influence of the SVP. Constructing an SDF based on straight-line propagation paths leads to inaccuracies. Additionally, MBES simulation scenarios often involve large spatial scales, which require significant storage and construction time. Therefore, constructing an SDF for MBES simulation is not cost-effective. As a result, we replace the SDF with a KD-tree. The KD-tree is a data structure commonly used for spatial indexing in high-dimensional space searches [24], enabling the efficient search for the nearest point G_{near} and the distance D_{near}, thereby reducing overall computational and construction time.

The distance D_{near} derived from the KD-tree, as depicted in Figure 4, does not precisely represent the actual distance from the current position to the underwater terrain due to the spatial resolution of the data, which introduces an associated error e_r. However, considering that curved paths also introduce some error and e_r is a relatively minor positive value, it is acceptable to initially ignore e_r. Subsequently, by establishing appropriate iterative stopping criteria, the influence of e_r can be minimized.

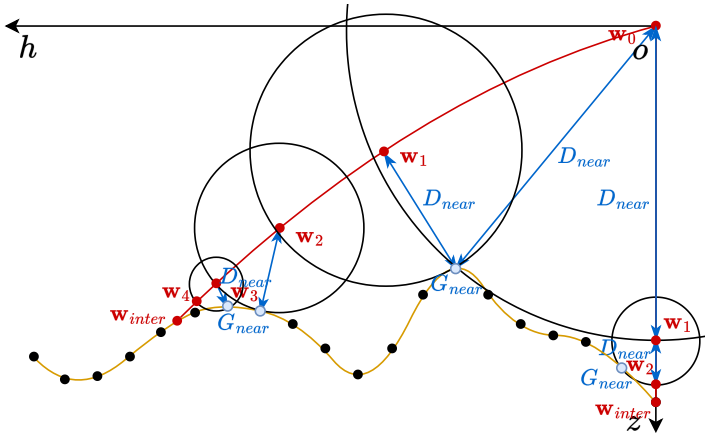

Figure 4. DASRM.

The KD-tree is constructed using the terrain data, as depicted in Figure 4. Within the KD-tree, we can obtain the nearest point G_{near} and its distance D_{near} from the current position $\mathbf{w}_i$. Then, the sound ray moves forward along its current direction $\mathbf{u}_i$. Due to the inherent properties of curved paths, D_{near}'s move along the ray ensures that the subsequent position $\mathbf{w}_{i+1}$ remains within a spherical boundary centered at G_{near} with a radius of D_{near}.

When calculating $\mathbf{w}_{i+1}$ based on a given propagation distance D_{near}, we can first calculate its possible maximum change $\Delta\theta$ in value of θ ($\Delta\theta = D_{near}/R_i$). If $|\Delta\theta|$ is smaller than or equal to the difference $|\theta_{i+1} - \theta_{w_i}|$ in the θ values corresponding to the boundary of the sound velocity layer, it implies that $\mathbf{w}_{i+1}$ lies within the current sound velocity layer. Finally, the corresponding position changes in coordinates $(\Delta h, \Delta z)$ can then be calculated using Equation (6).

$$\begin{aligned} C_{w_{i+1}} &= \frac{C_{w_i}}{\sin\theta_{w_i}} \cdot \sin(\theta_{w_i} + \Delta\theta) \\ \Delta z &= \frac{C_{w_{i+1}} - C_{w_i}}{g_i} \\ \Delta h &= \frac{C_{w_i}}{g_{w_i}\sin\theta_{w_i}} \cdot (\cos\theta_{w_i} - \cos\theta_{w_{i+1}}) \end{aligned} \tag{6}$$

If $\Delta\theta$ exceeds $|\theta_{i+1} - \theta_{w_i}|$, it suggests that $\mathbf{w}_{i+1}$ is located outside the current layer. Subsequently, we set $\Delta\theta$ as $\Delta\theta = \theta_{i+1} - \theta_{w_i}$ and propagate it to the boundary of the current sound velocity layer according to Equation (6). We update the remaining propagation distance as $D_{near} = D_{near} - R_i\Delta\theta$, and continue iterating until the remaining distance D_{near} equals 0. Simultaneously, we compute the propagation time t_{prop} using Equation (5).

When the sound ray approaches the terrain, it is necessary to calculate the intersection point. Since the KD-tree only provides an unsigned distance, which is insufficient to determine if $\mathbf{w}_i$ is inside the terrain, directly applying the ray marching algorithm for intersection point computation is not feasible. Therefore, it is necessary to establish an appropriate distance threshold D_{limit}. When D_{near} is less than or equal to D_{limit}, we can ignore the curved nature of the sound ray and search for surrounding n triangles centered on G_{near}. Then, we apply the Möller–Trumbore ray–triangle intersection algorithm to calculate the potential intersection point $\mathbf{w}_{inter}$ [25].

As n increases, accuracy improves, but computational performance decreases. Therefore, when choosing an appropriate value for n, it is important to balance the desired spatial resolution with the available computational resources. Generally, a value of n greater than or equal to 4 is recommended (where 4 represents the nearest neighbor triangle and its associated triangles). Based on our experience, 6 is a preferable choice. However, if the computational device's performance allows it, using a larger value of n can also be advantageous in finding intersection points more quickly. For larger values of n, considering calculating the AABB for the respective triangles may improve the computational performance.

The value of D_{limit} should be set carefully to strike a balance between algorithm efficiency and accuracy. Setting an excessively large D_{limit} may result in unnecessary intersection tests, which can reduce performance. Conversely, setting an overly small D_{limit} could lead to missed intersections, thereby affecting the accuracy.

When the proximity is sufficient, the ray can be approximated as a straight line that intersects with the triangle at the point w_{inter}. In this scenario, w_{i+1} is located on an arc with the center at w^i, a radius of D_i, and a chord length of l_i. The D_i denotes the distance from point w_i to the nearest vertex G_{near} of the triangle. The distance from w_{i+1} to G_{near} is denoted as D_{i+1}. Since the ray has intersected with the triangle, the chord length l_i resides within the plane of the triangle.

Given that the other vertices of the triangle are not the nearest ones, the distances between them to w_i exceed D_i, hence verifying that l_i is no greater than the maximum edge length l_{tri} of the triangle. Furthermore, since w_{i+1} lies on the arc, it follows that D_{i+1} is no greater than l_i. Consequently, an upper bound for D_{i+1} is established: $D_{i+1} \leq l_{tri}$. To prevent the omission of intersection points, the distance threshold D_{limit} should be equal to l_{tri}.

