IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 May 2026, accepted 23 June 2026. Date of publication 00 xxxx 0000, date of current version 00 xxxx 0000.

Digital Object Identifier 10.1109/ACCESS.2026.3708492

Extrapolated Asynchronous Sound Propagation for Real-Time Sound Rendering

SUKWON CHOI[1], (Graduate Student Member, IEEE), EUNJAE KIM[1], UIJUN KIM[1], JUNGWOONG SO[1], WOO-CHAN PARK[1], AND JAE-HO NAH[2], (Member, IEEE)

[1]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea
[2]Department of Computer Science, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Jae-Ho Nah (jaeho.nah@smu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant RS-2025-00521436.

ABSTRACT We propose an extrapolation-based asynchronous sound propagation algorithm to improve the performance of ray-tracing-based sound rendering. Traditional sound rendering algorithms require high computational costs during the sound propagation phase, making real-time processing highly challenging in environments with limited computational resources, such as devices with constrained CPU budgets for audio. To overcome this limitation, we implemented a method in which computationally intensive sound propagation is performed once every few frames, while the amplitudes for subsequent graphics frames are predicted using extrapolation. Experimental results on a workstation PC and a mobile device showed that the proposed asynchronous algorithm achieved average speedups of 3.5× and 3.83×, respectively, while maintaining acoustic similarity to the baseline method as measured by Perceptual Evaluation of Speech Quality (PESQ), Structural Similarity Index Measure (SSIM), and Scale-Invariant Signal-to-Noise Ratio (SI-SNR).

INDEX TERMS Real-time, sound rendering, ray tracing, sound propagation, geometric acoustics, extrapolation, asynchronous architecture.

I. INTRODUCTION

Advancements in digital media and interactive technologies have increased the demand for real-time rendering of not only graphics but also audio [1]. In virtual environments such as Virtual Reality (VR), Augmented Reality (AR), and video games, high-quality auditory experiences are pivotal for enhancing user immersion and realism. The realistic reproduction of spatial audio, in particular, plays a critical role in bringing user experiences closer to reality and enhancing user engagement in virtual environments [2], [3], [4], [5]. Consequently, the demand for advanced sound rendering has significantly increased [6].

Sound rendering is broadly divided into three stages: sound synthesis, sound propagation, and sound generation [7]. Throughout this paper, *sound rendering* denotes the complete pipeline that produces the final audio output at the listener using per-frame rendering updates and the latest available propagation results, whereas *sound propagation* refers specifically to the geometric-acoustics (GA)-based computation—including traversal, intersection tests, and valid-path verification—that estimates propagation-dependent acoustic parameters such as valid paths and path-associated amplitude coefficients from the scene geometry.

Sound synthesis generates audio signals by combining waveforms or processing data, while sound generation outputs these signals to the user, with both stages requiring relatively low computational costs. In contrast, the sound propagation stage involves calculating the paths of sound as it travels through virtual environments, which necessitates high computational costs for real-time processing [8], [9]. Unlike these propagation-dependent parameters, current rendering parameters—such as source–listener direction and current source–listener distance—are updated using the latest source and listener states and used in the final audio-output

The associate editor coordinating the review of this manuscript and approving it for publication was Tarcisio Ferreira Maciel.

generation process. Hereafter, we refer to these parameters as *spatial data*.

Real-time processing of sound propagation encounters difficulties in implementation due to its computational complexity [10]. To address this issue, several methods have been proposed to reduce computational load. For example, multiple sound sources can be clustered before sound propagation [11], or the depth level of propagation can be adaptively adjusted [12].

However, even with the application of such techniques, real-time sound propagation remains challenging if sufficient hardware resources are not allocated for audio processing. In practice, the CPU budget allocated for audio processing in typical games is only around 5–10% [13]. If the audio processing time per frame exceeds the graphics processing time, the display frame rate could be limited by the audio processing [14]. Therefore, for sound propagation to be practically used in real-time applications, the audio processing time must be equal to or, ideally, less than the graphics processing time.

To reduce computational resource requirements for real-time sound rendering, this paper proposes an asynchronous sound propagation algorithm. The key idea of this algorithm is to exploit extrapolation to predict amplitudes for subsequent graphics frames. Extrapolation predicts unknown or future values using existing data and is treated as a standard topic in numerical analysis [15], with related applications in machine learning [16] and image processing [17]. To the best of our knowledge, this work is among the first to apply extrapolation to GA-based sound propagation updates for decoupling expensive propagation computation from the graphics-frame loop.

To validate the effectiveness of the proposed algorithm, we compared its computational performance, measured in frames per second (FPS), and acoustic similarity with those of the multi-threaded synchronous baseline [18] on both a workstation PC and a mobile device. While the baseline performs full geometric acoustics (GA) propagation at every graphics frame, the proposed method executes it asynchronously and predicts amplitudes for subsequent graphics frames until the next SPS update. The proposed method achieved average speedups of 3.5× on the workstation PC and 3.83× on the mobile device, with maximum speedups of 12.43× and 6.79×, respectively.

Acoustic similarity was evaluated using PESQ, SSIM, and SI-SNR under static and dynamic listener conditions across both platforms and all benchmark scenes. Static listener conditions showed high acoustic fidelity, with PESQ scores ranging from 4.07 to 4.64, SSIM values remaining above 0.98, and SI-SNR exceeding 20 dB in most cases. Under dynamic listener conditions, PESQ ranged from 2.86 to 4.44, SSIM from 0.854 to 0.989, and SI-SNR from 4.6 to 19.33 dB.

This paper is organized as follows. Section II reviews related work on real-time sound rendering and extrapolation. Section III details the structure and workflow of the proposed algorithm. Section IV presents an analysis of the experimental results, evaluating performance and acoustic similarity across various benchmark scenes. Finally, Section V concludes the paper and discusses directions for future research. Table 1 summarizes the main symbols, notations, and acronyms used throughout this paper.

TABLE 1. Summary of symbols, notations, and acronyms.

Symbol	Description
GA	Geometric Acoustics
FPS	Frames Per Second
GPS	Graphics Processing Stage
SPS	Sound Propagation Stage
EPS	Extrapolation Stage
SGS	Sound Generation Stage
ZOH	Zero-Order Hold
PathID	Identifier used to match propagation paths across SPS updates
t	SPS update index ($t = 1, 2, \dots$)
k	Graphics-frame index within an extrapolation interval ($k = 0, 1, \dots, L_t - 1$)
p	Propagation path index
b	Frequency-band index
$T_{g,t}$	Graphics-frame time at SPS update index t
$T_{\mathrm{SPS},t}$	Processing time required to complete one SPS update
L_t	Target extrapolation level after SPS update t
$A_{p,b,t}$	Amplitude coefficient of path p and band b at SPS update t
$A_{p,b,t-1}$	Amplitude coefficient of path p and band b at the previous SPS update
$\tilde{A}_{p,b,t,k}$	Raw extrapolated amplitude before clamping
$\hat{A}_{p,b,t,k}$	Final extrapolated amplitude after clamping
$U_{p,b,t}$	Upper bound used in the amplitude limiter
PESQ	Perceptual Evaluation of Speech Quality
SSIM	Structural Similarity Index Measure
SI-SNR	Scale-Invariant Signal-to-Noise Ratio

II. RELATED WORK

This section reviews previous studies related to real-time sound rendering and acceleration approaches. First, we summarize geometric acoustics methods used to approximate sound propagation paths within virtual environments. We then review real-time sound rendering approaches that reduce the computational burden of sound propagation through hardware- and software-based acceleration. Finally, we discuss extrapolation techniques used in different fields and identify their limitations when directly applied to geometry-dependent sound propagation.

Through this review, we clarify the motivation for the proposed extrapolation-based asynchronous sound propagation algorithm. Table 2 provides a concise overview of representative studies and their key distinctions from the proposed approach.

