Audio Engineering Society

Convention Paper 10676

Presented at the 155th Convention

2023 October 25-27, New York, USA

Low-cost High-resolution Numerical Calculation of Head-related Transfer Function Using Finite-difference Time-domain Methods

Zhenxiang Hong[1], Tsubasa Kusano[2], Koki Takahashi[1], Chang Sun[1] and Kan Okubo[1]

[1] *Graduate School of Systems Design, Tokyo Metropolitan University, Japan*

[2] *Huawei Technologies Japan K.K., Japan*

Correspondence should be addressed to Zhenxiang Hong (hong.masaaki@gmail.com) and Kan Okubo (kanne@tmu.ac.jp)

ABSTRACT

The high-accuracy calculation of head-related transfer functions (HRTFs) using the voxel-based finite differential time domain (FDTD) method requires a relatively fine grid size on a staggered-structure grid. Consequently, it leads to intensive computational resource utilization and impractical time requirements. This paper presents a boundary indexing method that offers improved memory efficiency for storing model information during calculations. Compared to the full-indexing FDTD method with $O(N^3)$ memory complexity, the proposed method achieves an $O(N^2)$ memory complexity. The proposed boundary-based indexing method is considerably faster than the full indexing method. Combined implementation with the nonuniform-structured grid is discussed as a suitable method for HRTFs calculation, offering additional reductions in memory usage and calculation time. The implementation results demonstrate that the integrated approach can successfully produce HRTFs using the FDTD method with relatively low computational costs.

1 Introduction

Head-related transfer functions (HRTFs) represent the filtration process of sound propagating through the human head and ears into the eardrums. HRTFs are considered productive for enhancing spatial perception in immersive audio environments.

Personalized HRTFs are typically measured using specialized microphones in professional environments, making them difficult to access for individuals. The calculation of personalized HRTFs has been proposed using numerical simulation methods, such as the finite differential time-domain (FDTD) method [1–6] and the boundary element method (BEM) [7–9]. The FDTD method is a well-received time-domain numerical simulation method based on the Cartesian structured grid. The FDTD method calculates on the time domain and the BEM primarily simulates over the frequency domain. The BEM is a numerical technique that analyzes the data defined on the boundary of a domain, making its calculation time increase with model complexity. While the BEM is a well-established method for calculating HRTFs, the FDTD-based simulation excels in modeling scalability. This is owing to fact that the FDTD-based simulation maintains a consistent calculation time as long as the physical simulation space and grid size stay the same [10]. It facilitates the easy expansion of calculations to include complex models involving not only head models but also full-body models and surrounding objects, such as loudspeakers and tables.

In conventional FDTD-based HRTF calculations, indexing arrays that represent the shape information of the model store all the volumetric data points, known as voxels, resulting in a significant memory

requirement. Consequently, a relatively coarse mesh size is utilized owing to the machines' memory constraints, resulting in a lower upper bound in the frequency response and loss of individuality in the unique shape of each model. Instead of using a full 3D voxel array to store model information, this study employed a surface reflection-based boundary indexing algorithm that only stores the location information of the model surface, resulting in $O(N^2)$ memory complexity for the model data compared with $O(N^3)$ for the full-indexing FDTD method.

Additionally, instead of updating the coefficient arrays from the 3D voxel model information, this study adopted the surface impedance method, which overwrites the velocity at the model surface using the reflection calculated from the adjacent acoustic pressure value. Thus, the calculation speed is higher because the total memory access time is reduced. Additionally, it aligns more effectively with the HRTF calculation because wave propagation inside the human head is not considered in the discussion of HRTFs. Although the impedance reflection method has been commonly utilized for overwriting the boundary mesh, its practical implementation on intricate 3D models is limited. The developed program takes arbitrary simple-connected 3D STL files as model inputs and automatically converts mesh information into boundary-based indexing arrays required for the simulation.

To further reduce the memory usage and simulation time, the nonuniform mesh FDTD method is discussed as a suitable approach for HRTF calculation. This method utilizes a high-resolution fine grid around the complex model of the human head, where precision is crucial, while facilitating a relatively coarser mesh in the unobstructed air region, where precision can be sacrificed to improve the efficiency. The results of the uniform and nonuniform mesh FDTD methods are compared in terms of the computational speed, memory usage, and numerical variance.

2 Full-indexing calculation

In this section, several voxel-based FDTD implementation methods are discussed and compared in terms of the memory requirements and calculation speed [11,12].

Fundamentally, in a simulation mesh grid with a side length of *N*, FDTD methods that simulate acoustic propagation require at least $4O(N^3)$ memory space to represent the mesh grids of the acoustic pressure $\boldsymbol{P}$ and three-directional particle velocities $\boldsymbol{V_x}$, $\boldsymbol{V_y}$, and $\boldsymbol{V_z}$. The core algorithm for this type of generic FDTD is as follows.