2.3. Height-Aided Sound Ray Marching

The DASRM algorithm can address the simulation artifacts and real-time processing constraints that arise from long propagation distances; however, it still faces several challenges in practical scenarios due to the unique characteristics of underwater environments.

Firstly, in the simulation of MBES, there is a need to model and perform sonar simulations for large underwater terrains. Although using a KD-tree can decrease memory usage and speed up construction compared to SDF, constructing it still involves significant computational expenses. Moreover, the data structure of the KD-tree is not suitable for parallel processing, which makes GPU acceleration impractical. As a result, the DASRM still faces significant computational cost issues.

In the DASRM, during intersection testing, it is still necessary to employ the traditional ray–triangle intersection method. This requirement necessitates the need for storing the corresponding triangle meshes. Given that terrain scenes in simulations are often large, the required storage space is increased, which significantly diminishes the practicality of the simulation. Consequently, there is a need to refine the relevant data structures and iterative strategies to better accommodate MBES simulation requirements, thereby reducing both spatial and temporal complexity while enabling parallel processing to enhance simulation speed.

According to the constant gradient sound ray tracing theory [26], which assumes that sound rays are only influenced by changes in sound velocity along the z axis, horizontal variations in sound velocity are small and do not affect the trajectory of sound rays. Therefore, the trajectory of the sound ray in the horizontal direction only involves changes in horizontal coordinates, while maintaining its original direction. This means that the intersection point's horizontal coordinate must lie on the projection ray within the xoy plane. As a result, there is no need to search through the entire terrain dataset; instead, one can simply sample the terrain profile at the corresponding horizontal coordinate along the projection line in order to minimize retrieval overhead. Furthermore, when accelerated by GPU, it becomes possible to obtain results directly with a single search operation without requiring multiple iterations. This approach offers advantages over the KD-tree as it saves storage space and accelerates retrieval rate.

In the context of computational simulations, it is noted that sound rays with larger θ angles require a greater number of iterations to converge near the actual intersection point if they continue to move according to D_{near} as depicted in Figure 4 (the sound ray with a larger incident angle took four iterations to reach the vicinity of the intersection, while the vertically downward sound ray only took two iterations), which leads to a deceleration of the simulation.

Given that the depth at the intersection point w_{inter} is denoted as z_{inter}, and assuming that the sound ray propagates nearly in a straight line along this segment, the distance D_{inter} between w_{inter} and w is $\frac{z_{inter}-z_w}{\cos\theta}$. Equation (7) expresses the distance error ε_w between D_{inter} and D_{near}.

$$\varepsilon_w = \frac{z_{inter} - z_w}{\cos\theta} - D_{near} \tag{7}$$

Meanwhile, according to Equation (8), the terrain depth z is expressed as the sum of its mean $\bar{z}$ and random noise ν, which follows a normal distribution.

$$z = \bar{z} + \nu, \nu \sim \ N(0, \sigma_z^2) \tag{8}$$

Underwater terrains are predominantly flat, and these data typically have low resolution. Therefore, when the sound ray is at a considerable distance from the seafloor, the nearest point G_{near} is generally located just below or in close proximity to the w. In such cases, D_{near} can be approximated as $z_{near} - z_w$. By substituting Equation (8) into Equation (7), ε_w can be determined by Equation (9).

$$\varepsilon_w = (\frac{1}{\cos\theta} - 1)((\bar{z} - z_w) + 2\nu) \tag{9}$$

The error in Equation (9) is mainly influenced by three factors:

- The first factor, $\frac{1}{\cos\theta} - 1$, is dependent on θ. The larger the value of θ, the greater the value of ε_w becomes. When θ is zero, the sound ray propagates vertically downward and G_{near} and w_{inter} coincide, resulting in a zero value for ε_w;
- The second factor, represented by $\bar{z} - z_w$, indicates the distance from w to the seafloor. The greater this distance, the higher the value of ε_w. When the sound ray approaches close to the seafloor, this error can be disregarded;
- The third factor is random noise ν, mainly generated by the complex variations of the terrain, which primarily affects intersection testing. Algorithmic optimization is required to prevent the sound rays from missing the actual intersection positions.

The primary limitation for the second part of ε_w lies in the distance between MBES and the seafloor. In order to minimize ε_w and reduce the number of iterations, it is essential to concentrate on mitigating the influence of the first part on ε_w.

$$\varepsilon_w = \frac{z_{inter} - z_w}{\cos\theta} - \frac{z_{near} - z_w}{\cos\theta} = \frac{2\nu}{\cos\theta} \tag{10}$$

According to Equation (10), it is evident that the coefficient preceding z_{near} should be equivalent to that of z_{inter}, which is $\frac{1}{\cos\theta}$. As a result, the first factor of ε_w will naturally decrease to zero, and the constant error in the second factor, $\bar{z} - z_w$, will also be eliminated. This approach ensures that only the stochastic noise error component remains within ε_w.

When the propagation distance is set to $\frac{z_{near} - z_w}{\cos\theta}$, this essentially utilizes z_{near} as the target position depth z_{aim} and calculates the corresponding horizontal coordinate at the depth of z_{aim}, as given in Equation (11). If the maximum depth of the current sound velocity layer is less than z_{aim}, then we propagate the sound ray to the boundary of this layer. We repeat this procedure until $z_w = z_{aim}$.

$$\begin{aligned} C_{w_{i+1}} &= C_{w_i} + g_i \cdot (z_{aim} - z_{w_i}) \\ \theta_{w_{i+1}} &= \arcsin(\frac{C_{w_{i+1}}}{C_{w_i}} \cdot \sin(\theta_{w_i})) \\ \Delta h &= \frac{c_{w_i}}{g_{w_i} \sin\theta_{w_i}} \cdot (\cos\theta_{w_i} - \cos\theta_{w_{i+1}}) \end{aligned} \tag{11}$$

However, the sound ray may be affected by its curvature and the effects of ε_w, which could result in premature intersections with the terrain and computational inaccuracies. Therefore, it is crucial to define a suitable target depth z_{aim} instead of using z_{near} directly, in order to minimize the influence of residual random noise in ε_w and account for the curvature of the sound ray.

