electronics

MDPI

Article

An Efficient SBR Ray-Tracing Implementation for Urban Propagation Modeling Using LBVH Acceleration

Lang Zhan [1,2,*], Lang Yang [1,2], Xin Bian [1,2] and Mingqi Li [1,2,*]

[1] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China; yangl@sari.ac.cn (L.Y.); bianx@sari.ac.cn (X.B.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: zhanl@sari.ac.cn (L.Z.); limq@sari.ac.cn (M.L.)

Abstract

Repeated shooting-and-bouncing-ray (SBR) simulations in urban environments involve numerous ray–scene intersection queries, resulting in substantial computational cost. This paper presents an engineering implementation that combines an extended-Morton-code (EMC)-based linear bounding volume hierarchy (LBVH) with ray classification to accelerate SBR-based urban radio-propagation prediction. The static scene is hierarchically organized, and candidate-node lists are generated for spatial–directional shafts, allowing rays associated with different transmitter–receiver configurations to access relevant LBVH subtrees without repeating classification. The proposed implementation reduces ray-traversal and intersection-testing time, while predicted propagation paths and electromagnetic parameters agree closely with Wireless InSite results.

Keywords: extended Morton code; linear bounding volume hierarchy; ray classification; ray tracing; shooting and bouncing rays; urban radio propagation

1. Introduction

Ray-tracing-based radio-propagation prediction is widely used to identify feasible propagation paths between transmitters and receivers and to estimate electromagnetic-wave coverage in multipath environments, thereby supporting wireless-network design and optimization. Full-wave techniques, including the method of moments, the multilevel fast multipole method (MLFMM), the finite-element method, and the finite-difference time-domain method, can provide highly accurate electromagnetic solutions [1–4]. However, their computational and memory requirements are often prohibitive for electrically large urban environments. Ray tracing therefore offers a practical alternative for large-scale propagation prediction. Time-domain ray-tracing methods that account for lossy materials and multiple propagation effects have also been developed for complex wireless environments [5]. The computational efficiency of ray tracing, however, is strongly influenced by the large number of ray–object intersection tests required, making intersection processing a major performance bottleneck. This bottleneck becomes particularly pronounced when simulating complex scenarios involving emerging technologies like reconfigurable intelligent surfaces (RIS) or anomalous reflectors. Recent studies have noted that using commercial tools like Wireless InSite (v1.4.0) for such complex indoor environments can be excessively time-consuming, often prompting researchers to adopt customized MATLAB (v2023) ray tracers for efficiency [6,7].

In complex environments, each ray generally interacts with only a small fraction of the scene geometry. Acceleration structures are therefore required to avoid unnecessary

Check for updates

Academic Editor: Reza K. Amineh

Received: 9 October 2026
Revised: 23 November 2026
Accepted: 24 November 2026
Published: 27 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

intersection tests. Common approaches include spatial and hierarchical partitioning, dimensionality reduction [8,9], and hardware parallelization [10]. Uniform grids divide the scene into equal-sized cells, whereas octrees recursively subdivide the space using three mutually perpendicular planes [11]. Binary space partitioning trees employ arbitrarily oriented splitting planes [12], while KD-trees use axis-aligned partitions [13]. Although these techniques reduce ray–primitive intersection tests, spatial partitioning may assign the same primitive to multiple regions, resulting in duplicated storage and redundant boundary processing.

Bounding volume hierarchies (BVHs) instead adopt an object-partitioning strategy by grouping geometric primitives within hierarchical bounding boxes. The surface area heuristic (SAH), introduced by Salmon and Goldsmith, has been widely used to improve BVH construction and has also been applied to radio-propagation prediction [14]. Subsequent studies have investigated hybrid BVH–KD-tree structures for accelerating SBR calculations involving multilayer-coated objects [15] and BVH-accelerated bidirectional ray tracing for reducing unnecessary intersection tests [16]. Conventional BVHs are commonly constructed through recursive top-down partitioning, which requires multiple passes over the scene primitives. The resulting parent–child dependency also limits opportunities for efficient parallel construction. Recent SBR acceleration studies have reduced computational cost through adaptive ray launching [17], curved-surface ray tubes for diffraction tracing [18], and BVH-based Embree3 intersection processing [19]. These approaches improve ray generation, ray-tube representation, and geometric intersection testing. However, urban propagation simulation also requires efficient integration with multipath generation and channel metrics such as multipath components (MPCs), power-delay profiles (PDPs), and path loss.

LBVH construction, Morton coding, skip-link-based traversal, and spatial–directional ray classification have been extensively studied in computer graphics. Building on these established techniques, this work integrates an EMC-based LBVH with ray classification in an SBR-based urban radio-propagation solver. The implementation-specific adaptations concern stackless candidate-list construction and candidate-list-driven subtree traversal within the SBR workflow, rather than a new Morton encoding or ray-classification principle. The main contributions are summarized as follows:

1. A stackless adaptation of BVH-based candidate-list construction to a skip-linked EMC-LBVH is implemented. Left-child and skip-link transitions replace the explicit traversal stack during candidate-list construction, while retaining the underlying spatial–directional classification and candidate-selection principles.
2. Candidate-list lookup is connected to stackless subtree traversal through an entry-dependent stopping condition. For each candidate entry c, local traversal stops at $\mathrm{skip}(c)$, and the closest intersection is maintained across all candidate entries. This procedure is incorporated into the processing of reflected and diffracted ray segments in the SBR workflow.
3. The integrated implementation is evaluated in urban propagation scenarios by comparing the computational costs of the tested acceleration configurations and the predicted propagation quantities with Wireless InSite reference results. The evaluation examines computational performance and agreement with the reference predictions within the scope of the investigated scenarios.

2. Integrated Acceleration Framework

This section presents a ray-tracing acceleration framework that integrates LBVH and ray classification. The following subsections detail the core methodology, the skip-linked LBVH structure, and the classification mechanism.

2.1. Adopted EMC-LBVH Foundation

Three-dimensional scenes in ray-tracing simulations can be represented using various geometric primitives, including triangles, rectangles, and general polygons. In this work, we adopt a triangle-mesh representation of urban scenes. To accelerate ray–triangle intersection tests, a BVH organizes triangles into axis-aligned bounding boxes (AABBs) [19], allowing rays to efficiently cull groups of primitives when no intersection with their bounding boxes occurs.

Morton codes [20,21] map spatial coordinates to one-dimensional keys while approximately preserving spatial locality. In the present implementation, the spatial positions of triangle centroids are encoded using Morton codes, and the resulting codes are sorted to group spatially adjacent primitives and support the bottom-up construction of an LBVH [22,23] based on common prefix lengths. The EMC scheme further incorporates primitive size information and flexible axis-bit allocation to improve the spatial grouping and quality of the resulting LBVH.

It should be noted that the present implementation is a triangle-mesh SBR solver: its leaves store triangles, and its primitive test is a ray–triangle intersection test. Although a Morton-code-based LBVH can in principle be constructed from the bounding boxes of other primitive types, this does not mean that the present software directly supports rectangles or general polygons. Other primitive representations, such as rectangles and general polygons, may reduce the number of primitives required to describe certain scenes, but their potential computational advantages depend on the geometry and intersection algorithms employed. Extending the present implementation to directly support non-triangular primitives and evaluating the associated performance trade-offs remain topics for future investigation.