A. GEOMETRIC ACOUSTICS METHODS

GA methods are modeling techniques that approximate wave characteristics using approaches such as ray tracing, beam tracing, frustum tracing, and the image source method [10], [19]. These methods simulate phenomena such

TABLE 2. Summary of related work and key distinctions.

Category	References	Main Contribution	Key Distinction
GA-based sound propagation	[1], [10], [22]	Approximate sound propagation using geometric paths and ray/path tracing.	Require costly traversal, intersection tests, and valid-path verification for every frame.
GA-based room-acoustic rendering	[20], [21]	Approximate GA-based room-acoustic effects efficiently.	Focus on acoustic-effect approximation rather than sound-propagation update reduction.
Real-time sound rendering	[7], [8], [13]	Enable interactive sound synthesis, propagation, and rendering for VR and games.	Suffer from performance degradation with increased scene complexity and sound sources.
Sound propagation acceleration	[11], [12], [18], [23], [26]	Reduce the computational cost of sound propagation.	Reduce per-update propagation cost rather than update frequency.
Spatial-audio interpolation	[29], [30]	Interpolate RIR/HRTF data for unmeasured positions.	Focus on spatial response/filter estimation, not temporal path-amplitude prediction.
Extrapolation techniques	[15], [27], [28], [31], [32]	Predict values in numerical, audio, and graphics domains.	Operate on signal/image data rather than GA path coefficients.

as reflection, absorption, transmission, and diffraction occurring in real environments. When ray tracing is used, the propagation paths of rays emitted from a sound source are traced, and when the rays are reflected or diffracted, sound waves propagate following physical laws, with their energy attenuating based on the absorption coefficient of surfaces. These computational processes can be used to calculate the propagation paths within a given space. Recent GA-oriented room-acoustic rendering studies have also explored efficient approximations for acoustic radiance transfer, scattering, and diffuse reflections in room simulation [20], [21].

In this study, we use a ray-tracing-based GA pipeline as the baseline. It involves computationally intensive steps such as traversal, intersection tests, and valid-path verification, which present challenges for real-time processing [22], [23]. A valid path refers to the physical route through which sound emitted from a source can reach the listener, and the number of such paths may vary depending on the structural complexity of the scene. Rather than minimizing the cost of a single sound-propagation update, our proposed method focuses on *temporal acceleration* by reducing the update frequency and synthesizing extrapolated amplitudes for subsequent graphics frames.

B. REAL-TIME SOUND RENDERING APPROACHES

Some recent studies have proposed hardware-accelerated approaches to achieve real-time performance for the sound rendering pipeline. A high-performance, low-power hardware architecture dedicated to the sound rendering pipeline has been introduced, which significantly improves both processing speed and power efficiency compared to software-based approaches [23]. Additionally, some research has leveraged the parallel processing capabilities of GPUs to enhance computational performance and accurately model the physical properties of sound propagation [24], [25].

In addition to hardware-accelerated and GPU-based approaches, software-side acceleration techniques for CPU platforms have also been explored to achieve real-time performance. For instance, a multi-threaded synchronous pipeline was introduced to parallelize sound propagation tasks [18]. Furthermore, the impact of different acceleration structures, such as k-d trees and multi-bounding volume hierarchies (MBVHs), has been investigated with SIMD optimizations to improve ray-tracing efficiency for sound propagation on mobile CPUs [26]. Because these approaches accelerate the computation of a single sound-propagation update rather than changing how frequently sound propagation is performed, they are orthogonal to the proposed asynchronous approach and could in principle be combined with it for additional performance gains.

C. EXTRAPOLATION TECHNIQUES

The extrapolation technique employed in this study is utilized across various fields. In numerical analysis, it is used to predict solutions of differential equations or estimate the state of time-varying systems [15]. In machine learning, it has been studied for improving extrapolative prediction and control beyond the observed training range [16]. In image processing, it is applied to extrapolate data across frames, predicting motion or restoring lost image information [17].

In sound processing, audio-frame and audio-signal extrapolation techniques [27], [28] are used to extend or restore acoustic signal segments. These techniques analyze patterns in past signal segments and reconstruct missing acoustic data by extrapolating lost samples. However, such methods perform extrapolation in the signal domain, making them difficult to directly apply to GA-based sound propagation, which requires geometry-dependent path information.

In spatial audio, interpolation has also been studied for room impulse responses (RIRs) and head-related transfer functions (HRTFs). RIR interpolation methods estimate room impulse responses at unmeasured positions from measured RIRs using acoustic models and inverse-problem formulations [29]. HRTF interpolation methods estimate HRTFs at unmeasured source positions by interpolating measured HRTF data across azimuth, elevation, and distance [30]. Related temporal interpolation methods for spatial filters or binaural responses (e.g., BRIRs) mainly aim to maintain smooth auditory transitions during source or listener movement. In contrast, the proposed method extrapolates path-associated frequency-band amplitude coefficients produced by the SPS, with the goal of decoupling expensive GA propagation updates from the graphics-frame loop.

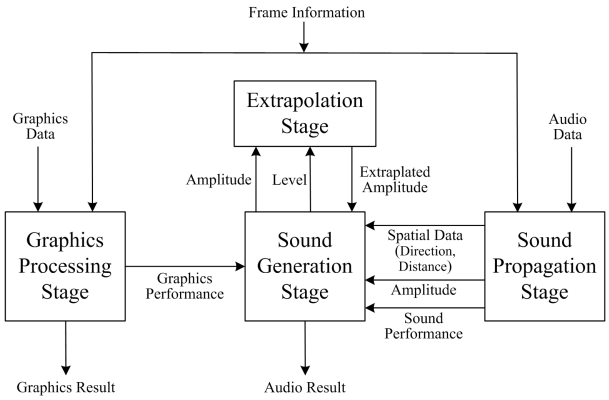

FIGURE 1. **Overall structure of the proposed extrapolation-based asynchronous sound propagation algorithm. SPS computes propagation-dependent path-associated amplitude coefficients, EPS extrapolates matched amplitudes between SPS updates, and SGS combines the latest available amplitudes with spatial data to generate the final audio result.**

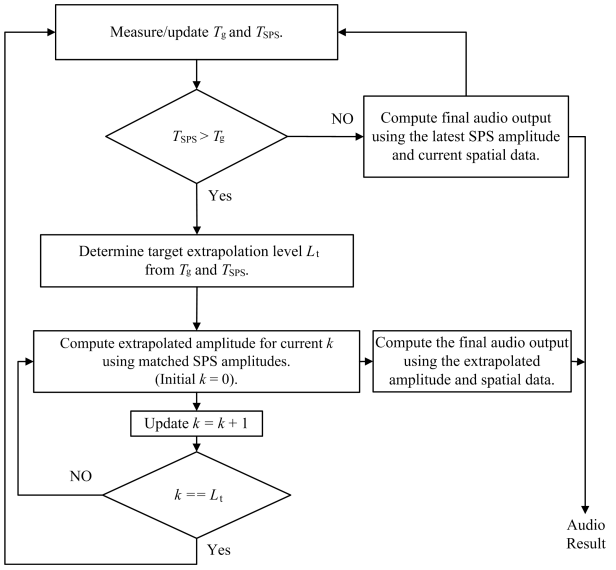

FIGURE 2. **Workflow of the proposed extrapolation-based asynchronous sound propagation algorithm. For readability, the update index t is omitted in the flowchart; thus, T_g and T_{SPS} denote $T_{g,t}$ and $T_{SPS,t}$, respectively.**

Extrapolation methods have also been applied to GPU-based rendering. In [31], a framework that combines spatial supersampling and extrapolation is proposed to generate high-resolution results from low-resolution inputs in real-time rendering. In [32], a method is introduced to reduce latency and generate high-quality, high-resolution images through frame extrapolation, without the need for a G-buffer. However, these extrapolation-based frame generation methods are not directly applicable to GA-based sound propagation, as they target graphics rather than audio.