According to Snell's law, when the incident angle (θ) is 0°, the sound ray enters the medium vertically without being affected by the velocity of sound and propagates downward in a vertical manner. As θ approaches 90°, there is a tendency for the horizontal propagation of the sound ray. If there are further changes in sound velocity causing bending, gradual propagation towards decreasing depths occurs. Therefore, achieving the theoretical sweeping width corresponding to its opening angle is challenging in MBES due to both bending and attenuation effects on sound rays. As a result, it can be considered that incident angles of sound rays always fall within 0° to 90°. Consequently, the depth z increases monotonically with the horizontal distance h, which can be represented as the continuous function $z_w = F(h_w)$ for the trajectory of the sound ray. Similarly, the terrain profile, derived from sampling the terrain height field, can also be described by a continuous function $d = G(h)$.

Both functions, F and G, are continuous. The intersection point between the sound ray trajectory and the terrain profile corresponds to the first zero value of the continuous function $P(h) = F(h) - G(h)$, as indicated in Equation (12).

$$\begin{aligned} P(h) &= F(h) - G(h) \\ h_{inter} &= P^{-1}(0_{1st}) \end{aligned} \tag{12}$$

However, both the terrain profile and sound ray trajectory are complex curves, and cannot be precisely represented by functional expressions. Consequently, directly calculating the intersection points through zero values is not feasible. It is essential to conduct a detailed analysis of the intersection between the terrain profile and sound ray trajectory to narrow down the potential search range of intersection points. Then, the parametric approach can be utilized to approximate the functions and obtain the intersection points.

The terrain profile data are derived from sampling the height field of the terrain. During the sampling process, various interpolation algorithms are typically applied, such as linear interpolation, bilinear interpolation, or cubic interpolation. Each of these interpolation algorithms follows a specific set of mathematical rules. Generally, the height value at an interpolated point is a linear combination of the height values of adjacent sampling points. As a result, the terrain profile is usually continuous and differentiable.

Therefore, as shown in Figure 5, the intersection between the terrain profile and sound ray can be broadly classified into three scenarios:

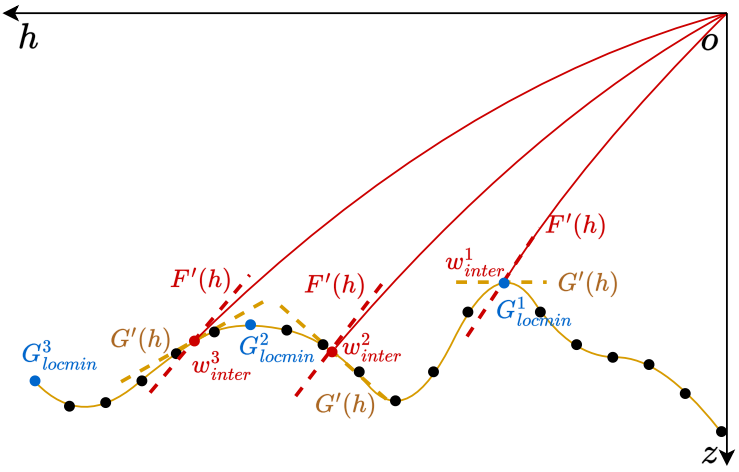

Figure 5. The situations of intersection points.

- $G'(h) = 0$, which is precisely located at the position of a local minimum point G_{locmin};
- $G'(h) < 0$, indicating a decreasing trend in terrain depth, which is opposite in sign to the derivative of sound propagation $F'(h)$;
- $G'(h) > 0$, indicating an upward trend in the depth of the terrain, while $F'(h) > G'(h)$.

When $G'(h) < 0$, the depth of the terrain decreases as h increases. However, it is not possible for the underwater terrain depth to decrease indefinitely; therefore, there will inevitably be a local minimum G_{locmin}. In this case, once we accurately identify G_{locmin}, we can find w_{inter} within a smaller range of horizontal coordinates.

When $G'(h) > 0$, the depth of the terrain increases with an increase in h. However, it is not possible for the depth of the terrain to infinitely increase. Therefore, the depth of the terrain will inevitably reach a maximum value and then decrease to G_{locmin}.

Similarly, when $G'(h) = 0$, regardless of whether the intersection is located at a local maximum or minimum point, it must lie within the interior region of a G_{locmin}. However, due to limitations in our search range, there may be situations where the terrain profile within this range changes monotonically without containing any extrema. Therefore, we

should also consider values at the boundaries of our search range. As long as these values are smaller than those on their inner side, they can also qualify as local minima.

Therefore, the problem can be efficiently detected by dividing it into two phases. Firstly, find the nearest local minimum value G_{locmin} of w_{inter}, which helps narrow down the search range to $(h_w, h_{G_{min}})$. Subsequently, we conduct an accurate search to obtain the w_{inter}.

Before establishing the search range $(h_w, h_{G_{min}})$, we can initially set the search range as $(h_w, n(G(h_w) - z_w))$, where n is determined by the actual performance of the MBES's sweeping width. This ensures that the subsequent position w_{i+1} remains within the search range.

$$G_{min} = \min(\{G_{locmin} | G_{locmin} > z_w\}) \tag{13}$$

According to Equation (13), the target depth z_{aim} should be established as the shallowest depth G_{min} within the local minima G_{locmin} of the function $G(h)$ within the current search range. It is important to note that any minima with depths less than or equal to z_w are excluded from consideration. This avoids scenarios where the sound ray is unable to iteratively move towards the intersection point after its position has been adjusted according to the local minimum. These scenarios, as shown in Figure 6, include instances where h_w equals, exceeds, or falls short of $h_{G_{min}}$.