2.2. Stackless Traversal Using Skip Links

For a nearest ray–triangle intersection query, the LBVH is traversed iteratively in a fixed left-first depth-first order. Each node stores a skip link [24] pointing to the first node following its entire subtree in this order. The target may be an internal node, a leaf node, or the sentinel S; it is not necessarily the node's right child. If an internal node's axis-aligned bounding box (AABB) is intersected, traversal continues to its left child. Otherwise, its skip link is followed to bypass the entire subtree. At a leaf node, the enclosed triangle is tested only if the leaf AABB is intersected, after which traversal follows the leaf's skip link regardless of whether a triangle intersection is found. Each leaf contains one triangle in the present implementation. Whenever a valid triangle intersection is found, the closest intersection is updated if its ray parameter t is smaller than the current closest value t_{best}, which is initialized to infinity. A node can also be skipped when its AABB entry distance exceeds t_{best}. Traversal terminates when S is reached, requiring neither an explicit traversal stack nor recursive calls.

Figure 1 illustrates this procedure. Let I_i and L_i denote internal and leaf nodes, respectively. For clarity, consider an example without distance-based pruning, in which the ray intersects the triangles in L_1 and L_5 but misses the AABBs of L_2, I_3, and I_6. The visited-node sequence is

$$I_1 \rightarrow I_4 \rightarrow I_2 \rightarrow L_1 \rightarrow L_2 \rightarrow I_3 \rightarrow I_5 \rightarrow L_5 \rightarrow I_6 \rightarrow S.$$

After L_1 is processed, its skip link leads to L_2. The subsequent miss at I_3 bypasses the subtree containing L_3 and L_4, while the miss at I_6 bypasses the subtree containing L_6, L_7, and L_8. Table 1 details the decisions at each visited node. The nearest valid intersection encountered during traversal is returned.

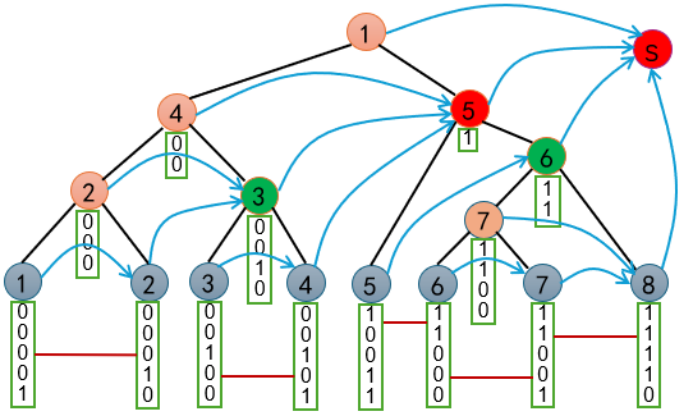

Figure 1. Ordered binary radix tree with skip links. Internal and leaf nodes are denoted by I_i and L_i, respectively. Black edges indicate parent–child relationships, whereas blue arrows indicate skip links leading to the first node after the corresponding subtree in a left-first depth-first traversal. Links leaving the rightmost traversal branch lead to the sentinel S.

Table 1. Step-by-step skip-link traversal for the example in Figure 1.

Node	Intersection Result	Traversal Action	Next Node
I_1	AABB hit	Descend to the left child.	I_4
I_4	AABB hit	Descend to the left child.	I_2
I_2	AABB hit	Descend to the left child.	L_1
L_1	AABB and triangle hit	Update the closest intersection and follow the skip link.	L_2
L_2	AABB miss	Follow the skip link without testing the triangle.	I_3
I_3	AABB miss	Follow the skip link, bypassing the subtree containing L_3 and L_4.	I_5
I_5	AABB hit	Descend to the left child.	L_5
L_5	AABB and triangle hit	Update the closest intersection if nearer, then follow the skip link.	I_6
I_6	AABB miss	Follow the skip link, bypassing the subtree containing L_6, L_7, and L_8.	S
S	Sentinel reached	Terminate traversal and return the closest intersection.	—

Distance-based pruning is omitted only in this illustrative example. In the implementation, a node is also skipped when its AABB entry distance exceeds the current closest intersection distance.

Given n primitives, the skip-link LBVH constructs a binary radix tree that hierarchically encodes the common prefixes of a sorted key set. Each leaf node stores a single key, while each internal node represents the longest common prefix shared by the keys within its range. A tree with n leaves contains $n-1$ internal nodes, each corresponding to a split between two adjacent keys [23]. The construction proceeds by merging key ranges from the leaves upward.

Let $\delta(i,j)$ be the length of the longest common prefix between keys m_i and m_j for $0 \le i < j < n$, and define $\delta(i,j) = -\infty$ otherwise. Furthermore, define $\delta^*(i) = \delta(i, i+1)$. An internal node that spans the range $[i,j]$ is split at position γ such that $\delta(\gamma, \gamma+1) = \delta(i,j)$, producing child ranges $[i,\gamma]$ and $[\gamma+1,j]$.

Starting with a single key, the δ^* values at the left and right boundaries of its range are compared to determine whether the range corresponds to the *left* or *right* child of its parent. The parent node is initially indexed by the split position p; however, directly using p may introduce significant index gaps between sibling nodes. Since the range of

an internal node is fixed, the right boundary of its range indexes the left child, and the left boundary indexes the right child. Accordingly, internal nodes are re-indexed using an index q, which corresponds to the boundary with the smaller δ^* value. By comparing the δ^* values of adjacent keys at node boundaries, the target index of the skip link is obtained as the boundary index plus one. Figure 1 illustrates the resulting node layout for eight keys, and Algorithm 1 presents the corresponding pseudocode.

Algorithm 1 LBVH construction with skip-link installation

Let $\diamond$ denote an invalid entry (e.g., -1), and let $\blacklozenge$ denote the sentinel node.
for each leaf node with index $i \in [1, n]$ **do**
  $range_{\text{left}} \leftarrow i$; $\delta_{\text{left}} \leftarrow \delta^*(range_{\text{left}} - 1)$
  $range_{\text{right}} \leftarrow i$; $\delta_{\text{right}} \leftarrow \delta^*(range_{\text{right}})$
  if $i = n$ **then** $L_{i,\text{skip}} = \blacklozenge$ **else if** $\delta_{\text{right}} < \delta^*(i+1)$ **then** $L_{i,\text{skip}} = I_{i+1}$ **else** $L_{i,\text{skip}} = L_{i+1}$ **end if**
  repeat
    if $\delta_{\text{right}} < \delta_{\text{left}}$ **then**
      $p = range_{\text{left}} - 1$; $store[p]$.right $= range_{\text{right}}$; $range_{\text{left}} = store[p]$.left
      if $range_{\text{left}} = \diamond$ **then return**
      $\delta_{\text{left}} = \delta^*(range_{\text{left}} - 1)$
    else
      $p = range_{\text{right}}$; $store[p]$.left $= range_{\text{left}}$; $range_{\text{right}} = store[p]$.right
      if $range_{\text{right}} = \diamond$ **then return**
      $\delta_{\text{right}} = \delta^*(range_{\text{right}})$
    end if
    $q \leftarrow \begin{cases} range_{\text{left}}, & \text{if } \delta_{\text{right}} \le \delta_{\text{left}} \\ range_{\text{right}}, & \text{otherwise} \end{cases}$
    if $range_{\text{left}} = p$ **then** $I_{q,\text{left}} = L_p$ **else** $I_{q,\text{left}} = I_p$
    if $range_{\text{right}} = n$: $I_{q,\text{skip}} = \blacklozenge$
    else: $r = range_{\text{right}} + 1$;
    $I_{q,\text{skip}} = \begin{cases} I_r, & \text{if } \delta_{\text{right}} < \delta^*(r) \\ L_r, & \text{otherwise} \end{cases}$
    $i = q$
  until $i = 1$
end for