III. PROPOSED ALGORITHM

This section describes the proposed extrapolation-based asynchronous sound propagation algorithm. The key idea is to decouple the expensive Sound Propagation Stage (SPS) from the high-frequency graphics rendering loop by predicting amplitudes for subsequent graphics frames. The spatial data, such as direction and distance, are sensitive to per-frame updates and thus are updated at every graphics frame to maintain responsiveness.

This design is motivated by the computational mismatch between high-frequency graphics rendering and expensive sound propagation. Since GA-based propagation involves traversal, intersection tests, and valid-path verification, executing the SPS at every graphics frame can create a rendering bottleneck. Amplitude is chosen as the primary extrapolation target because it is a propagation-dependent acoustic parameter obtained from SPS updates and can be temporally predicted between consecutive propagation results. In contrast, spatial data such as direction and distance are updated per frame to ensure immediate perceptual responsiveness to listener movement. Linear extrapolation is employed to keep the EPS computationally lightweight and prevent the prediction process from introducing additional overhead.

A. SYSTEM OVERVIEW

Figure 1 illustrates the overall structure. The system consists of four stages: *Graphics Processing Stage* (GPS), *Sound Propagation Stage* (SPS), *Extrapolation Stage* (EPS), and *Sound Generation Stage* (SGS).

As defined in Section I, *sound propagation* refers to the GA-based computation performed within the SPS, whereas *sound rendering* denotes the complete pipeline that produces the final audio output using per-frame rendering updates and the latest available propagation results. This separation is motivated by the mismatch between high-frequency graphics updates and the substantial computational cost of GA-based propagation. Therefore, the proposed method decouples the SPS from the graphics-frame loop by executing propagation asynchronously; the EPS estimates path-associated amplitude coefficients for subsequent graphics frames, while the SGS remains active at the graphics-frame rate using the latest available amplitudes and current spatial data.

The GPS processes per-frame scene information and produces both graphics output and an estimate of the graphics frame rate. The SPS performs geometric-acoustics-based sound propagation to compute propagation-dependent acoustic data (e.g., amplitude). The SGS generates the final audio output by combining the most recent amplitude with the current frame's spatial data. When the SPS updates are not available at every graphics frame, the EPS extrapolates amplitude values for subsequent graphics frames, thereby enabling the SGS to update the audio rendering parameters at every graphics frame without requiring SPS execution each time.

More specifically, the amplitude passed from the SPS to the EPS is a path-associated frequency-band coefficient, not a raw source-signal amplitude. This coefficient is computed for each valid path generated by the SPS and already

includes propagation-dependent effects such as geometry-dependent visibility and occlusion, material interaction, reflection/transmission loss, diffraction response, path-length-dependent attenuation, and angle-dependent reflection or absorption effects where applicable. The EPS only predicts these path-associated amplitude coefficients between SPS updates and does not recompute geometric propagation effects. The SGS then combines the latest available path-associated amplitude coefficients with the spatial data to generate the final audio output.

B. RUNTIME WORKFLOW

Figure 2 shows the runtime workflow of the proposed asynchronous sound propagation algorithm. Let $T_{g,t}$ denote the graphics-frame time at SPS update index t, and let $T_{\mathrm{SPS},t}$ denote the processing time required to complete one SPS update. The runtime logic for EPS activation and target extrapolation-level selection compares the processing time of one SPS execution with the graphics-frame time.

If the SPS processing time can be accommodated within the graphics-frame time, the SGS directly generates the final audio output using the latest SPS amplitude and the current frame's spatial data, without extrapolation. Otherwise, the system activates the EPS and determines the target extrapolation level L_t for the following extrapolation interval. For each subsequent graphics frame indexed by $k = 0, \ldots, L_t - 1$, the EPS computes an extrapolated amplitude, and the SGS combines it with the corresponding frame's spatial data to produce the final audio output.

C. EXTRAPOLATION LEVEL AND AMPLITUDE EXTRAPOLATION

We define two components for subsequent-frame synthesis: the target extrapolation level L_t and normalized amplitude extrapolation. Here, t denotes the SPS update index, while k denotes the graphics-frame index within the extrapolation interval after the SPS update.

- **Extrapolation level:** We define the target extrapolation level L_t as the number of subsequent graphics frames after the SPS update at index t for which extrapolated amplitudes are used before the next SPS result is incorporated. Let $T_{g,t}$ denote the graphics-frame time, and let $T_{\mathrm{SPS},t}$ denote the processing time required to complete one SPS update. The target extrapolation level is then defined as

$$L_t = \max\left(0, \left\lceil \frac{T_{\mathrm{SPS},t}}{T_{g,t}} \right\rceil - 1\right). \tag{1}$$

  This definition uses the measured processing time of one SPS execution rather than the elapsed interval between scheduled SPS updates. Therefore, the computation of L_t is not circular. Once L_t is determined after an SPS update, it remains fixed during the following extrapolation interval and is not recomputed until the next SPS result becomes available.
- **Amplitude extrapolation:** We adopt linear extrapolation to keep the EPS computationally lightweight and to prevent the prediction step from becoming a secondary bottleneck. As shown in Figure 3, let A_{t-1} and A_t denote the path-associated eight-band amplitude coefficients from the previous and current SPS results, respectively. For path p and frequency band b, the corresponding coefficients are written as $A_{p,b,t-1}$ and $A_{p,b,t}$. Since these SPS updates are separated by approximately $L_t + 1$ graphics-frame steps, the difference $A_{p,b,t} - A_{p,b,t-1}$ represents the amplitude variation of the corresponding path and frequency band over one SPS interval, not over a single graphics frame. Therefore, the per-frame extrapolation slope is obtained by normalizing this difference by $L_t + 1$.

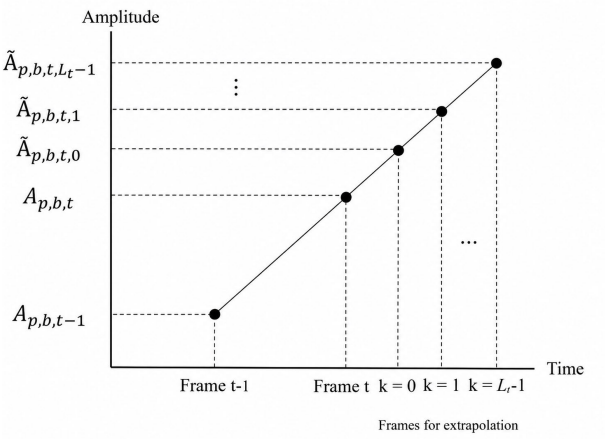

FIGURE 3. Conceptual illustration of normalized amplitude extrapolation. The amplitude difference between two consecutive SPS updates is divided by $L_t + 1$ to obtain a graphics-frame-level extrapolation step.

For a subsequent graphics frame indexed by $k = 0, \ldots, L_t - 1$ after the SPS update at index t, the EPS first computes the raw extrapolated amplitude $\tilde{A}_{p,b,t,k}$ for path p and frequency band b as

$$\tilde{A}_{p,b,t,k} = A_{p,b,t} + \frac{k+1}{L_t+1}\left(A_{p,b,t} - A_{p,b,t-1}\right) \tag{2}$$

Here, the factor $(k+1)/(L_t+1)$ converts the amplitude variation observed over one SPS interval into a graphics-frame-level extrapolation step. The raw extrapolated amplitude $\tilde{A}_{p,b,t,k}$ is then passed to the clamping and limiter step described in Section III-E. When the next SPS result becomes available, the extrapolated amplitude is replaced by the newly computed SPS amplitude.