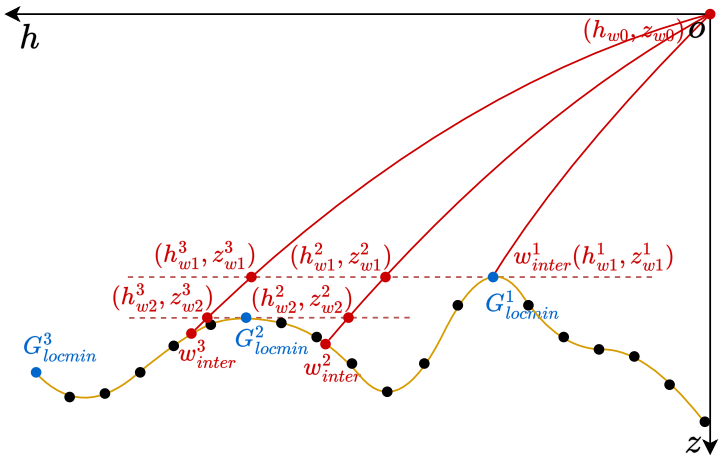

Figure 6. The sound ray movement strategy. Three sound rays initiate their search from (h_{w0}, z_{w0}). After the first search, z_{aim} for these sound rays is determined to be the local minimum G^1_{locmin}. Sound ray 1 then moves directly to the intersection point. For sound rays 2 and 3, since their horizontal positions satisfy $h^3_{w1} > h^2_{w1} > h_{G^1_{locmin}}$, it is clear that G^1_{locmin} is not the target minimum for them, and they require further movement. After setting their target depth to a new local minimum G^2_{locmin} and executing the movement, it is observed that sound ray 2 intersects with the interval $(h_w, h_{G^2_{locmin}})$, where h^2_{w2} is less than the horizontal position of G^2_{locmin}. This successfully obtains the precise search range for the w^2_{inter}.

When h_w equals $h_{G_{min}}$, the sound ray intersects with the terrain. If h_w exceeds $h_{G_{min}}$, the local minimum G_{min} is considered to be outside of the current search range and should be excluded from further consideration. Therefore, only scenarios where h_w is less than $h_{G_{min}}$ need to be analyzed.

In this scenario, since the movement of the sound ray is guided by the local minimum G_{min}, it will not intersect with the terrain before reaching the intersection point. Therefore, it is confirmed that the intersection point h_{inter} lies within the interval $(h_w, h_{G_{min}})$. However, G_{min} may not be the closest local minimum to the actual intersection point. To refine the search range and find a closer minimum to w_{inter}, we should move the sound ray towards deeper local minimum values within this range.

As Equation (13) illustrates, when calculating the movement distance, only local minimum values G_{locmin} with depths exceeding the current sound ray depth z_w should be taken into account because shallower minimum values have already been evaluated. When there are no more G_{locmin} to consider, it indicates that the sound ray has completed its initial search for w_{inter}.

Once the search range for the intersection point is established, to streamline subsequent computational processes, the maximum terrain depth G_{max} is designated as the maximum depth z_{max}, and the current sound ray depth z_w is designated as the minimum depth z_{min}. This depth interval (z_{min}, z_{max}) is then used to represent the search range. If this range is sufficiently narrow, a parametric approach can be applied to compute the location of the intersection point.

In cases where the search range is larger, a threshold z_{lim} for the search range can be established. When the search range exceeds this threshold, the dichotomy method can be used to further narrow down the range. The target depth z_{aim} is set as the midpoint between the current z_{max} and z_{min}, $(z_{max} + z_{min})/2$. After relocating the sound ray, adjustments are made to either z_{max} or z_{min} based on the difference between z_w and the terrain depth $G(h_w)$, as well as between h_w and $h_{G_{min}}$, effectively reducing the search range by half in each iteration.

The threshold z_{lim} is set to ensure the accuracy of the parametric approach in determining the search range. The accuracy of calculating intersection points using parametric approaches mainly depends on the precision of approximating the difference $P(h)$ between terrain profile curve and sound ray curve. The simpler the $P(h)$ curve, naturally, the higher the accuracy of parametric approximation. Generally speaking, since the search distance near intersection points is not too large, sound ray curvature has a relatively small impact on $P(h)$. The complexity of $P(h)$ is primarily influenced by variations in the terrain.

Therefore, setting thresholds is related to the terrain resolution and interpolation method. For example, in this paper, 10 m resolution terrain data were used for simulation with bilinear interpolation for terrain sampling. Therefore, the horizontal range threshold h_{lim} for the search range would be 10 m. Bilinear interpolation uses four points for interpolation, and the distance between the nearest four terrain grid points around one terrain grid point is 10 m. Thus, when the horizontal range limit is set to 10 m, bilinear interpolation can ensure that the sampled values of the terrain profile within this range conform to a relatively simple curve. If other interpolation methods are used, h_{lim} can be redefined based on the window size and calculation method of the interpolation method. Additionally, the depth range threshold z_{lim} would be calculated as $h_{lim}/tan(\theta)$.

As shown in Equation (14), when the size of the search range is relatively small, we can employ a parametric approach to handle the depth difference function $P(h)$ between sound rays and the terrain. Specifically, we can approximate this function, compute relevant coefficients through sampling, and ultimately calculate the zero positions of $P(h)$ to obtain intersection points. This approach takes advantage of the fact that the search range is already sufficiently small, ensuring both accuracy in approximation and efficiency in computation.

$$P(h) = k_1 h + k_2 h^2 + \ldots + b \tag{14}$$

In Equation (14), the coefficients $k_1, k_2, \ldots, b$ have yet to be determined and need to be solved by sampling. Due to the gentle nature of underwater terrain and the low curvature characteristics of the sound ray, a first- or second-order Taylor expansion is sufficient for obtaining accurate estimates of intersection positions. On the other hand, higher-order expansions are more complex when solving for zero values and are not suitable for real-time computation.

By following a three-step strategy that begins with searching for the target location of the minimum value, narrowing down the search range using the dichotomy method, and concluding with precise intersection calculations through the parametric approach, we ensure both precision and real-time capabilities in the simulation. This approach

significantly reduces storage requirements by utilizing only height field data, eliminating the need for additional KD-tree and mesh data storage.

However, when conducting terrain profile sampling based on the height field, a coarse-to-fine strategy is also necessary. This requirement arises from the need to synchronize the sampling of multiple terrain profiles when using parallelized computation techniques. To maintain this synchronization, a uniform number of sampling points is required. Consequently, for a height field with a set resolution, a large search range may result in undersampling and potentially omitting the minimum value points of interest, thus affecting the accuracy.

Therefore, the use of a pyramid data structure is essential. The terrain pyramid is constructed by repeatedly applying a 2×2 window and downsampling the minimum depth value of the terrain with a stride of 2 until a pyramid with n levels of progressively lower resolution terrain data is established. Lower resolution terrain data are utilized for sampling in large search ranges, while higher resolution terrain data are preferred for relatively small search ranges.

As depicted in Figure 7, the dashed line represents the terrain profile achieved through minimum downsampling. While this technique reduces the terrain's resolution, making it appear flatter, all minimum values are maintained, preventing premature intersections. When the search range is narrow enough, employing high-resolution terrain data in conjunction with interpolation algorithms enhances the data's resolution and precision, ensuring accurate intersection calculations and reducing the need for pre-interpolation.