2.3. Frustum Shaft-Based Ray Classification

When testing for environment intersection, a ray examines only objects in its propagation direction, not the whole scene geometry. To avoid unnecessary intersection tests, rays are grouped into convex frustum shafts. We adopt a five-dimensional spatial–directional ray-classification scheme, following the ray-space formulation of Arvo and Kirk [25] and the BVH candidate-list approach of Hendrich et al. [26]. The classification principle is adopted from these studies, whereas its execution is adapted here to the skip-linked EMC-LBVH used in the SBR solver. A ray is classified using three coordinates of its origin and two parameters of its direction. Rays belonging to the same spatial–directional class share a precomputed candidate list of LBVH subtrees. This is a deterministic lookup scheme; it does not require the rays within a class to be traced simultaneously.

The scene AABB is first partitioned into a regular $N_x \times N_y \times N_z$ spatial grid. For a ray origin $\mathbf{o} = (o_x, o_y, o_z)$ inside the scene domain, the corresponding voxel indices are

$$i_a = \left\lfloor \frac{N_a(o_a - b_a^{\min})}{b_a^{\max} - b_a^{\min}} \right\rfloor, \qquad a \in \{x, y, z\}, \tag{1}$$

where $b_a^{\min}$ and $b_a^{\max}$ are the scene bounds. The spatial intervals are treated as half-open, with the outermost upper boundary assigned to the last voxel.

Ray directions are partitioned using cube mapping. For a nonzero direction $\mathbf{d} = (d_x, d_y, d_z)$, the dominant axis is selected as

$$a = \arg\max_{q \in \{x,y,z\}} |d_q|, \tag{2}$$

with ties resolved by a fixed rule. The dominant axis and the sign of d_a select one of the six cube faces. Let (a, b, c) follow the cyclic order (x, y, z), (y, z, x), or (z, x, y). Using a signed local-coordinate convention, the coordinates on the selected face are

$$u = \frac{d_b}{d_a}, \qquad v = \frac{d_c}{d_a}, \qquad u, v \in [-1, 1]. \tag{3}$$

Each face is divided into an $M \times M$ regular grid. The directional-bin indices are

$$\begin{aligned} i_u &= \min\left(M - 1, \left\lfloor \frac{M(u+1)}{2} \right\rfloor\right), \\ i_v &= \min\left(M - 1, \left\lfloor \frac{M(v+1)}{2} \right\rfloor\right). \end{aligned} \tag{4}$$

Thus, each bin has width $2/M$ in the local face coordinates. The minimum operation assigns the upper boundary to the last bin. The subdivision is uniform in (u, v), rather than in solid angle. For example, when $M = 2$, a direction proportional to $(1, 0.3, -0.7)$ belongs to the $+x$ face and gives $(i_u, i_v) = (1, 0)$.

The spatial voxel, cube face, and directional-bin indices jointly identify a ray class. The complete partition therefore contains $6N_xN_yN_zM^2$ classes. Each class corresponds to a convex frustum shaft enclosing the rays with origins in the selected voxel and directions within the selected directional interval. As illustrated in Figure 2, shaft–AABB tests identify the parts of the hierarchy that can be relevant to the frustum shaft. The associated candidate list stores indices of entry nodes in the original LBVH, rather than copies of the referenced geometry.

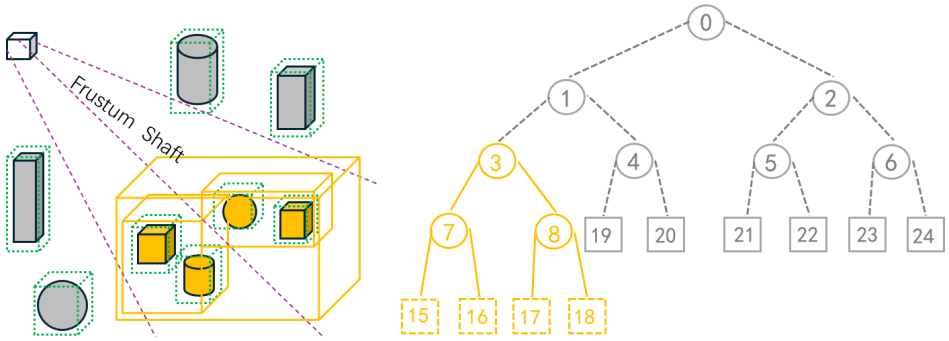

Figure 2. Frustum shaft and candidate list. A selected frustum (**left**) encloses only a subset of scenario geometry, represented by corresponding nodes in the BVH candidate list (**right**). During traversal, nodes and subtrees outside the shaft (dashed/gray) are skipped.

Candidate-list construction uses the left-child and skip-link transitions of the EMC-LBVH instead of an explicit traversal stack. During traversal from the root, a node whose AABB is rejected by the shaft-culling tests is bypassed through its skip link. An overlapping leaf is added to the candidate list. For an overlapping internal node, an AABB-based hit-frequency estimate is used to decide whether to expand the node or retain a coarser subtree entry.

Specifically, N_s sample rays are generated within each shaft, and the same sample set is reused to evaluate its overlapping internal nodes. The hit-frequency estimate for node n is

$$\hat{p}_n = \frac{1}{N_s} \sum_{j=1}^{N_s} \mathbf{1}\{r_j \text{ intersects AABB}(n)\}, \tag{5}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. If $\hat{p}_n \geq p_{\min}$, construction continues to the node's left child and subsequently processes the remaining branches through skip links. Otherwise, the current node is retained as a candidate entry unless it can be replaced by a descendant that still covers all geometry potentially intersected by the shaft within the current subtree. Such a replacement is permitted only when conservative shaft–AABB tests establish that the excluded branches cannot contain relevant geometry. After the candidate entry is recorded, construction proceeds through the original node's skip link. A low estimated hit frequency alone is never used to discard a subtree.

Following the k-ary BVH criterion discussed by Hendrich et al. [26], the initial node-opening threshold is

$$p_{\min,0} = \frac{k-1}{k}. \tag{6}$$

For the binary LBVH used here, $k = 2$, giving $p_{\min,0} = 0.5$. The present implementation uses $N_s = 30$ sample rays per shaft. If the candidate list exceeds the target length of 31 entries, the threshold is progressively increased and the list is reconstructed, with at most ten construction passes. The threshold controls the trade-off between candidate-list length and the depth of the referenced subtrees. It is a traversal-cost parameter, rather than a criterion for accepting or rejecting a physical ray–triangle intersection.