In the initial bootstrap stage, extrapolation is disabled until a previous–current SPS pair is available. Before this pair is formed, each available SPS result is used directly by the SGS without extrapolation. Once the pair is available, it forms the previous–current SPS pair shown in Fig. 3, corresponding to A_{t-1} and A_t, from which subsequent extrapolated amplitudes are generated. The SGS combines the latest available path-associated amplitude coefficients with the spatial data for final rendering and spatialization.

D. PATH CORRESPONDENCE AND AMPLITUDE HANDLING

The amplitude extrapolation in Eq. (2) assumes that the amplitudes from two consecutive SPS results correspond to the same propagation path. However, the SPS output is a list of valid propagation paths, and this list can change between SPS updates. Therefore, before applying amplitude extrapolation, the EPS first establishes path correspondence between the two most recent SPS results.

To align the two path lists, the EPS uses PathID. In the implementation, PathID is stored as an integer identifier for comparing paths across consecutive SPS results. After sorting both path lists by PathID, paths with matching PathIDs are treated as matched path pairs, and their amplitudes are extrapolated using Eq. (2).

For paths that exist in only one of the two SPS results, the EPS uses a PathID-based handling rule instead of linear extrapolation. If a path appears only in the current SPS result, its current SPS amplitude is used without extrapolation during the subsequent extrapolation interval, because no matched previous amplitude is available for slope estimation. If a path exists only in the previous SPS result but not in the current SPS result, it is excluded from the EPS output for that interval. Consequently, Eq. (2) is applied only to paths whose PathID is matched across consecutive SPS updates.

E. AMPLITUDE CLAMPING AND DIRECT-PATH HANDLING

Because the extrapolated quantity represents a physical amplitude coefficient, the raw extrapolated value must remain within a physically valid range. In particular, linear extrapolation can produce non-physical negative amplitudes or excessive overshoot when the amplitude changes rapidly between two SPS updates. To prevent this behavior, the proposed EPS applies a lightweight clamping and limiter step after raw amplitude extrapolation.

Let $\tilde{A}_{p,b,t,k}$ denote the raw extrapolated amplitude of path p and frequency band b for the k-th graphics frame after the SPS update at index t. We first define the upper bound as

$$U_{p,b,t} = \max\left(A_{p,b,t}, A_{p,b,t-1}\right) + \left|A_{p,b,t} - A_{p,b,t-1}\right|. \quad (3)$$

The final amplitude used by the SGS is then computed as

$$\hat{A}_{p,b,t,k} = \min\left(\max\left(\tilde{A}_{p,b,t,k}, 0\right), U_{p,b,t}\right). \quad (4)$$

The lower bound prevents non-physical negative amplitudes. The upper bound $U_{p,b,t}$ acts as a lightweight limiter that suppresses excessive overshoot relative to the two most recent SPS amplitudes.

Direct paths are handled more conservatively because they are highly sensitive to source–listener visibility. In the proposed implementation, direct paths are revalidated at every graphics frame and are not extrapolated in the EPS; therefore, occlusion or visibility changes are reflected using the latest direct-path validation result.

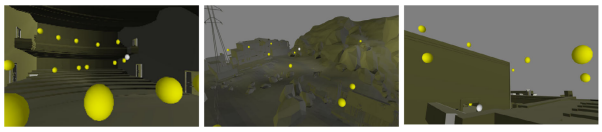

FIGURE 4. Benchmark scenes: (from left to right) Concert Hall, Bootcamp, and Angrybot.

IV. EXPERIMENTS AND RESULTS

This section evaluates the computational performance and acoustic similarity of the proposed asynchronous sound propagation algorithm. We measured performance in FPS and acoustic similarity across scenes, sound-source counts, and listener movement conditions. The evaluation aims to quantify the performance speedup and analyze the trade-off between rendering efficiency and acoustic similarity.

This section is organized as follows. Subsection A describes the experimental setup. Subsections B and C present the performance results on the workstation PC and mobile device, respectively. Subsection D analyzes acoustic similarity using multiple evaluation metrics.

A. EXPERIMENTAL SETUP

The experiments were conducted on two platforms: a workstation PC equipped with an Intel Core i9-10850K CPU, an NVIDIA GeForce RTX 3080 GPU, and 32 GB of RAM, and a Samsung Galaxy S20+ smartphone equipped with a Qualcomm Snapdragon 865 SoC and 12 GB of RAM. To evaluate the proposed asynchronous sound propagation algorithm, we utilized the three benchmark scenes shown in Figure 4.

These scenes were selected to represent different acoustic characteristics and scene complexities: Concert Hall represents an indoor environment with dense reflective surfaces, Bootcamp represents an outdoor open space with fewer reflections, and Angrybot represents a hybrid environment containing both indoor and outdoor structures. In each scene, the number of sound sources and valid paths directly affect the computational cost, serving as key factors in the performance of the proposed algorithm. Thus, the selected benchmark scenes provide different propagation workloads for evaluating the proposed method.

As a baseline for evaluation, we implemented the multi-threaded synchronous sound propagation pipeline [18] using two threads to match the CPU-thread budget of the proposed implementation. In all experiments, the acoustic rays were traced with a maximum reflection depth of 4, 1,024 guide rays, and 128 source rays per sound source, following prior real-time GA-based sound propagation systems [18], [23]. The maximum reflection depth was selected to include multi-bounce propagation, and the guide-ray and source-ray counts were chosen to provide sufficient scene-dependent path sampling under the evaluated real-time setting. These parameters were fixed across all benchmark scenes and source-count conditions so that the measured performance differences reflect the asynchronous update structure rather than changes in propagation fidelity.

TABLE 3. Performance comparison on a workstation PC in FPS. In the Scene column, parentheses indicate the triangle count (k) and environment type; in the Proposed rows, they indicate the speedup factor relative to the baseline.

Platform: Workstation PC						
		Number of Sound Sources				
Scene	Method	1	2	4	8	16
Concert Hall (25K, Indoor)	Baseline	45.6	30.2	19.4	11.0	6.0
	Proposed	122.1 (2.68×)	101.2 (3.36×)	94.1 (4.84×)	84.6 (7.69×)	74.6 (**12.43**×)
Bootcamp (130K, Outdoor)	Baseline	249.1	204.8	160.8	96.4	57.3
	Proposed	251.0 (1.01×)	213.7 (1.04×)	165.8 (1.03×)	162.0 (1.68×)	124.8 (2.18×)
Angrybot (26K, Hybrid)	Baseline	97.2	71.0	46.2	26.9	17.1
	Proposed	166.3 (1.71×)	129.8 (1.83×)	108.7 (2.35×)	95.3 (3.55×)	87.1 (5.09×)

TABLE 4. Performance comparison on a mobile device in FPS (parentheses as in Table 3).

Platform: Mobile Device						
		Number of Sound Sources				
Scene	Method	1	2	4	8	16
Concert Hall (25K, Indoor)	Baseline	21.7	15.6	10.3	6.1	3.8
	Proposed	82.4 (3.79×)	68.6 (4.39×)	48.2 (4.68×)	35.6 (5.86×)	25.8 (**6.79**×)
Bootcamp (130K, Outdoor)	Baseline	48.8	40.2	29.8	20.4	12.3
	Proposed	88.2 (1.81×)	82.6 (2.05×)	77.9 (2.62×)	73.0 (3.58×)	65.6 (5.35×)
Angrybot (26K, Hybrid)	Baseline	39.7	32.0	23.4	14.8	7.8
	Proposed	83.4 (2.10×)	75.4 (2.36×)	64.2 (2.75×)	53.5 (3.62×)	45.0 (5.75×)

In the static condition, the listener remained fixed. In the dynamic condition, the listener moved back and forth along a predefined straight left–right trajectory at a constant speed, while the sound sources remained fixed. The listener reversed direction at the endpoints of the trajectory. The same trajectory was used for all evaluated methods.