Algorithm 1 Generic FDTD method

For *i,j,k* in meshgrid(*Nx,Ny,Nz*) **Do**

$$\boldsymbol{P}[i,j,k] \leftarrow \boldsymbol{P}[i,j,k] - CP_x(\boldsymbol{V_x}[i+1,j,k] - \boldsymbol{V_x}[i,j,k]) - CP_y(\boldsymbol{V_y}[i,j+1,k] - \boldsymbol{V_y}[i,j,k]) - CP_z(\boldsymbol{V_z}[i,j,k+1] - \boldsymbol{V_z}[i,j,k])$$

$$\boldsymbol{V_x}[i,j,k] \leftarrow \boldsymbol{V_x}[i,j,k] + CV_x(\boldsymbol{P}[i-1,j,k] - \boldsymbol{P}[i,j,k])$$

$$\boldsymbol{V_y}[i,j,k] \leftarrow \boldsymbol{V_y}[i,j,k] + CV_y(\boldsymbol{P}[i,j-1,k] - \boldsymbol{P}[i,j,k])$$

$$\boldsymbol{V_z}[i,j,k] \leftarrow \boldsymbol{V_z}[i,j,k] + CV_z(\boldsymbol{P}[i,j,k-1] - \boldsymbol{P}[i,j,k])$$

End For

The three-layer nested loop that iterates over 3D simulation space with size *Nx* × *Ny* × *Nz* is concisely represented as 'for *i,j,k* in meshgrid (*Nx,Ny,Nz*)'. The constant update coefficients, CP_x, CP_y, CP_z, CV_x, CV_y, and CV_z are calculated based on the material density, bulk modulus, grid size, and the length of discrete time step. The coefficients are calculated before the main calculation loop and accessed as constants. The variable in bold indicates that the variable is in the form of a 3D vector, and the non-bold variables are scalar values, such as constant coefficients and loop indexes.

However, additional memory space is required to include the 3D shape information of different materials in the simulation, which is necessary for HRTF calculations. In conventional full-indexing FDTD simulations, a 3D model is voxelized to represent the shape of the model in the calculation, and the update coefficients are changed from constants to 3D arrays. The resulting core algorithm is as follows [12]:

Algorithm 2 Model voxelization FDTD method

For *i,j,k* in meshgrid(*Nx,Ny,Nz*) **Do**

$$\boldsymbol{P}[i,j,k] \leftarrow \boldsymbol{P}[i,j,k] - \boldsymbol{CP_x}[i,j,k](\boldsymbol{V_x}[i+1,j,k] - \boldsymbol{V_x}[i,j,k])$$

$$-\boldsymbol{CP_y}[i,j,k]\big(\boldsymbol{V_y}[i,j+1,k]-\boldsymbol{V_y}[i,j,k]\big)$$
$$-\boldsymbol{CP_z}[i,j,k](\boldsymbol{V_z}[i,j,k+1]-\boldsymbol{V_z}[i,j,k])$$
$$\boldsymbol{V_x}[i,j,k]\leftarrow\boldsymbol{V_x}[i,j,k]$$
$$+\boldsymbol{CV_x}[i,j,k](\boldsymbol{P}[i-1,j,k]-\boldsymbol{P}[i,j,k])$$
$$\boldsymbol{V_y}[i,j,k]\leftarrow\boldsymbol{V_y}[i,j,k]$$
$$+\boldsymbol{CV_y}[i,j,k](\boldsymbol{P}[i,j-1,k]-\boldsymbol{P}[i,j,k])$$
$$\boldsymbol{V_z}[i,j,k]\leftarrow\boldsymbol{V_z}[i,j,k]$$
$$+\boldsymbol{CV_z}[i,j,k](\boldsymbol{P}[i,j,k-1]-\boldsymbol{P}[i,j,k])$$

End For

This adds additional $6O(N^3)$ memory space requirements to maintain the computation speed. It is technically feasible to only utilize the $1O(N^3)$ memory space for voxelized material information and calculate each update coefficient on the fly. However, the approach results in substantial amounts of repetitive calculations that significantly reduce the calculation speed.

In this simulation method, the wave propagates into as well as out of the model, resulting in additional reflections from inside the model. Their reflections may contribute to the inaccuracy of the simulation results. One of the surface-reflection-only implementations overwrites the velocity elements on the model surfaces with the corresponding impedance and adjacent acoustic pressure. However, the algorithm requires each voxel to be judged with adjacent voxels to determine whether its velocity needs to be overwritten and which side of the acoustic pressure is to be utilized. This assessment programmatically relies on computationally intensive conditional statements, resulting in considerably longer calculation times.

As an extension of voxel-based coefficient indexing, a plausible approach is to assign a coefficient to each element, combining both regular FDTD calculations and surface impedance reflection calculations into one formula.