The sample count and opening threshold control the granularity and computational cost of the candidate representation, rather than the acceptance of physical intersections. Increasing the sample count generally improves the statistical stability of the hit-frequency estimate but increases preprocessing work. A lower opening threshold tends to produce deeper and more numerous candidate entries, whereas a higher threshold tends to retain coarser subtree entries. The resulting tracing cost depends on the balance between candidate-list processing and traversal within the referenced subtrees.

Provided that shaft culling is conservative and candidate coverage is preserved, sampling errors can change the selected subtree decomposition but should not, by themselves, remove geometry relevant to the shaft. Ray–AABB and ray–triangle tests are subsequently performed during traversal of the retained subtrees.

During ray tracing, the current ray origin and direction are used to retrieve the corresponding candidate list. Each candidate entry c serves as the root of a local stackless traversal. Its stopping node is defined as

$$s_c = \mathrm{skip}(c). \tag{7}$$

Traversal starts at c and follows the rules described in Section 2.2 until the current node equals s_c. Because s_c is the first node following the entire subtree rooted at c, this condition confines processing to that candidate subtree. Processing then continues with the next entry, while the closest-intersection record is maintained across all entries. The stopping node may be an internal node, a leaf node, or the global sentinel S, depending on the entry's position in the hierarchy.

In Figure 1, the green nodes illustrate candidate entries, whereas the red nodes indicate their corresponding stopping nodes. The same candidate-list lookup and local traversal procedure is applied to reflected and diffracted ray segments using their current origins and directions. For a fixed scene, LBVH, and spatial–directional partition, the candidate lists are constructed during preprocessing and reused across subsequent ray queries. The reported timing comparisons concern the tested acceleration configurations as a whole; they do not isolate the performance effect of stackless traversal.

The candidate lists constitute scene-dependent preprocessing data and need not be rebuilt for every communication link. They can be reused when the scene geometry, the LBVH, the spatial and directional partitions, and the candidate-list construction parameters remain unchanged, provided that the traced rays fall within the precomputed spatial and

directional coverage. For shafts defined by spatial voxels and directional bins, reuse is not intrinsically restricted to a single transmitter–receiver pair. Rays generated for another link can query the same candidate lists through their origin and direction. However, changes to the scene geometry, hierarchy, partitioning, or construction parameters require the affected preprocessing data to be updated.

3. Electromagnetic Modeling

This section outlines the electromagnetic propagation models employed in the ray-tracing simulation, covering the line-of-sight (LoS) component, specular reflections, and wedge-based diffraction.

The electromagnetic calculations use established propagation models rather than a newly developed material model. Each material is specified by its relative permittivity ϵ_r and electrical conductivity σ. Under the $e^{j\omega t}$ time convention, the complex relative permittivity is written as

$$\tilde{\epsilon}_r = \epsilon_r - j\frac{\sigma}{2\pi f \epsilon_0}, \tag{8}$$

where f is the operating frequency and ϵ_0 is the permittivity of free space. The complex relative permittivity is used in the Fresnel reflection coefficients and in the material-dependent reflection terms of the adopted diffraction model. No new electromagnetic interaction model is proposed in this work. The acceleration concerns the organization and traversal of geometric intersection queries, rather than a modified evaluation of material properties.

3.1. Line-of-Sight (LoS) Component

A LoS cluster consists of a single ray whose gain is computed using the Friis transmission equation, with the corresponding complex path coefficient given by

$$\alpha_{\mathrm{LoS}} = \frac{\lambda}{4\pi d}\exp\left(-j\frac{2\pi}{\lambda}d\right), \tag{9}$$

where λ is the wavelength and d is the distance between the transmitter (Tx) and receiver (Rx). The propagation delay is $\tau = d/c$, where c is the speed of light.

3.2. Specularly Reflected Rays

Reflection is the change in direction of an electromagnetic wave upon striking a smooth surface. Specular reflection is modeled using Fresnel reflection coefficients, which depend on the material properties, the incidence angle, and the polarization relative to the local plane of incidence. For a smooth surface, the Fresnel reflection coefficients for parallel ($\parallel$) and perpendicular ($\perp$) polarizations are

$$\Gamma^{\parallel} = \frac{\tilde{\epsilon}_r \sin\theta - \sqrt{\tilde{\epsilon}_r - \cos^2\theta}}{\tilde{\epsilon}_r \sin\theta + \sqrt{\tilde{\epsilon}_r - \cos^2\theta}}, \quad \Gamma^{\perp} = \frac{\sin\theta - \sqrt{\tilde{\epsilon}_r - \cos^2\theta}}{\sin\theta + \sqrt{\tilde{\epsilon}_r - \cos^2\theta}}, \tag{10}$$

where θ is the grazing angle and $\tilde{\epsilon}_r$ is the complex relative permittivity defined in Equation (8). Scattering loss due to surface roughness is calculated to scale smooth-surface reflection results, accounting for energy reduction in the specular reflection direction, as given by

$$\rho_{\mathrm{S}} = \exp\left[-8\left(\frac{\pi\sigma_{\mathrm{h}}\sin\theta}{\lambda}\right)^2\right], \tag{11}$$

with σ_{h} being the standard deviation of the surface height. The power received at the Rx is assumed to be contributed by n_{S} rays, with the path gain coefficient for an n-th-order reflection [27] given by

$$\alpha_{\mathrm{Ref}}^{[n]} = e^{j\zeta}\left[\prod_{i=1}^{n}\rho_{\mathrm{S}}^{[i]}\Gamma^{[i]}\right]\sqrt{\frac{1}{n_{\mathrm{S}}}}\frac{\lambda}{4\pi d_{\mathrm{T}}}, \tag{12}$$

where d_T is the total path distance and $e^{j\zeta}$ represents a phase term.

3.3. Diffraction Rays by Wedges

Although diffraction is limited at millimeter-wave frequencies, it should be considered in obstructed line-of-sight (OLoS) links where the receiver is near a shadow boundary, in order to accurately model the transition from LoS to NLoS conditions. This is achieved by constructing a wedge geometry. According to the uniform theory of diffraction (UTD) [27,28], a diffracted ray is multiplied by a diffraction coefficient D, which for a wedge is given by

$$D = D(L, n; \phi, \phi') = D_1 + D_2 + \Gamma_0 D_3 + \Gamma_n D_4, \tag{13}$$

where Γ_0 and Γ_n are the reflection coefficients for the zero- and n-faces. The components D_l ($l = 1, 2, 3, 4$) are expressed as

$$D_l(L, n; \phi, \phi') = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}} \cot\gamma_l F(2kLn^2 \sin^2\gamma_l), \tag{14}$$

where k is the wavenumber, $L = dd'/(d + d')$ is the distance parameter, $n\pi$ is the exterior wedge angle, $F(\cdot)$ is the transition function defined in [27], and the angles γ_l are defined as

$$\begin{aligned}
\gamma_1 &= [\pi - (\phi - \phi')]/(2n), \\
\gamma_2 &= [\pi + (\phi - \phi')]/(2n), \\
\gamma_3 &= [\pi - (\phi + \phi')]/(2n), \\
\gamma_4 &= [\pi + (\phi + \phi')]/(2n).
\end{aligned}$$

The fundamental idea of UTD is to treat a diffracted ray similarly to a reflected ray in geometrical optics (GO). The path gain coefficient for a diffracted ray can therefore be formulated as

$$\alpha_{\text{Diff}} = \frac{e^{-jkd'}}{d'} D \sqrt{\frac{d'}{dd_T}} e^{-jkd} = \frac{e^{-jkd_T}}{d_T} D \sqrt{\frac{d_T}{d'd}} \tag{15}$$

where $d_T = d + d'$ is the total path distance. The contribution of diffraction is ultimately limited compared to that of LoS and building reflections.