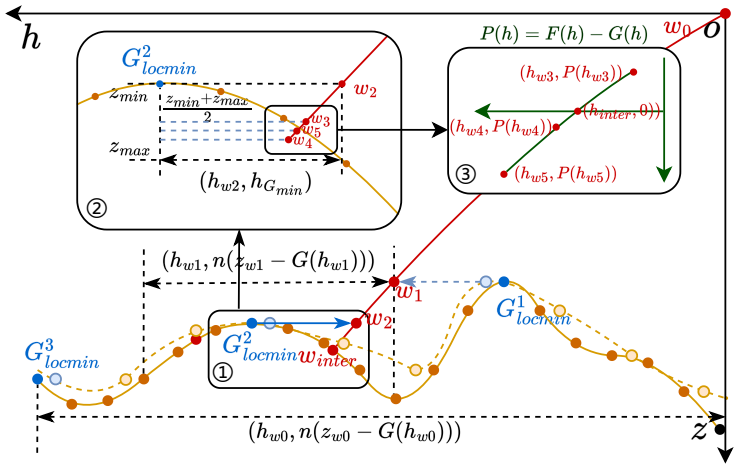

Figure 7. HASRM. The sound ray originates at w_0 and searches for G^1_{locmin} within the low-resolution terrain profile. Once the target depth is determined, the sound ray is directed to that depth. Then, utilizing the high-resolution data, it identifies the subsequent local minimum G^2_{locmin} within a constrained search range and relocates to w_2. Since there are no additional local minima, the interval $(h_{w2}, h_{G_{min}})$ is established to finalize the first phase. In the second phase, the depth range (z_{min}, z_{max}) is defined by a horizontal range, and through the dichotomy method, the sound ray moves to positions w_3, w_4, and w_5, thereby reducing the search range. Finally, by employing a parametric approach and using data samples from w_3, w_4, and w_5, the difference function $P(h)$ is computed to pinpoint the intersection point (w_{inter}).

The basic principle of the HASRM algorithm is illustrated in Figures 7 and 8, which combines all the mentioned intersection calculation strategies and terrain profile sampling methods.

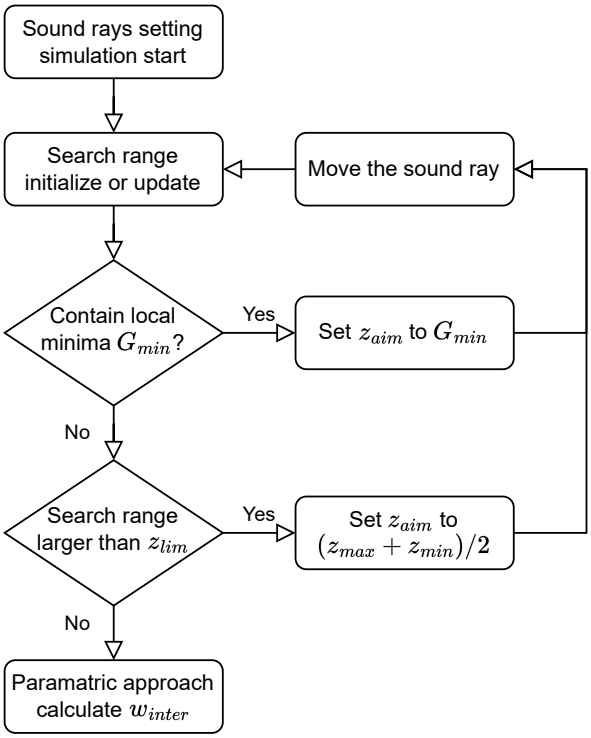

Figure 8. The flowchart of HASRM.

3. Results

The real-time performance of the proposed algorithms was validated by employing a traditional ray-casting algorithm based on BVH and AABB as the comparative algorithm. The rays propagate according to constant gradient sound ray tracing and advance in fixed increments of 20 m at each step. If the AABB distance exceeds D_{limit}, the ray will move to the next position without further AABB testing or ray–triangle intersection testing. Detection is considered successful when the distance between w_{inter} and w_i is smaller than D_{limit}.

To ensure a fair assessment, the BVH, DASRM, and HASRM were implemented in Python 3.9 and executed serially on a CPU. The experiments were conducted on a platform with a 32 GB memory and an Intel(R) Core(TM) i9-13980HX processor.

The study employed three simulated underwater terrains with average depths of 100 m, 500 m, and 1000 m, respectively. The simulation utilized a beam configuration consisting of four rays, with 128 beams oriented at an opening angle of $1^\circ \times 1^\circ$, while the MBES had an opening angle of 120°.

The MBES simulation was conducted with 100 trials for each of these algorithms, and the average simulation times are presented in Table 1.

Table 1. Average simulation times of different algorithms.

	Average Depth		
Method	**100 m**	**500 m**	**1000 m**
BVH + AABB	403 ms	926 ms	1529 ms
DASRM	23 ms	38 ms	41 ms
HASRM	19 ms	19 ms	20 ms

The results presented in Table 1 indicate that DASRM and HASRM have significant advantages over the traditional method in terms of average simulation time. Furthermore, as the average depth increases, the advantage becomes more pronounced. This is because the traditional algorithm based on fixed distance movement requires more iterations to

reach the intersection point with increasing depth. In contrast, DASRM or HASRM can calculate more reasonable propagation distances based on the actual terrain distribution, enabling faster arrival at the intersection point. Due to its superior movement strategy and faster data retrieval speed, HASRM has a greater advantage.

Furthermore, HASRM is particularly well suited for leveraging the parallel computing capabilities of GPUs, thereby significantly enhancing simulation speed. We employed the PyTorch 1.13 framework to effectively parallelize computations involved in both DASRM and HASRM (excluding the KD-tree). By employing an RTX 4070 laptop GPU, we validated the performance of DASRM and HASRM and compared these outcomes with those obtained from their CPU-based implementations.

By configuring the MBES parameters to include 128 beams with the MBES opening angle of 120° and a beam opening angle of $1° \times 1°$, We conducted simulations on beams with varying ray quantities, performing 100 trials for each quantity and averaging the simulation times on a terrain with an average depth of 1000 m. The obtained results are presented in Table 2.

Table 2. Average simulation time.