Performance evaluation was conducted by measuring FPS in each scene while varying the number of sound sources, 1, 2, 4, 8, and 16. Acoustic similarity evaluation was performed under both static and dynamic listener conditions across all scenes and source counts. The baseline method performs full GA propagation at every graphics frame, whereas the proposed method executes propagation asynchronously and extrapolates amplitudes for subsequent graphics frames. For the sound sources, speech signals sampled at 44.1 kHz were used consistently across all benchmark scenes as controlled audio inputs for acoustic similarity evaluation. In multi-source scenarios, sound sources were spatially distributed across the environment.

B. WORKSTATION PC EVALUATION

Table 3 compares the FPS of the baseline method and the proposed algorithm across three scenes with varying levels of complexity. The proposed algorithm consistently outperforms the baseline method in all test environments, and the speedup increases as the number of sound sources grows. This trend is most pronounced in the Concert Hall scene, where the baseline is heavily bottlenecked by frequent path recomputations.

This performance difference becomes particularly pronounced in indoor environments, such as the Concert Hall. In the baseline method, FPS drops significantly in scenes with a high number of valid paths, as propagation paths are recalculated for every frame. In contrast, the proposed algorithm mitigates performance degradation by extrapolating path-associated amplitudes between SPS updates, thereby reducing the need for full path recomputation at every graphics frame. The maximum speedup of 12.43× was achieved in the Concert Hall scene with 16 sound sources, where the baseline method was most constrained by computational cost.

C. MOBILE DEVICE EVALUATION

Table 4 reports the FPS results on the mobile device. The proposed method consistently outperformed the baseline across all scenes and source counts. The average mobile speedup was approximately 3.83×, and the maximum speedup reached 6.79× in the Concert Hall scene with 16 sound sources. This trend is consistent with the workstation experiment, where the largest gain was also observed in Concert Hall. The indoor reflective structure of Concert Hall produces a dense valid-path workload, making repeated full SPS execution expensive.

A cross-platform comparison further shows that the performance trend is not determined by valid-path density alone. Although Bootcamp has a lower valid-path density than Concert Hall, it contains the largest scene geometry among the tested environments. On the workstation, Bootcamp maintained a relatively high baseline FPS of 57.3 even with 16 sound sources, and the corresponding speedup was limited to 2.18×. On the mobile device, however, the same 16-source Bootcamp condition dropped to 12.3 baseline FPS and achieved a 5.35× speedup. These results indicate that reducing the frequency of full SPS updates remains effective when geometric traversal workloads become a major contributor to per-frame computation.

D. ACOUSTIC SIMILARITY ANALYSIS

1) EVALUATION METRICS

To provide a multi-dimensional assessment of acoustic similarity, we employed the following metrics:

Scale-Invariant Signal-to-Noise Ratio (SI-SNR) [33]: SI-SNR measures the signal-to-error energy ratio in the time domain after optimally compensating for scale differences between a reference signal and an estimated signal. Higher SI-SNR values indicate that the estimated signal is closer

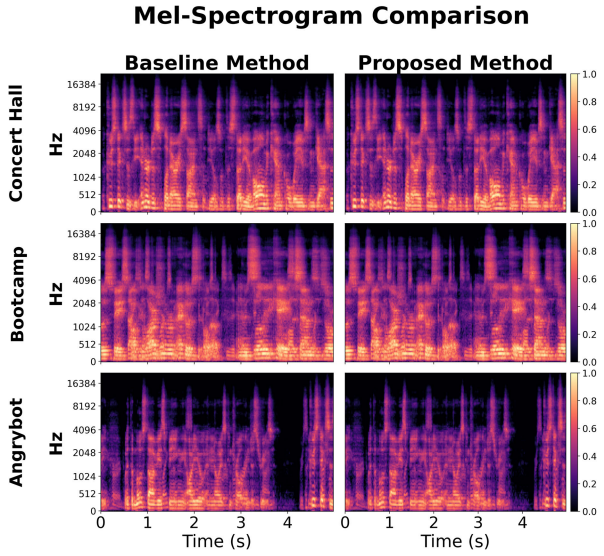

FIGURE 5. Comparison of Mel-spectrograms for 16-source dynamic listener conditions.

to the reference. SI-SNR is defined as follows:

$$s_{\text{target}} = \frac{\langle \hat{s}, s \rangle}{\|s\|^2} s, \qquad e_{\text{noise}} = \hat{s} - s_{\text{target}}, \tag{5}$$

$$\text{SI}-\text{SNR} = 10 \log_{10} \left(\frac{\|s_{\text{target}}\|^2}{\|e_{\text{noise}}\|^2} \right). \tag{6}$$

Here, s denotes the reference signal and $\hat{s}$ denotes the estimated signal. The projection s_{target} represents the scale-aligned target component of $\hat{s}$, and e_{noise} represents the residual error component.

Perceptual Evaluation of Speech Quality (PESQ) [34]: PESQ is an objective metric standardized by ITU-T that estimates subjective Mean Opinion Score (MOS) by modeling human auditory perception. Higher scores indicate better perceptual quality.

$$\text{PESQ} = a_0 + a_1 D_{\text{ind}} + a_2 A_{\text{ind}}, \tag{7}$$

where D_{ind} represents the average disturbance value measuring perceptual degradation, A_{ind} represents the average asymmetric disturbance value accounting for added noise components, and a_0, a_1 and a_2 are regression coefficients to the MOS-LQO scale. We employ the wideband extension (ITU-T P.862.2), which yields scores up to 4.64 for perceptually identical signals. PESQ wideband evaluates speech quality up to approximately 8 kHz and is not intended to assess fidelity in higher-frequency bands. This metric is utilized as a validated approach to quantify human auditory perception, leveraging its proven high correlation with subjective listening tests [35].

Structural Similarity Index Measure (SSIM) [36]: SSIM quantifies structural similarity between two 2D representations by comparing local luminance, contrast, and structural patterns. In this work, we compute SSIM on log Mel-Spectrograms [37], with each spectrogram normalized to the same dynamic range. Higher SSIM indicates closer

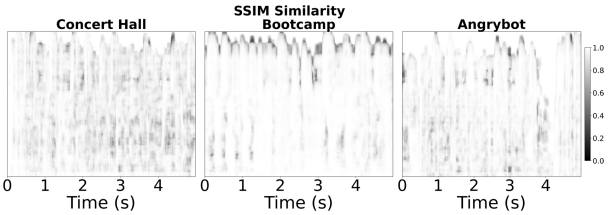

FIGURE 6. SSIM similarity maps for 16-source dynamic listener conditions.

time–frequency similarity (SSIM $\approx$ 1 denotes near-identical structures).

Figures 5 and 6 compare the proposed method and the baseline for the evaluated scenes (Concert Hall, Bootcamp, and Angrybot) under a 16-source dynamic listener condition. As shown in Figure 5, the two methods exhibit closely matched spectral bands and overall energy distributions. Figure 6 shows localized discrepancies, which are mainly associated with asynchronous integration intervals where newly valid paths are incorporated during the baseline's traversal. These differences appear primarily as brief misalignments in high-frequency components before the extrapolated result is updated.

2) QUANTITATIVE ACOUSTIC SIMILARITY RESULTS

Table 5 summarizes the acoustic similarity metrics across all test scenarios. The results demonstrate that the proposed method achieves high fidelity under static listener conditions and generally preserves perceptual and structural similarity even under dynamic listener conditions.

Under static listener conditions, all metrics indicate high acoustic similarity. PESQ scores approach or reach the maximum value of 4.64 in most cases, ranging from 4.07 to 4.64, and SSIM values remain above 0.98 in all cases and above 0.99 in most cases, indicating strong spectral structure preservation. SI-SNR also remains high in most static cases, with many values exceeding 20 dB and several cases exceeding the 40 dB reporting threshold. These results indicate that when the listener is stationary, the final rendered outputs of the proposed method closely match those of the frame-synchronous propagation baseline. The small variance observed in the static Concert Hall case is likely due to Monte Carlo ray-sampling noise, which becomes more evident in an indoor environment with dense reflected paths.