4. Simulations and Discussion

The principal physical parameters and solver-specific settings used in the propagation comparison are summarized in Table 2. Both solvers use a carrier frequency of 2.45 GHz, a transmit power of 30 dBm, and Tx and Rx antenna gains of 1 dBi each. The nominal material permittivities, conductivities, and roughness parameters are aligned, including a ground roughness parameter of 0.2 m. Both configurations allow at most three reflections and one diffraction, with transmission excluded.

These aligned inputs do not imply identical internal electromagnetic implementations. The present implementation uses a single-interface scalar reflection treatment, whereas the WI project specifies a 0.3 m layered-dielectric building material. The receiver-capture criteria also remain solver-specific, as indicated in Table 2. Accordingly, the comparison is conducted under aligned principal physical inputs, without claiming equivalence between the internal implementations of the two solvers.

Acceleration performance is evaluated in two scenes: G1, containing 2793 triangles, and G2, containing 58,841 triangles. Four LBVH-based configurations are compared: a conventional Morton-coded LBVH without ray classification, denoted MC (unclass); a fixed-bit-order Morton-coded LBVH without classification, denoted NFEMC (unclass); an extended-Morton-code-based LBVH without classification, denoted EMC (unclass); and an EMC-based LBVH with ray classification, denoted EMC (class). As shown in

Figure 3, at 50,000 initial rays, the tracing time in G1 decreases from 196.475 s for unclassified EMC to 79.698 s for classified EMC, corresponding to a reduction of 59.44%. For G2, the corresponding times are 373.312 s and 280.600 s, giving a reduction of 24.83%. These percentages are calculated relative to the unclassified EMC configuration at the same initial ray count. They describe the tracing-stage results of the original implementation and do not include the classification preprocessing overhead. The performance advantage becomes more pronounced as the number of rays increases and remains evident in the larger test scene. In both scenes, the choice of Morton encoding has only a minor effect on LBVH construction time. Moreover, ray classification is conducted as a separate preprocessing step subsequent to LBVH construction and independent of the online ray-tracing stage, and therefore does not interfere with the hierarchy construction.

Table 2. Physical parameters and solver-specific settings used in the propagation comparison. Matching nominal inputs does not imply identical internal electromagnetic implementations.

Parameter or Setting	Present Implementation	Wireless InSite
Carrier frequency	2.45 GHz	2.45 GHz
Transmit power	30 dBm	30 dBm
Tx antenna gain	1 dBi	1 dBi
Rx antenna gain	1 dBi	1 dBi
Building relative permittivity	7	7
Building conductivity	0.015 S/m	0.015 S/m
Building roughness parameter	0.001 m	0.001 m
Ground relative permittivity	20	20
Ground conductivity	0.02 S/m	0.02 S/m
Ground roughness parameter	0.2 m	0.2 m
Maximum number of reflections	3	3
Maximum number of diffractions	1	1
Transmission	Not included	Disabled
Building boundary representation	Single-interface scalar reflection treatment	A 0.3 m layered-dielectric setting
Receiver-capture criterion	Distance- and angular-spacing-dependent receiver sphere	Receiver settings of the WI runs
Aggregate received power	Noncoherent summation of retained path powers	Noncoherent summation of exported, retained path powers

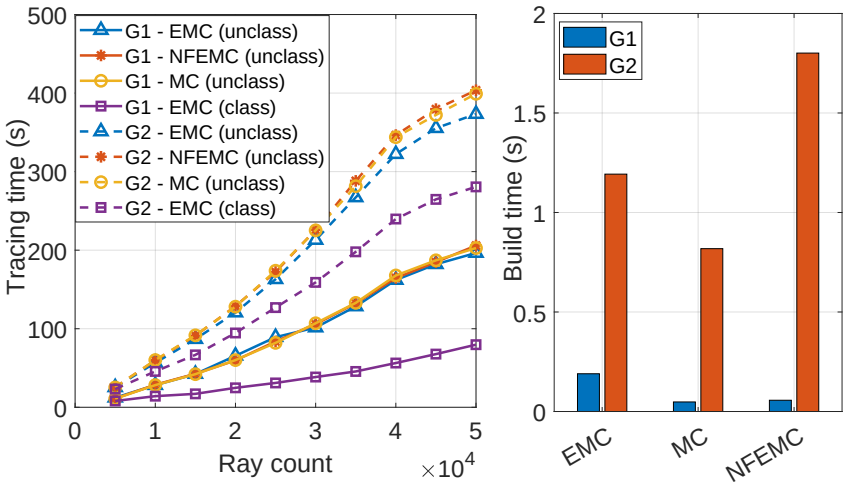

Figure 3. Comparison of ray-tracing time with varying ray counts and LBVH construction time with different Morton encodings in G1 and G2.

The supplementary preprocessing costs are summarized in Table 3. For each scene and encoding method, six observations were obtained across two measurement rounds. The code-generation and LBVH-construction times are reported as medians. Table 3 shows that code generation, sorting, and LBVH construction incur relatively small costs compared with ray-classification preprocessing in the evaluated scenes. Although classification requires additional preprocessing time, it is not necessarily repeated for every link. Under the reuse conditions specified in Section 2.3, the same candidate lists can support multiple link evaluations within the precomputed coverage, allowing the classification cost to be amortized across these evaluations.

Table 3. Code generation, sorting, hierarchy construction, and classification times are reported as medians of six observations. All times are in seconds. A dash indicates that classification is not applied.

Scene	Configuration	Code Generation	Sorting	Construction	Classification
G1	EMC	0.322755	0.052392	0.038220	–
G1	NFEMC	0.170553	0.047847	0.036720	–
G1	MC	0.162752	0.049052	0.035615	–
G1	Classified EMC	0.322755	0.052392	0.038220	123.999
G2	EMC	6.604647	1.101149	0.875731	–
G2	NFEMC	3.169779	1.155948	0.984083	–
G2	MC	3.006915	1.131647	0.825177	–
G2	Classified EMC	6.604647	1.101149	0.875731	103.873

Storage was additionally profiled for the two existing scenes using the instrumented implementation. Both scenes use 75,600 spatial–directional shafts. Storage is evaluated through MATLAB variable-size accounting and reported in MiB, with 1 MiB = 2^{20} bytes. The measurements characterize the explicitly accounted data structures; they are not measurements of peak process memory or total physical RAM usage.