Ray Quantities for Each Beam	4	16	25	100	400
DASRM(CPU)	41 ms	86 ms	110 ms	301 ms	563 ms
DASRM(GPU)	113 ms	125 ms	137 ms	232 ms	461 ms
HASRM(CPU)	20 ms	35 ms	51 ms	156 ms	410 ms
HASRM(GPU)	40 ms	41 ms	41 ms	73 ms	78 ms

The simulation results presented in Table 2 demonstrate that, for both algorithms, the GPU versions outperform the CPU versions in terms of speed when the number of rays is higher. This improvement is due to the GPU versions' increased utilization of parallel processing capabilities as the number of rays grows. However, the performance enhancement achievable with GPUs is restricted in DASRM due to its reliance on the KD-tree for terrain data retrieval through the CPU before processing by the GPU, which sets it apart from HASRM.

Despite its relatively limited parallel processing capabilities, the CPU still maintains an advantage in executing complex computations. As a result, when the number of rays is low, the CPU versions outperform their GPU counterparts in terms of speed.

The proposed algorithms can simulate long-range detection of MBES thanks to improved algorithm strategies. Assuming 15 beams with an opening angle of 120° and a flat ground surface, the simulation effect is shown in Figure 9. The red solid line represents the central path of the sound beam, light blue represents the edge of the beam, green dashed lines represent rays along each beam's initial straight direction at its starting point, black dots and solid lines represent actual intersections between sound beams and underwater terrain, while pink dots represent measured terrain obtained by a sound beam tracking algorithm based on SVP calculation. The SVP used in the simulation is shown on the left side.

The trajectory of the beam deviates from its initial straight path in response to variations in SVP, as shown in Figure 9. Beyond a depth of 300 m, the path starts bending upwards and then transitions into a downward curve at depths exceeding 600 m, ultimately resulting in straight-line propagation. The actual measurement value is generated by the intersection between the beam edge and the terrain, which is influenced by the conical shape and specific opening angle of the acoustic beam in MBES. Consequently, the depth calculated along the central ray tends to be slightly shallower than the actual depth, with a subtle upward bend observed at the edge of computed terrain.

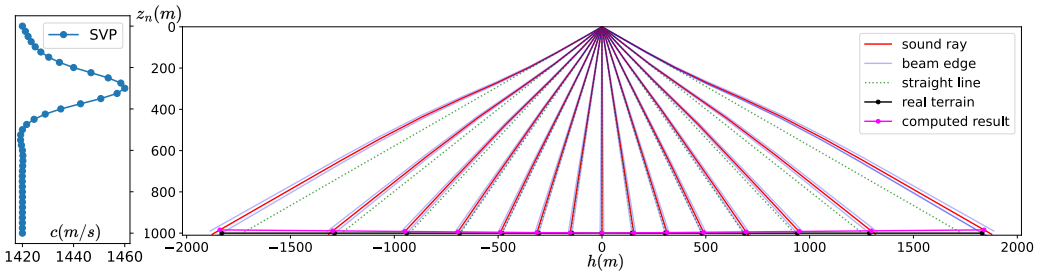

Figure 9. Simulation of sound ray trajectory by HASRM. Due to the change in SVP, the sound ray will undergo distortion. Utilizing HASRM can effectively simulate this bending of the sound ray.

In MBES measurements, inaccuracies in the SVP can result in increased measurement errors. HASRM effectively simulates this phenomenon. Figure 10 illustrates calculations using SVPs with varying gradients: (a–c). In Figure 10a, the employed SVP accurately matches the simulated one, thereby attributing errors primarily to variations in beam opening angle that progressively increase with propagation distance. This error pattern is analogous to that observed in Figure 9. In Figure 10b, the utilization of a higher gradient in sound velocity and average sound velocity leads to increased curvature along the propagation path, resulting in computed terrain depths that exceed the actual depth with an upward curvature. Conversely, Figure 10c exhibits a distinctly contrasting scenario.

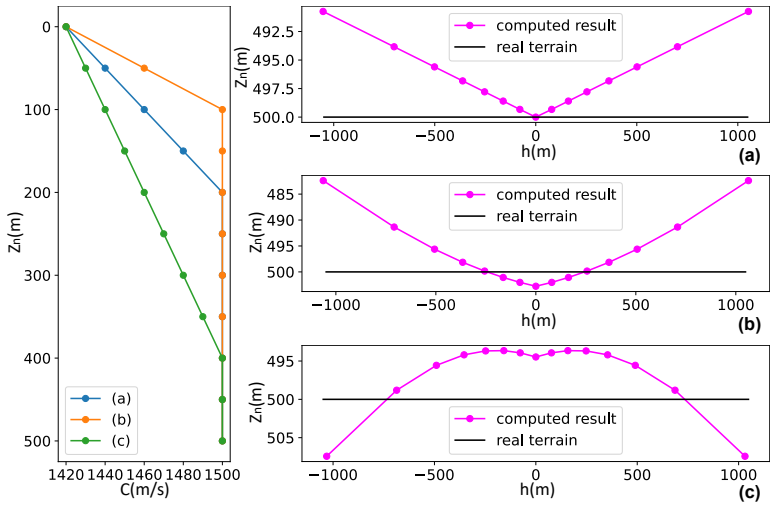

Figure 10. Simulated measurement errors corresponding to different SVPs. (**a**) Simulated measurement error of zero-error SVP. (**b**) Simulated measurement error of greater SVP. (**c**) Simulated measurement error of smaller SVP. The utilization of HASRM and DASRM enables precise simulation of acoustic signal propagation time, facilitating an accurate sound ray tracing algorithm for pinpointing measurement positions. This approach offers a more realistic simulation of MBES measurement errors rather than imposing a fixed error term based on the system's measurement principles.

We maintained consistent simulation conditions and conducted terrain measurements across randomly generated terrain, characterized by an average depth of 500 m and a spatial resolution of 10 m × 10 m. We utilized the SVP illustrated in Figure 10 for simulation purposes, and computed the measurement outcomes for each sound velocity profile presented in Figure 10a–c. The obtained results are depicted in Figure 11. The observed trend of measurement errors displayed in the figure aligns with that demonstrated in Figure 10.