Algorithm 3 Reflection-only model voxelization FDTD method

For *i,j,k* in meshgrid(*Nx,Ny,Nz*) **Do**

$$\boldsymbol{P}[i,j,k]\leftarrow\boldsymbol{aP}[i,j,k](\boldsymbol{P}[i,j,k]$$
$$-\boldsymbol{CP_x}[i,j,k](\boldsymbol{V_x}[i+1,j,k]-\boldsymbol{V_x}[i,j,k])$$
$$-\boldsymbol{CP_y}[i,j,k]\big(\boldsymbol{V_y}[i,j+1,k]-\boldsymbol{V_y}[i,j,k]\big)$$
$$-\boldsymbol{CP_z}[i,j,k](\boldsymbol{V_z}[i,j,k+1]-\boldsymbol{V_z}[i,j,k]))$$
$$\boldsymbol{V_x}[i,j,k]\leftarrow\boldsymbol{aCV_x}[i,j,k]\boldsymbol{V_x}[i,j,k]$$
$$+\boldsymbol{bCV_x}[i,j,k]\boldsymbol{P}[i-1,j,k]$$
$$-\boldsymbol{cCV_x}[i,j,k]\boldsymbol{P}[i,j,k]$$
$$\boldsymbol{V_y}[i,j,k]\leftarrow\boldsymbol{aCV_y}[i,j,k]\boldsymbol{V_y}[i,j,k]$$
$$+\boldsymbol{bCV_y}[i,j,k]\boldsymbol{P}[i,j-1,k]$$
$$-\boldsymbol{cCV_y}[i,j,k]\boldsymbol{P}[i,j,k]$$
$$\boldsymbol{V_z}[i,j,k]\leftarrow\boldsymbol{aCV_z}[i,j,k]\boldsymbol{V_z}[i,j,k]$$
$$+\boldsymbol{bCV_z}[i,j,k]\boldsymbol{P}[i,j,k-1]$$
$$-\boldsymbol{cCV_z}[i,j,k]\boldsymbol{P}[i,j,k]$$

End For

The additional coefficient arrays $\boldsymbol{aP}$, $\boldsymbol{aCV_{x,y,z}}$, $\boldsymbol{bCV_{x,y,z}}$ and $\boldsymbol{aCV_{x,y,z}}$ in the *x*, *y*, and *z* directions are pre-generated based on the model voxel. However, while this method is surface-reflection only, the $10O(N^3)$ memory usage makes it infeasible for calculating large-scale models with a high resolution.

3 Boundary-based calculation

Considering the advantages and disadvantages of voxel-based methods, a boundary-based indexing method can be considered a viable approach for high-resolution reflection-only FDTD calculations, offering low additional memory requirements and relatively high speed.

By scanning the triangular mesh of the 3D model on the coordinates of the voxels, the model can be sliced into multiple 2D planes with connected edges that represent the boundary between the inside and outside of the model. Subsequently, the algorithm checks the intersections between the boundary edges and the voxel's Cartesian grid, and the intersection coordinates are transformed into a mesh grid index and stored sequentially before the following FDTD computation. The same process is performed in three directions to generate the arrays used separately for the calculations of $\boldsymbol{V_x}$, $\boldsymbol{V_y}$, and $\boldsymbol{V_z}$. Such model-boundary-based index arrays are initiated before the main time-loop calculation and exhibit $3O(N^2)$ memory complexity.

During each time step of the simulation, a 3D generic calculation of the particle velocity is performed.

Algorithm 4.1 Boundary index FDTD method-general velocity

For *i*,*j*,*k* in meshgrid(*Nx*,*Ny*,*Nz*) **Do**

$$\boldsymbol{V_x}[i,j,k] \leftarrow \boldsymbol{V_x}[i,j,k] + CV_x(\boldsymbol{P}[i-1,j,k] - \boldsymbol{P}[i,j,k])$$

$$\boldsymbol{V_y}[i,j,k] \leftarrow \boldsymbol{V_y}[i,j,k] + CV_y(\boldsymbol{P}[i,j-1,k] - \boldsymbol{P}[i,j,k])$$

$$\boldsymbol{V_z}[i,j,k] \leftarrow \boldsymbol{V_z}[i,j,k] + CV_z(\boldsymbol{P}[i,j,k-1] - \boldsymbol{P}[i,j,k])$$

End For

Herein, we present a simple impedance boundary implementation. However, additional complex boundary representations can be implemented via overwriting. The need for determining which side of the acoustic pressure to use is eliminated as long as the simulation model is simple-connected without any "holes" or 2D "handles," as each grid line consistently passes from the inside to the outside of the model. Even when the model is a hollow shell or has concave features, there are even numbers of intersections, and the direction flips successively for each intersection.