As shown in Table 4, scene-plus-hierarchy storage is 5.389 MiB for G1 and 115.344 MiB for G2. Classification adds 2.473 MiB and 2.118 MiB, corresponding to overheads of 45.90% and 1.84%, respectively. Classification stores hierarchy references rather than duplicated geometry. Its storage depends on shaft count and candidate-list lengths, so a larger scene does not necessarily require more classification storage.

Separately accounted classification-construction auxiliaries occupy approximately 412.153 MiB for G1 and 411.443 MiB for G2. These stage-specific variable-storage sums are excluded from the persistent totals and do not represent peak process memory.

Table 4. Persistent variable storage of the supplementary instrumented implementation. All storage values are in MiB. The total includes common scene data, the LBVH, and classification data but excludes construction auxiliaries, ray/path working sets, and runtime overhead.

Scene	Triangles	Common Data	LBVH	Classification	Total
G1	2793	0.518	4.871	2.473	7.863
G2	58,841	12.135	103.209	2.118	117.462

With one primitive per leaf and a fixed node representation, hierarchy storage grows linearly with primitive count, whereas classification storage depends additionally on shaft count and candidate references. Computational cost also depends on scene geometry, spatial overlap, ray count, and propagation interactions. The two scenes therefore do not establish scalability beyond the evaluated sizes. Scene partitioning remains a possible extension, but its cross-partition propagation handling and computational benefits have not been evaluated. No large-scale performance or peak-memory scalability claim is made.

For the propagation-result comparison with Wireless InSite, the principal physical inputs and power-aggregation rule are aligned as described in Table 2. Nevertheless, differences in the predicted propagation quantities may remain because the two solvers do not use identical numerical implementations. In particular, material-boundary representations, receiver-capture criteria, and path-selection procedures can affect the retained paths and their received powers. These potential sources of discrepancy should not be attributed to the geometric acceleration mechanism without further verification. The comparison assesses agreement under the stated settings rather than exact equivalence between the two solvers.

Next, the integrated implementation is compared with the commercial tool Wireless InSite (WI). The 25 strongest multipath components (MPCs) for 10 transmitter–receiver pairs (Tx–Rx1 –Tx–Rx10) are visualized in Figures 4 and 5. Direct LoS paths are shown in green, reflected paths in cyan-blue, and diffracted paths in magenta. The results indicate that multipath propagation is dominated by direct and reflected components, with relatively few diffracted paths due to higher diffraction loss. Moreover, the spatial distribution of propagation paths closely matches that obtained from WI (Figure 4).

With the transmit power set to 30 dBm, Figure 6 shows, for Tx2–Rx2 as a representative case, a comparison of the power-delay profiles (PDPs) of the 100 strongest MPCs obtained using the proposed method and WI. Both approaches exhibit similar characteristics, including a dominant LoS peak, high-power reflections at short delays, and weaker diffracted components. As shown in Table 5, for an illustrative comparison of stronger components, both solvers' received powers are restricted to values not lower than −70 dBm. Ten pairs with small normalized delay–power distances are selected without reusing either component.

Table 5. Selected delay–power proximity examples with received powers not lower than −70 dBm in both solvers.

τ_{WI} (ns)	τ_{LBVH} (ns)	$\lvert\Delta\tau\rvert$ (ns)	P_{WI} (dBm)	P_{LBVH} (dBm)	$\lvert\Delta P\rvert$ (dB)
276.474	276.386	0.088	−46.6012	−46.5978	0.0034
279.781	279.661	0.120	−57.9061	−57.8852	0.0209
286.255	286.129	0.126	−61.5540	−61.6282	0.0742
279.107	279.468	0.361	−55.5043	−55.7386	0.2343
279.775	279.431	0.344	−63.3955	−63.9672	0.5717
278.431	277.773	0.658	−65.4603	−65.8615	0.4012
278.796	279.560	0.764	−58.3733	−58.1833	0.1900
279.130	280.128	0.998	−59.9060	−59.8248	0.0812
292.614	291.692	0.922	−67.3634	−67.9947	0.6313
279.778	279.283	0.495	−66.3532	−65.2716	1.0816

For these ten selected pairs, the mean absolute time of arrival (TOA) and received-power differences are approximately 0.488 ns and 0.3290 dB, respectively. The maximum differences are 0.998 ns and 1.0816 dB. These examples illustrate local numerical agreement among relatively strong delay–power components.

Figure 7 presents the path-loss values for 10 Tx–Rx pairs obtained using the two methods. The average path-loss difference over the ten Tx–Rx links is approximately 1.03 dB under the considered simulation settings, indicating close agreement with the reference tool.

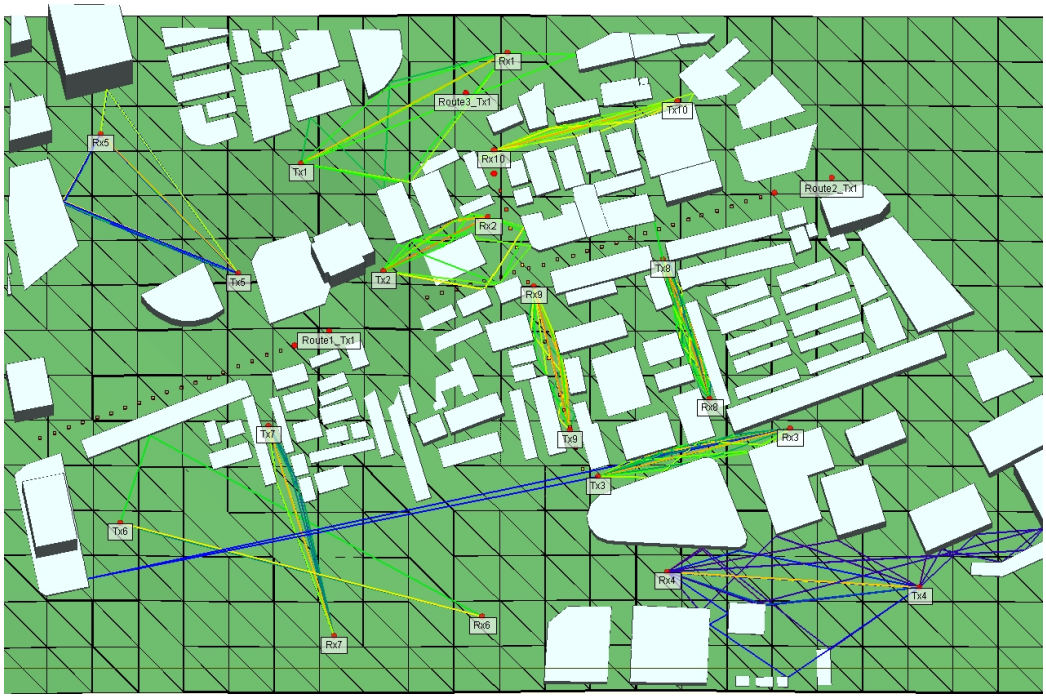

Figure 4. Multipath propagation predicted by Wireless InSite for the urban scenario. Direct, reflected, and diffracted paths are shown together with the transmitter (Tx) and receiver (Rx).