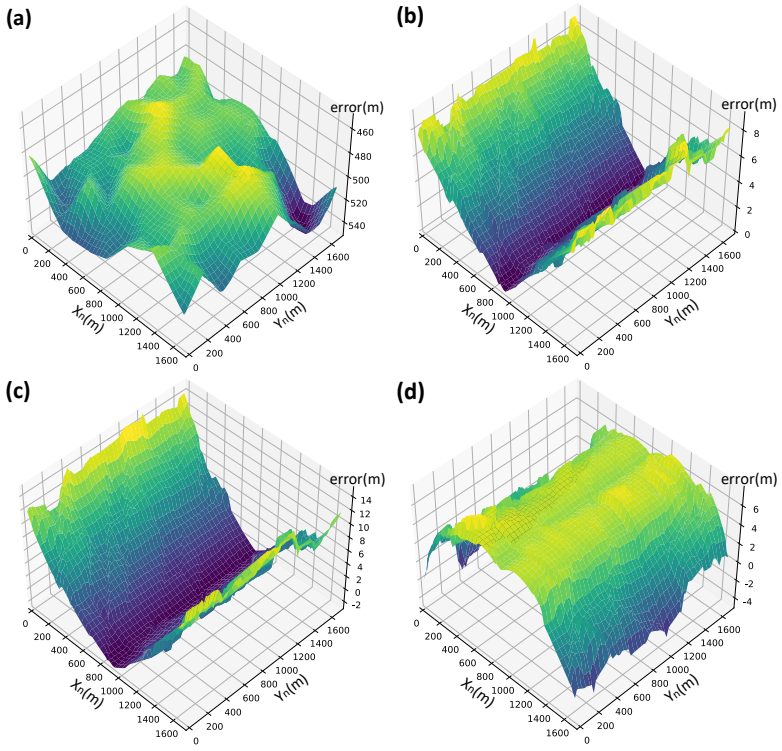

Figure 11. Comparison of simulated measurement errors for different SVPs in MBES. (**a**) Real terrain. (**b**) Simulated measurement error of zero-error SVP. (**c**) Simulated measurement error of greater SVP. (**d**) Simulated measurement error of smaller SVP.

4. Discussion

The experimental results demonstrate that the DASRM and HASRM algorithms introduced in this study outperform traditional methods in terms of real-time performance. They are also capable of accurately simulating the complex propagation paths of acoustic signals, enabling precise simulation of MBES measurement errors. The DASRM algorithm, an adaptive improvement based on sphere tracing, enhances the iterative efficiency of finding intersection points. In addition to the DASRM algorithm, the HASRM algorithm further focuses on underwater terrain measurement scenarios of MBES. It analyzes factors that affect the accuracy of sound ray movement in iteration and optimizes the sound ray simulation process by adopting a progressive intersection detection strategy—from searching for minimum value positions to narrowing down search ranges using dichotomy method, and finally, calculating intersections using parametric approach. This algorithm uses target depth to move sound rays, replacing the nearest neighbor distance in DASRM, thereby reducing the negative impact of large incident angles on simulation speed.

The adoption of the HASRM algorithm ensures a relatively consistent number of iterations for each sound ray, regardless of the incident angle or propagation distance, as shown in Figure 12. Instead, it is solely influenced by the actual distribution of terrain profiles. Consequently, intersection detection can be completed more quickly than with DASRM.

In the context of data structures used for scene construction, DASRM incorporates the KD-tree as a replacement for SDF and maintains the mesh data of the scene for intersection testing. Conversely, the HASRM algorithm utilizes height field pyramids for retrieving and sampling terrain data, effectively replacing the KD-tree and mesh data. This approach not only further reduces memory usage but also enhances retrieval speed, resulting in a significant improvement in parallel computation performance. However, since the HASRM algorithm relies on height field data for scene construction, it is unable to accommodate relatively complex objects due to the inherent limitations of height fields in representing complex geometries. Nevertheless, this restriction does not significantly impair the ef-

fectiveness of the simulation algorithm given that MBES devices are primarily used for underwater terrain measuring.

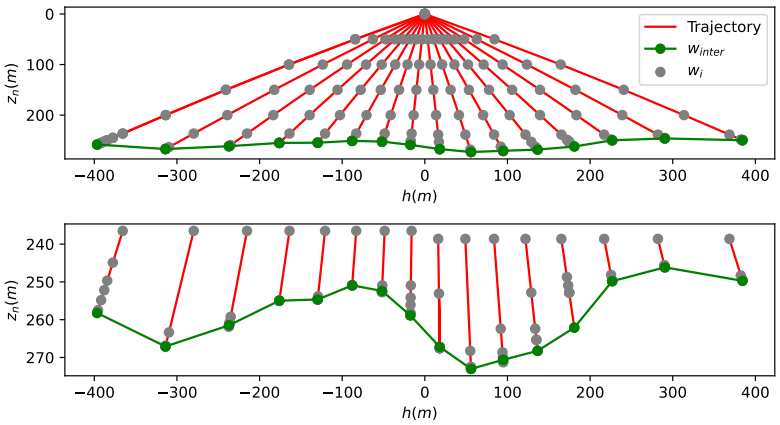

Figure 12. The iteration number of sound rays with different θ in HASRM.

Therefore, when considering the application of the algorithms presented in this paper to other sonar systems or more complex scenarios, DASRM might be a more suitable choice. This is due to DASRM's implementation of data structures, which offer better scene description capabilities. To expand the scope of HASRM, one could explore the construction of multidimensional height fields across various dimensions such as x, y, and z, or consider adopting more sophisticated data structures to augment HASRM's capacity for simulating complex scenes.

Furthermore, there is potential for further refinement and advancement in the algorithms introduced in this paper. For instance, incorporating the spectral characteristics of acoustic signals could result in a more precise simulation of the underwater acoustic field, thereby improving the applicability of these algorithms. Additionally, in order to enhance the precision of the simulation, it is crucial to consider the influence of horizontal velocities on sound rays and revise HASRM's terrain profile sampling strategy, as the computation of sound ray paths currently assumes that rays are only affected by the vertical component of SVP. This may necessitate an independent calculation of the projected trajectory of sound rays within the xoy plane.