Under dynamic listener conditions, listener motion changes the source–listener distances and can alter the set of valid propagation paths. These time-varying changes result in lower similarity metrics compared with the static listener condition. Across all dynamic cases on both platforms, PESQ ranges from 2.86 to 4.44, SSIM ranges from 0.854 to 0.989, and SI-SNR ranges from 4.6 to 19.33 dB. This reduction is expected because time-varying propagation changes make the extrapolation of path-associated amplitude coefficients more challenging between SPS updates. With more sound sources, the system must extrapolate more

TABLE 5. Acoustic similarity metrics on the workstation PC and mobile device across different environments, listener conditions, and source counts.

Scene	Metric	Number of Sound Sources				
		1	2	4	8	16
Workstation PC – Static Listener Condition						
Concert Hall	PESQ	4.62±0.01	4.57±0.01	4.61±0.01	4.60±0.01	4.61±0.02
	SSIM	0.998	0.998	0.999	0.999	0.999
	SI-SNR	32.20±0.29	24.02±0.76	26.83±0.17	26.61±0.18	27.28±0.17
Bootcamp	PESQ	4.64	4.64	4.64	4.64	4.64
	SSIM	1.000	1.000	1.000	1.000	1.000
	SI-SNR	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$
Angrybot	PESQ	4.64	4.64	4.64	4.64	4.64
	SSIM	1.000	1.000	1.000	1.000	1.000
	SI-SNR	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$
Workstation PC – Dynamic Listener Condition						
Concert Hall	PESQ	3.80±0.20	3.75±0.17	3.49±0.14	3.16±0.12	2.86±0.11
	SSIM	0.953±0.010	0.954±0.002	0.915±0.006	0.885±0.005	0.854±0.005
	SI-SNR	11.62±1.49	10.84±0.25	6.24±0.39	4.93±0.19	4.60±0.39
Bootcamp	PESQ	4.44±0.09	4.10±0.10	3.83±0.08	3.68±0.17	3.78±0.32
	SSIM	0.989	0.965	0.928	0.923±0.012	0.918±0.023
	SI-SNR	19.33±3.27	10.62±1.98	7.88±0.43	7.74±0.73	7.59±1.45
Angrybot	PESQ	4.37±0.10	4.15±0.21	3.85±0.13	3.55±0.11	3.41±0.19
	SSIM	0.989	0.973	0.961±0.012	0.941	0.907
	SI-SNR	18.67±1.00	13.92±2.05	9.93±0.93	9.38±0.35	7.31±0.24
Mobile Device – Static Listener Condition						
Concert Hall	PESQ	4.59±0.01	4.53±0.01	4.49	4.40±0.05	4.07±0.03
	SSIM	0.997	0.998	0.996	0.984	0.983
	SI-SNR	35.55±0.06	32.53±0.25	25.79±0.14	25.71±1.11	16.95±0.35
Bootcamp	PESQ	4.64±0.01	4.64	4.62±0.04	4.64	4.64
	SSIM	0.999	1.000	0.998	1.000	1.000
	SI-SNR	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$
Angrybot	PESQ	4.64	4.64	4.64	4.64	4.54±0.01
	SSIM	1.000	1.000	1.000	1.000	0.998
	SI-SNR	$> 40^*$	$> 40^*$	$> 40^*$	$> 40^*$	36.70±0.24
Mobile Device – Dynamic Listener Condition						
Concert Hall	PESQ	4.19±0.18	4.19±0.07	3.77±0.09	3.01±0.20	3.43±0.08
	SSIM	0.962	0.951	0.916±0.018	0.879±0.019	0.928
	SI-SNR	12.34±1.64	10.35±0.84	8.78±0.33	5.95±0.37	6.57±0.38
Bootcamp	PESQ	4.30±0.06	4.28±0.08	4.19±0.03	4.06±0.09	3.71±0.10
	SSIM	0.971	0.970	0.956	0.938	0.928
	SI-SNR	14.74±2.74	13.07±1.25	11.60±1.34	9.64±1.15	8.12±0.58
Angrybot	PESQ	4.25±0.07	4.19±0.10	3.48±0.18	3.38±0.27	3.21±0.05
	SSIM	0.986	0.978	0.920	0.906±0.032	0.854
	SI-SNR	16.10±0.37	13.10±0.24	10.82±0.25	8.08±0.79	7.90±0.36

* SI-SNR values above 40 dB are reported as > 40 dB, indicating near-zero residual noise.
Values are mean ± standard deviation; zero-rounded standard deviations are omitted.

source-path contributions, so small amplitude mismatches can accumulate in the final mixed audio signal. Therefore, these reductions reflect a natural trade-off of reducing the frequency of full SPS updates.

The Concert Hall scene shows this trade-off most clearly on the workstation PC. As an indoor enclosed environment, Concert Hall contains dense reflected paths, making path amplitudes and valid-path composition more sensitive to listener motion than in acoustically simpler or more open scenes. The frame-synchronous baseline must recompute these complex paths every frame, which leads to substantial FPS degradation. In contrast, the proposed asynchronous method reduces this computational burden by lowering the frequency of full SPS updates and using linear extrapolation between SPS results. This explains the observed trade-off: Concert Hall achieves the highest speedup of 12.43×, while PESQ and SI-SNR decrease under dynamic listener conditions.

The mobile device results further show that the overall tendency is not limited to the workstation PC. Although the absolute metric values vary across scenes and source counts, the proposed method shows similar behavior on the

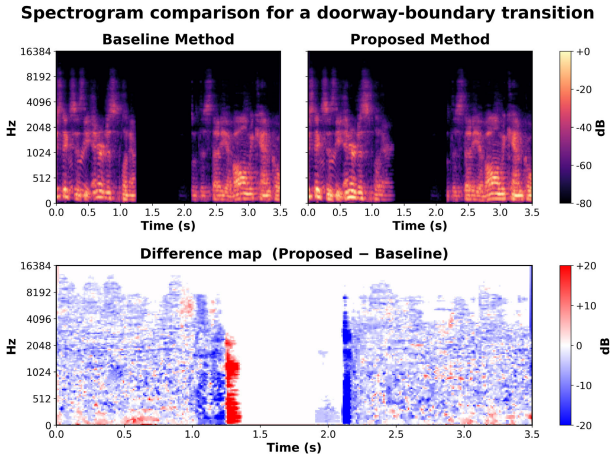

FIGURE 7. Spectrogram comparison for a doorway-boundary transition. The selected window contains the moment at which the listener intentionally crosses a wall boundary near a doorway, causing valid propagation paths to rapidly disappear and reappear. The proposed asynchronous method exhibits a pronounced but localized mismatch near the transition interval, reflecting the temporary lag between the current listener position and the most recent SPS result.

mobile platform: static listener conditions yield consistently high similarity, whereas dynamic listener conditions reduce the metrics due to time-varying propagation changes. A notable exception to this source-count-wise degradation trend appears in the mobile dynamic Concert Hall case, where all three metrics increase from 8 to 16 sources rather than decreasing monotonically. This suggests a limitation of linear extrapolation-based amplitude prediction: because the EPS predicts intermediate path-associated amplitudes rather than recomputing them through the SPS, similarity depends on how closely the temporal variation of these amplitudes follows a linear trend, not strictly on the number of sources.

3) ANALYSIS OF ABRUPT PATH CHANGES

To examine the proposed method under a worst-case sudden propagation change, we analyze a representative doorway-boundary stress case in which the listener intentionally crosses a wall boundary near a doorway. This motion causes many valid propagation paths to disappear or reappear within a short interval and creates a temporary mismatch between the current listener position and the most recent SPS result.