Figure 5. Multipath propagation predicted by the proposed integrated LBVH and ray-classification implementation for the same urban scenario. Direct, reflected, and diffracted paths are shown together with the transmitter (Tx) and receiver (Rx).

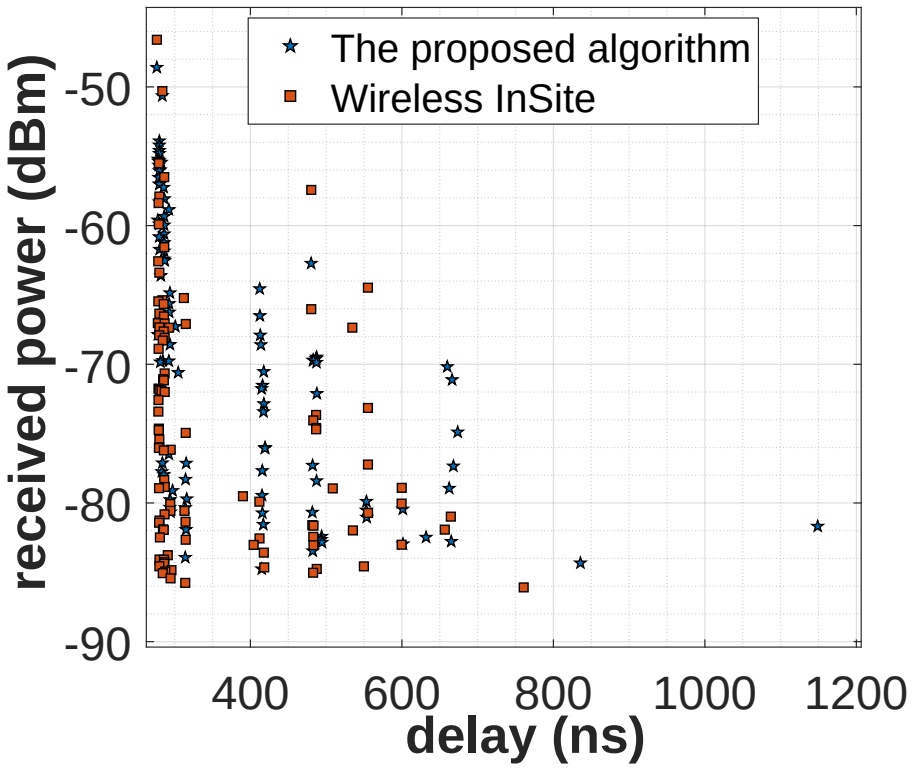

Figure 6. PDP of the 100 strongest MPCs in the Tx2–Rx2 link.

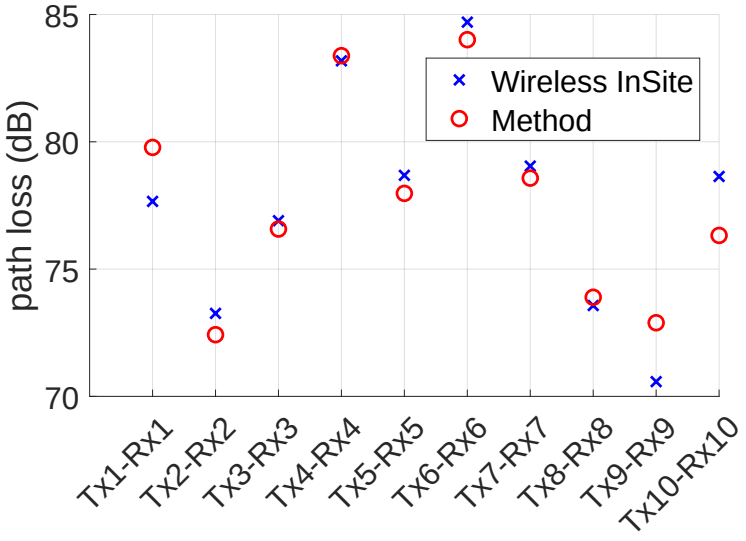

Figure 7. Comparison of location-dependent path loss with that predicted by the WI model.

Figure 8 also compares the path loss along three predefined routes as a function of the Tx–Rx distance. The results closely match those obtained using WI, showing the expected increase in path loss with distance. The average prediction errors for path loss are approximately 2.3 dB (Route 1), 1.51 dB (Route 2), and 1.77 dB (Route 3), confirming good agreement with the commercial tool.

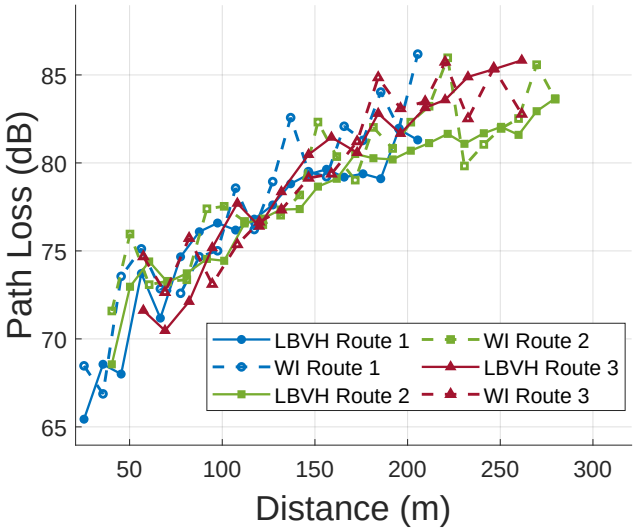

Figure 8. Comparison of distance-dependent path loss with that predicted by the WI model.

5. Conclusions

This paper presents an integrated acceleration framework for SBR-based urban radio-propagation simulation. The framework combines EMC-based LBVH construction, stackless traversal using skip links, and frustum-based ray classification to reduce unnecessary node traversal and ray–triangle intersection tests. Numerical experiments conducted on two urban meshes demonstrate clear improvements in computational efficiency. In addition, the predicted multipath trajectories, power-delay profiles, and path-loss values show close agreement with those obtained using Wireless InSite, confirming the accuracy and practical applicability of the framework for urban propagation prediction. Future work will investigate preprocessing and memory costs, larger and more complex scenes, parallel implementations, and extensions to dynamic environments. Future work may extend the implementation to RIS-assisted propagation scenarios [6], subject to the development and validation of suitable surface-interaction models.

Author Contributions: L.Z.: Conceptualization, Methodology, Software, Validation, Formal analysis, Visualization, Writing—original draft. L.Y.: Software, Data curation, Investigation, Writing—review and editing. X.B.: Methodology, Validation, Resources, Writing—review and editing. M.L.: Supervision, Project administration, Funding acquisition, Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request. The data are not publicly available due to privacy restrictions.

Acknowledgments: This work was supported in part by the Science and Technology Commission Foundation of Shanghai under Grant 25DP1500202 and Grant 24DP1501205.

Conflicts of Interest: The authors declare no conflicts of interest. This work was supported by the Science and Technology Commission Foundation of Shanghai under Grant 25DP1500202 and Grant 24DP1501205. The funding sources had no role in the design of the study; the collection, analysis, or interpretation of data; the writing of the manuscript; or the decision to submit the manuscript for publication.