Author Contributions: Conceptualization, J.C. and J.G.; methodology, J.G.; software, J.G.; validation, J.G. and R.B.; formal analysis, J.G.; investigation, J.G.; resources, R.B.; data curation, R.B.; writing—original draft preparation, J.G.; writing—review and editing, J.G.; visualization, J.G.; supervision, J.C.; project administration, J.C.; funding acquisition, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 62073093 and 62003108.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: We thank the anonymous reviewers for their comments and suggestions, which greatly improved the manuscript.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MBES	Multi-beam echo sounder
DASRM	Distance-aided sound ray marching
HASRM	Height-aided sound ray marching
SDF	Signed distance field
BVH	Bounding volume hierarchy
AABB	Axis-Aligned Bounding Box
SVP	Sound velocity profile

References

1. Foote, K.G. Underwater acoustic technology: Review of some recent developments. In Proceedings of the OCEANS 2008, Quebec City, QC, Canada, 15–18 September 2008; Volume 2008, pp. 1–6.
2. Etter, P.C. *Underwater Acoustic Modeling and Simulation*; CRC Press: Boca Raton, FL, USA, 2018.
3. Williams, A.; Barrus, S.; Morley, R.K.; Shirley, P. An Efficient and Robust Ray-Box Intersection Algorithm. *J. Graph. Tools* **2011**, *10*, 49–54.
4. Bell, J.M.; Linnett, L. Simulation and analysis of synthetic sidescan sonar images. *IEE Proc. Radar Sonar Navig.* **1997**, *144*, 219–226.
5. Gueriot, D.; Sintes, C.; Garello, R. Sonar data simulation based on tube tracing. In Proceedings of the OCEANS 2007—Europe International Conference, Aberdeen, Scotland, 18–21 June 2007; Volume 1–3, pp. 1508–1513.
6. Coiras, E.; Groen, J. *Simulation and 3D Reconstruction of Side-Looking Sonar Images*; Chapter; IntechOpen: London, UK, 2009; Volume 1.
7. Gu, J.H.; Joe, H.G.; Yu, S.C. Development of image sonar simulator for underwater object recognition. In Proceedings of the 2013 OCEANS, San Diego, CA, USA, 23–27 September 2013; pp. 1–6.
8. Kwak, S.; Ji, Y.; Yamashita, A.; Asama, H. Development of acoustic camera-imaging simulator based on novel model. In Proceedings of the 2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC), Rome, Italy, 10–13 June 2015; pp. 1719–1724.
9. Sac, H.; Leblebicioğlu, K.; Bozdağı Akar, G. 2D high-frequency forward-looking sonar simulator based on continuous surfaces approach. *Turk. J. Electr. Eng. Comput. Sci.* **2015**, *23*, 2289.
10. Aykin, M.D.; Negahdaripour, S. Efficient ray-casting of quadric surfaces for forward-scan sonars. In Proceedings of the OCEANS 2016 MTS/IEEE, Monterey, CA, USA, 19–23 September 2016; pp. 1–7.
11. DeMarco, K.J.; West, M.E.; Howard, A.M. A computationally-efficient 2D imaging sonar model for underwater robotics simulations in Gazebo. In Proceedings of the OCEANS 2015-MTS/IEEE, Washington, DC, USA, 19–22 October 2015; pp. 1–7.
12. Watanabe, T.; Neves, G.; Cerqueira, R.; Trocoli, T.; Reis, M.; Joyeux, S.; Albiez, J. The Rock-Gazebo Integration and a Real-Time AUV Simulation. In Proceedings of the 2015 12TH Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), Uberlandia, Brazil, 29–31 October 2015; pp. 132–138. https://doi.org/10.1109/LARS-SBR.2015.15.
13. Gwon, D.H.; Kim, J.; Kim, M.H.; Park, H.G.; Kim, T.Y.; Kim, A. Development of a side scan sonar module for the underwater simulator. In Proceedings of the 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Jeju, Republic of Korea, 28 June–1 July 2017; pp. 662–665.
14. Cerqueira, R.; Trocoli, T.; Neves, G.; Joyeux, S.; Albiez, J.; Oliveira, L. A novel GPU-based sonar simulator for real-time applications. *Comput. Graph.* **2017**, *68*, 66–76.
15. Cerqueira, R.; Trocoli, T.; Albiez, J.; Oliveira, L. A rasterized ray-tracer pipeline for real-time, multi-device sonar simulation. *Graph. Model.* **2020**, *111*, 101086.
16. Prats, M.; Perez, J.; Javier Fernandez, J.; Sanz, P.J. An Open Source Tool for Simulation and Supervision of Underwater Intervention Missions. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, 7–12 October 2012; pp. 2577–2582.
17. Ding, R.; Liu, S. Underwater sound propagation for virtual environments. *Vis. Comput.* **2021**, *37*, 2797–2807. https://doi.org/10.1007/s00371-021-02175-6.
18. Li, L.; Jin, X.; Lu, C.; Wei, Z.; Li, J. Modelling and Simulation on Acoustic Channel of Underwater Sensor Networks. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 8263600. https://doi.org/10.1155/2021/8263600.
19. Zhao, X.; Yang, P.; Zhao, R.; Han, J. Research on acoustic conduction mechanism of underwater acoustic channel based on metamaterials. *AIP Adv.* **2020**, *10*, 115321. https://doi.org/10.1063/5.0030198.
20. Zhao, R.; Li, M.; Bai, W. Underwater Acoustic Networks Environment Simulation with Combination of BELLHOP and OPNET Modeler. In Proceedings of the OCEANS 2017—Aberdeen, Oceans Aberdeen Conference, Aberdeen, UK, 19–22 June 2017; 2017.
21. Hart, J.C.; DeFanti, T.A. Efficient antialiased rendering of 3-D linear fractals. In Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1 July 1991; pp. 91–100.
22. Hart, J.C. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *Vis. Comput.* **1996**, *12*, 527–545.
23. Jones, M.W.; Baerentzen, J.A.; Sramek, M. 3D distance fields: A survey of techniques and applications. *IEEE Trans. Vis. Comput. Graph.* **2006**, *12*, 581–599.

24. Freidman, J.H.; Bentley, J.L.; Finkel, R.A. An algorithm for finding best matches in logarithmic expected time. *Acm Trans. Math. Softw. (TOMS)* **1977**, *3*, 209–226.
25. Möller, T.; Trumbore, B. Fast, Minimum Storage Ray-Triangle Intersection. *J. Graph. Tools* **1997**, *2*, 21–28. https://doi.org/10.1080/10867651.1997.10487468.
26. Jian, Z.; Qing, Z.; Liang, L.; Chun, C.; Hailun, H. Discussion of Multibeam Sound Velocity Profile Correction. *Hydrogr. Surv. Charting* **2014**, *34*, 62–65+68.