As shown in Figure 7, the difference map exhibits two adjacent regions of opposite sign around the transition: a residual-energy region where stale paths remain temporarily active, and a missing-energy region where newly valid paths are not yet reflected. This arises because the extrapolated path-associated amplitudes are refreshed only when the next SPS result becomes available. This result highlights a limitation of the proposed asynchronous update scheme: abrupt path changes can introduce short-term mismatch. In this representative stress case, the mismatch is concentrated around the boundary-crossing interval rather than being distributed throughout the entire window.

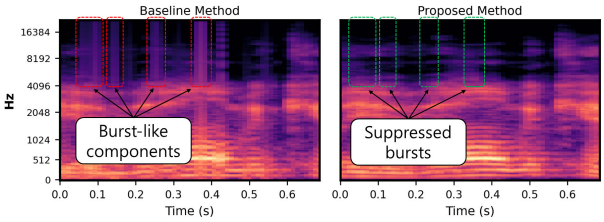

FIGURE 8. Representative zoomed spectrogram excerpt from real-time playback recordings of the 8-source dynamic Concert Hall condition. The synchronous baseline shows burst-like components in several mid- and high-frequency regions under a heavy SPS workload, whereas the corresponding regions in the proposed playback appear weaker or suppressed.

TABLE 6. Scene- and source-count-wise summary of the ZOH comparison under dynamic listener conditions. Each entry reports Extrapolation/ZOH wins over PESQ, SSIM, and SI-SNR after averaging repeated paired runs.

	# of sound sources				
Scene	1	2	4	8	16
Concert Hall	1/2	1/2	2/1	2/1	3/0
Bootcamp	3/0	0/3	0/3	2/1	2/1
Angrybot	1/2	3/0	3/0	2/1	2/1
Total	5/4	4/5	5/4	6/3	7/2

4) REAL-TIME PLAYBACK OBSERVATION

Separately from the stored-output-based acoustic similarity metrics reported in Table 5, we examined real-time playback recordings in the 8-source dynamic Concert Hall condition. This condition was selected as a representative heavy-workload case in which the SPS workload of the synchronous baseline can exceed the available frame budget.

When the SPS workload exceeds the frame budget, the synchronous baseline can exhibit irregular playback timing and temporal drift during real-time execution. As shown in Figure 8, these timing irregularities appear in the recorded playback as narrow burst-like components in the mid- and high-frequency bands and are consistent with audible playback artifacts observed during execution. Because the proposed method decouples the SPS from the graphics loop, the corresponding regions in the proposed playback appear weaker or suppressed.

5) COMPARISON WITH ZERO-ORDER HOLD

As an additional ablation of the extrapolation stage, we compare the proposed linear extrapolation with a zero-order hold (ZOH) strategy. ZOH uses the same asynchronous SPS update schedule as the proposed method, but reuses the most recent SPS amplitude coefficients until the next SPS result becomes available. Thus, this comparison isolates the effect of amplitude prediction.

For each scene–source condition, linear extrapolation and ZOH are compared under the same asynchronous SPS update schedule and target extrapolation level using PESQ, SSIM, and SI-SNR. The results from repeated paired runs are first averaged, and the method with the higher average score is counted for each metric.

Table 6 summarizes the Extrapolation/ZOH wins for each scene and source count. ZOH is computationally simpler than linear extrapolation, but it assumes that the amplitude remains constant within an SPS interval. This assumption is often sufficient in lower-source conditions: in the 1-, 2-, and 4-source conditions, the total win difference remains within one metric comparison, and the two methods perform comparably. As the number of sound sources increases, however, source-dependent propagation changes become more complex, and the comparison shifts toward linear extrapolation. Linear extrapolation achieves Extrapolation/ZOH wins of 6/3 and 7/2 in the 8- and 16-source conditions, respectively, by predicting intermediate amplitude variations that ZOH does not explicitly represent.

V. CONCLUSION AND FUTURE WORK

In this paper, we have demonstrated that the extrapolation-based asynchronous sound propagation algorithm improves computational efficiency over a synchronous baseline while maintaining acoustic similarity. By reducing the frequency of GA propagation updates and synthesizing intermediate amplitudes via extrapolation, the proposed method produces perceptually plausible acoustic results with reduced computational load.

Experimental results across all benchmark scenes show that the proposed algorithm consistently achieves higher FPS than the baseline while preserving perceptual quality. In particular, the speedup becomes more pronounced in scenarios with many valid paths, higher scene complexity, and multiple sound sources.

Our future research will focus on developing algorithms capable of maintaining perceptually realistic audio in real time under diverse conditions, particularly for VR and AR applications. We plan to extend the current linear extrapolation to non-linear extrapolation models by leveraging a history buffer of multiple past frames. Inspired by temporal accumulation with history rejection in temporal antialiasing (TAA) [38], this approach is intended to improve robustness in dynamic environments with rapidly changing objects and viewpoints. By integrating multi-frame temporal information, we expect to reduce acoustic discontinuities and achieve a more stable trade-off between computational efficiency and perceptual audio quality across different extrapolation levels.

To ensure broad applicability, we will investigate the effectiveness of the proposed method on various platforms, including mobile devices and HMDs. We also plan to conduct comprehensive objective and subjective evaluations to characterize performance–quality trade-offs, including experiments with non-speech sound inputs to further evaluate the generality of the proposed method. Furthermore, by combining our approach with orthogonal acceleration techniques such as sound source clustering [11], [39], we aim to provide a scalable solution capable of handling dense auditory scenes.

Finally, although this paper focuses on GA-based propagation, we will study the applicability of our approach to wave-based methods and explore the integration with deep-learning-based reconstruction techniques (e.g., DLSS) to further improve real-time performance.

REFERENCES

[1] C. Cao, Z. Ren, C. Schissler, D. Manocha, and K. Zhou, "Interactive sound propagation with bidirectional path tracing," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–11, Nov. 2016.

[2] E. Lakka, A. G. Malamos, K. G. Pavlakis, and J. A. Ware, "Spatial sound rendering–a survey," *Int. J. Interact. Multim. Artif. Intell.*, vol. 5, no. 3, pp. 33–45, 2018.

[3] J.-M. Jot, R. Audfray, M. Hertensteiner, and B. Schmidt, "Rendering spatial sound for interoperable experiences in the audio metaverse," in *Proc. Immersive 3D Audio, Archit. Automot. (I3DA)*, Sep. 2021, pp. 1–15.

[4] F. Rebelo, P. Noriega, E. Duarte, and M. Soares, "Using virtual reality to assess user experience," *Human Factors*, vol. 54, no. 6, pp. 964–982, Dec. 2012.

[5] T. Potter, Z. Cvetković, and E. De Sena, "On the relative importance of visual and spatial audio rendering on VR immersion," *Frontiers Signal Process.*, vol. 2, Sep. 2022, Art. no. 904866.

[6] M. Beig, B. Kapralos, K. Collins, and P. Mirza-Babaei, "An introduction to spatial sound rendering in virtual environments and games," *Comput. Games J.*, vol. 8, nos. 3–4, pp. 199–214, 2019.

[7] N. Raghuvanshi, C. Lauterbach, A. Chandak, D. Manocha, and M. C. Lin, "Real-time sound synthesis and propagation for games," *Commun. ACM*, vol. 50, no. 7, pp. 66–73, Jul. 2007.

[8] M. T. Taylor, A. Chandak, L. Antani, and D. Manocha, "RESound: Interactive sound rendering for dynamic virtual environments," in *Proc. 17th ACM Int. Conf. Multimedia*, Oct. 2009, pp. 271–280.

[9] C. Schissler and D. Manocha, "Interactive sound rendering on mobile devices using ray-parameterized reverberation filters," 2018, *arXiv:1803.00430*.

[10] S. Liu and D. Manocha, *Sound Synthesis, Propagation, and Rendering*. San Rafael, CA, USA: Morgan & Claypool, 2022.