References

1. Athanaileas, T.E.; Athanasiadou, G.E.; Tsoulos, G.V.; Kaklamani, D.I. Parallel radio-wave propagation modeling with image-based ray tracing techniques. *Parallel Comput.* **2010**, *36*, 679–695. [CrossRef]
2. Tao, S.; Fan, Z.; Liu, W.; Chen, R. Electromagnetic Scattering Analysis of a Conductor Coated by Multilayer Thin Materials. *IEEE Antennas Wirel. Propag. Lett.* **2013**, *12*, 1033–1036. [CrossRef]
3. Liu, J.; Jin, J.M. A highly effective preconditioner for solving the finite element-boundary integral matrix equation of 3-D scattering. *IEEE Trans. Antennas Propag.* **2002**, *50*, 1212–1221. [CrossRef]
4. Fuschini, F.; Falciasecca, G.; Vitucci, E.M.; Dardari, D. Ray tracing propagation modeling for future small-cell and indoor applications: A review of current techniques. *Radio Sci.* **2015**, *50*, 695–716. [CrossRef]
5. Saez de Adana, F.; Gutiérrez, O.; Navarro, M.A.; Mohan, A.S. Efficient time-domain ray-tracing technique for the analysis of ultra-wideband indoor environments including lossy materials and multiple effects. *Int. J. Antennas Propag.* **2009**, *2009*, 390782. [CrossRef]
6. Hao, L.; Vuyyuru, S.K.R.; Tretyakov, S.A.; Salihu, A.; Rupp, M.; Valkonen, R. Modeling RIS from Electromagnetic Principles to Communication Systems—Part II: System-Level Simulation, Ray Tracing, and Measurements. *IEEE Trans. Antennas Propag.* **2025**, *73*, 1756–1767. [CrossRef]
7. Hao, L.; Vuyyuru, S.K.R.; Tretyakov, S.A.; Rupp, M.; Valkonen, R. Analysis of Scalable Electromagnetically-Modeled Anomalous Reflectors Through Ray Tracing and Measurements. *IEEE Open J. Antennas Propag.* **2025**, *6*, 1609–1619. [CrossRef]
8. Lai, Z.; Song, H.; Wang, P.; Mu, H.; Wu, L.; Zhang, J. Implementation and validation of a 2.5D intelligent ray launching algorithm for large urban scenarios. In Proceedings of the 2012 6th European Conference on Antennas and Propagation, Prague, Czech Republic, 26–30 March 2012 ; pp. 2396–2400.
9. Liu, Z.Y.; Guo, L.X.; Zhang, Z.B. An acceleration technique for 2D ray tracing simulation based on diffraction in urban environment. In Proceedings of the 2010 International Symposium on Antennas, Propagation and EM Theory, Guangzhou, China, 29 November–2 December 2010; pp. 493–496.
10. Capozzoli, A.; Kilic, O.; Curcio, C.; Liseno, A. GPU computing in applied electromagnetics: A review. *IEEE Antennas Propag. Mag.* **2017**, *59*, 32–45.
11. Li, J.; Meng, W.; Guo, L.X.; Xi, Y.J. Fast Solution of Scattering from Moving Target by Dynamic-Octree-Based SBR Algorithm. *IEEE Antennas Wirel. Propag. Lett.* **2023**, *22*, 362–366. [CrossRef]
12. Radha, H.; Vetterli, M.; Leonardi, R. Image compression using binary space partitioning trees. *IEEE Trans. Image Process.* **1996**, *5*, 1610–1624. [CrossRef] [PubMed]
13. Wang, H.; Wei, B.; Liu, H.W. An Enhanced KD-Tree-Based Ray Tube Subdivision SBR for Fast RCS Prediction. *IEEE Antennas Wirel. Propag. Lett.* **2024**, *23*, 4228–4232. [CrossRef]
14. Guo, J.; Guo, L.; Liu, Z.; Wang, Y. A Fast Ray Tracing Algorithm for Propagation Prediction in Urban Environments. In Proceedings of the 2024 IEEE International Symposium on Antennas and Propagation, Florence, Italy, 14–19 July 2024; pp. 2185–2186.
15. Meister, D.; Ogaki, S.; Benthin, C.; Doyle, M.J.; Guthe, M.; Bittner, J. A Survey on Bounding Volume Hierarchies for Ray Tracing. *Comput. Graph. Forum* **2021**, *40*, 683–712. [CrossRef]
16. Liu, H.; Wei, B.; Wang, H.; Wei, H. Improved Bidirectional Ray-Tracing SBR Algorithm Based on BVH Acceleration. *IEEE Antennas Wirel. Propag. Lett.* **2024**, *23*, 1839–1843. [CrossRef]
17. Shi, D.; Lv, N.; Wang, N.; Gao, Y. An Improved Shooting and Bouncing Ray Method for Outdoor Wave Propagation Prediction. *Appl. Comput. Electromagn. Soc. J. (ACES)* **2021**, *32*, 581–585.
18. Shi, D.; Lv, N.; Gao, Y. A Diffraction Ray Tracing Method Based on Curved Surface Ray Tube for Complex Environment. *Appl. Comput. Electromagn. Soc. J. (ACES)* **2021**, *32*, 608–613.
19. 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*, 143:1–143:8. [CrossRef]
20. Morton, G.M. *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*; IBM Technical Report; IBM Ltd.: Ottawa, ON, Canada , 1966.
21. Lauterbach, C.; Garland, M.; Sengupta, S.; Luebke, D.; Manocha, D. Fast BVH Construction on GPUs. *Comput. Graph. Forum* **2009**, *28*, 375–384. [CrossRef]
22. Karras, T. Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees. In Proceedings of the Eurographics/ACM SIGGRAPH Symposium on High Performance Graphics, Paris, France, 25–27 June 2012; pp. 33–37. [CrossRef]
23. Apetrei, C. Fast and Simple Agglomerative LBVH Construction. In Proceedings of the Theory and Practice of Computer Graphics, Leeds, UK, 10–11 September 2014; pp. 33–40.
24. Park, S.; Baek, N. A Shader-Based Ray Tracing Engine. *Appl. Sci.* **2021**, *11*, 3264. [CrossRef]
25. Arvo, J.; Kirk, D. Fast Ray Tracing by Ray Classification. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*; ACM: New York, NY, USA, 1987; pp. 55–64. [CrossRef]

26. Hendrich, J.; Pospisil, A.; Meister, D.; Bittner, J. Ray Classification for Accelerated BVH Traversal. *Comput. Graph. Forum* **2019**, *38*, 485–496. [CrossRef]
27. Deng, J. A Hybrid Millimeter-Wave Channel Simulator for Joint Communication and Localization. In Proceedings of the 2022 IEEE International Conference on Computer and Communications, Chengdu, China, 9–12 December 2022; pp. 84–89.
28. He, R.; Zhong, Z.; Ai, B.; Guan, K.; Chen, R. Simplified analytical propagation model for railway environments based on uniform theory of diffraction. *Electron. Lett.* **2013**, *49*, 397–399. [CrossRef]

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