[11] C. Schissler and D. Manocha, "Interactive sound propagation and rendering for large multi-source scenes," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–12, Jul. 2017.

[12] E. Kim, J. Yun, W. Chung, J.-H. Nah, Y. Kim, C. G. Kim, and W.-C. Park, "Effective algorithm to control depth level for performance improvement of sound tracing," *J. Web Eng.*, vol. 21, no. 3, pp. 713–728, Feb. 2022.

[13] C. Schissler and D. Manocha, "GSound: Interactive sound propagation for games," in *Proc. Audio Eng. Soc. 41st Int. Conf., Audio Games*, 2011, pp. 1–6.

[14] N. Tsingos, E. Gallo, and G. Drettakis, "Perceptual audio rendering of complex virtual environments," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 249–258, Aug. 2004.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge, U.K.: Cambridge Univ. Press, 2007.

[16] S. S. Sahoo, C. H. Lampert, and G. Martius, "Learning equations for extrapolation and control," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4442–4450.

[17] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, "Pushing the boundaries of view extrapolation with multiplane images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 175–184.

[18] E. Kim, S. Choi, C. G. Kim, and W.-C. Park, "Multi-threaded sound propagation algorithm to improve performance on mobile devices," *Sensors*, vol. 23, no. 2, p. 973, Jan. 2023.

[19] C. Cao, Z. An, Z. Ren, D. Manocha, and K. Zhou, "BEDRF: Bidirectional edge diffraction response function for interactive sound propagation," 2023, *arXiv:2306.01974*.

[20] M. Scerbo, L. Savioja, and E. De Sena, "Room acoustic rendering networks with control of scattering and early reflections," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 32, pp. 3745–3758, 2024.

[21] S. D. Ewert, N. Gößling, O. Buttler, S. van de Par, and H. Hu, "Computationally-efficient rendering of diffuse reflections for geometrical acoustics based room simulation," *Acta Acustica*, vol. 9, p. 9, Jan. 2025.

[22] L. Savioja and U. P. Svensson, "Overview of geometrical room acoustic modeling techniques," *J. Acoust. Soc. Amer.*, vol. 138, no. 2, pp. 708–730, Aug. 2015.

[23] E. Kim, S. Choi, J. K. Kim, J. Nah, W. Jung, T.-H. Lee, Y.-K. Moon, and W.-C. Park, "An architecture and implementation of real-time sound propagation hardware for mobile devices," in *Proc. SIGGRAPH Asia Conf. Papers*, 2023, pp. 1–9.
[24] M. Kadlubiak, M. Rojek, and J. Chraponski, "GPU-accelerated simulation of elastic wave propagation," *Comput. Inform.*, vol. 37, no. 2, pp. 404–425, 2018.
[25] R. Mehra, N. Raghuvanshi, L. Savioja, M. C. Lin, and D. Manocha, "An efficient GPU-based time domain solver for the acoustic wave equation," *Appl. Acoust.*, vol. 73, no. 2, pp. 83–93, Feb. 2012.
[26] H.-K. Lee, H. Kim, D.-Y. Kim, W.-C. Park, and J. Nah, "Considerations for the acceleration structure of sound propagation on mobile devices: Kd-trees versus multi-bounding volume hierarchies," *IEEE Access*, vol. 13, pp. 188618–188629, 2025.
[27] I. Kauppinen and K. Roth, "Audio signal extrapolation—Theory and applications," in *Proc. 5th Int. Conf. Digital Audio Effects (DAFx)*, 2002, pp. 105–110.
[28] M. Fink, M. Holters, and U. Zölzer, "Comparison of various predictors for audio extrapolation," in *Proc. 16th Int. Conf. Digital Audio Effects (DAFx)*, 2013, pp. 1–8.
[29] N. Antonello, E. De Sena, M. Moonen, P. A. Naylor, and T. van Waterschoot, "Room impulse response interpolation using a sparse spatio-temporal representation of the sound field," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 10, pp. 1929–1941, Oct. 2017.
[30] H. Gamper, "Head-related transfer function interpolation in azimuth, elevation, and distance," *J. Acoust. Soc. Amer.*, vol. 134, no. 6, pp. EL547–EL553, Dec. 2013.
[31] S. Wu, S. Kim, Z. Zeng, D. Vembar, S. Jha, A. Kaplanyan, and L. Q. Yan, "ExtraSS: A framework for joint spatial super sampling and frame extrapolation," in *Proc. SIGGRAPH Asia Conf. Papers*, 2023, pp. 1–11.
[32] S. Wu, D. Vembar, A. Sochenov, S. Panneer, S. Kim, A. Kaplanyan, and L. Yan, "GFFE: G-buffer free frame extrapolation for low-latency real-time rendering," *ACM Trans. Graph.*, vol. 43, no. 6, pp. 1–15, 2024.
[33] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–half-baked or well done?" in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 626–630.
[34] *Wideband Extension To Recommendation P.862 for the Assessment of Wideband Telephone Networks and Speech Codecs*, document P.862.2, 2007.
[35] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)—A new method for speech quality assessment of telephone networks and codecs," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2001, pp. 749–752.
[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[37] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.
[38] L. Yang, S. Liu, and M. Salvi, "A survey of temporal antialiasing techniques," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 607–634, May 2020.
[39] X. Peng, K. Chen, I. Roman, J. P. Bello, Q. Sun, and P. Chakravarthula, "Perceptually-guided acoustic 'Foveation,'" in *Proc. IEEE Conf. Virtual Reality 3D User Interface (VR)*, Mar. 2025, pp. 450–460.

SUKWON CHOI (Graduate Student Member, IEEE) received the B.S. degree in computer science from Sangmyung University, Seoul, South Korea, in 2019. He is currently pursuing the Ph.D. degree with the integrated master's and Ph.D. program with the Department of Computer Science and Engineering, Sejong University, Seoul. His research interests include CNN-based processing, audio denoising, and real-time sound propagation.

EUNJAE KIM received the B.S. degree in game engineering from the Tech University of Korea, in 2017, and the Ph.D. degree in computer engineering from Sejong University, in 2024. He is currently a Lead Software Engineer with Cadence Design Systems, South Korea. His research interests include real-time sound tracing, computer graphics, computer architecture, and game engines.

UIJUN KIM received the B.S. degree in computer science from Sejong Cyber University, Seoul, South Korea, in 2022. He is currently pursuing the Ph.D. degree with the integrated master's and Ph.D. program with the Department of Computer Science and Engineering, Sejong University, Seoul. His research interests include ray tracing, rendering algorithms, and game engines.

JUNGWOONG SO received the B.S. and M.S. degrees in computer science and engineering from Sejong University, Seoul, South Korea, in 2022 and 2024, respectively. His research interests include sound tracing, image processing, and hardware architectures for image-based artificial intelligence.

WOO-CHAN PARK received the B.S., M.S., and Ph.D. degrees in computer science from Yonsei University, Seoul, South Korea, in 1993, 1995, and 2000, respectively. From 2001 to 2003, he was a Research Professor with Yonsei University. He is currently a Professor with the Department of Computer Science and Engineering, Sejong University, Seoul. His research interests include ray-tracing processor architecture, 3-D rendering processor architecture, real-time rendering, advanced computer architecture, computer arithmetic, lossless image compression hardware, and application-specific integrated circuit design.

JAE-HO NAH (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Yonsei University, Seoul, South Korea, in 2005, 2007, and 2012, respectively. He is currently an Assistant Professor with Sangmyung University, Seoul. Prior to joining academia, he was with LG Electronics. He has co-authored more than 30 technical papers published in international journals and conferences, including ACM SIGGRAPH, ACM SIGGRAPH Asia, *ACM Transactions on Graphics*, IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, *Computer Graphics Forum*, High-Performance Graphics, and *Computers and Graphics*. His research interests include ray tracing, rendering algorithms, and graphics hardware.