The coordinates of the voxels that correspond to the boundary location of the model are sequentially read, and these voxels are overwritten with the surface impedance reflection calculation. The general process of $\boldsymbol{V_x}$ is shown, whereas the calculations of $\boldsymbol{V_y}$ and $\boldsymbol{V_z}$ follow the same process as their respective boundary index arrays. For the calculation of $\boldsymbol{V_x}$, array **boundaryIndexArrayX** is used. *L* and *R* represent the position index of two sides of the simulation model. $Z0$ denotes the acoustic impedance of the model.

Algorithm 4.2 Boundary index FDTD method-velocity on model boundary

For [*L*,*j*,*k*] and [*R*,*j*,*k*] in **boundaryIndexArrayX** (~,*j*,*k*) **Do**

$$\boldsymbol{V_x}[L,j,k] \leftarrow \frac{1}{Z0}\boldsymbol{P}[L-1,j,k]$$

$$\boldsymbol{V_x}[R,j,k] \leftarrow -\frac{1}{Z0}\boldsymbol{P}[R,j,k]$$

End For

Following velocity overwriting, the 3D generic calculation of the particle velocity is executed.

Algorithm 4.3 Boundary index FDTD method-general acoustic pressure

For *i*,*j*,*k* in meshgrid(*Nx*,*Ny*,*Nz*) **Do**

$$\boldsymbol{P}[i,j,k] \leftarrow \boldsymbol{P}[i,j,k] - CP_x(\boldsymbol{V_x}[i+1,j,k] - \boldsymbol{V_x}[i,j,k]) - CP_y\big(\boldsymbol{V_y}[i,j+1,k] - \boldsymbol{V_y}[i,j,k]\big) - CP_z(\boldsymbol{V_z}[i,j,k+1] - \boldsymbol{V_z}[i,j,k])$$

End For

The boundary index arrays are utilized again to overwrite waves that propagate inside the model to zero; this does not affect the calculation results but always keeps the inside of the model constantly at zero, thus improving the simplicity and clarity of the animation generated in the visualization process.

Algorithm 4.4 Boundary index FDTD method-acoustic pressure on model boundary

For [*L*,*j*,*k*] and [*R*,*j*,*k*] in **boundaryIndexArrayX** (~,*j*,*k*) **Do**

$$\boldsymbol{P}[L,j,k] \leftarrow 0$$

$$\boldsymbol{P}[R-1,j,k] \leftarrow 0$$

End For

With a time complexity of $O(N^2)$, the overwrite process adds negligible computational time to the overall generic FDTD calculations. The generation time for the boundary index array depends on the complexity of the simulated 3D model and mesh grid size. However, as this process is only executed once, it does not impact the performance of the main calculation loop.

4 Nonuniform mesh implementation

To further reduce memory usage and calculations, a combined implementation of boundary index overwriting and a nonuniform mesh for the structured Cartesian grid is discussed in this section [13,14]. The nonuniform mesh method employs varying mesh sizes at different locations, aligning perfectly with the requirements of the HRTF simulation. In a pure HRTF calculation, a fine mesh size is desirable around the human head, while the remaining areas

can be represented with coarser mesh sizes as they represent unobstructed air.

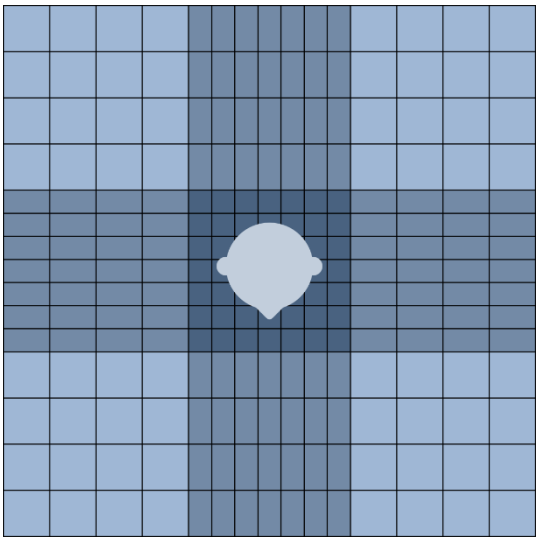

Figure 1. HRTF calculation schematic of the nonuniform mesh

Figure 1 illustrates an overhead view of the 2D scheme utilized in the nonuniform-mesh HRTF calculation. Different colors are used to represent varying mesh sizes.

In real-life measurements of HRTFs, sound sources are placed 1–2 m from the human hearing center to avoid the proximity effect. In the FDTD simulation, a similar 3D space was required to match the real-life measurements.

Because the memory complexity is $O(N^3)$ for the FDTD algorithm, even a modest increase in the mesh size substantially reduces memory usage and calculation time. A slightly high numerical inaccuracy is inevitable when waves propagate between meshes of different sizes. However, the inaccuracy in the HRTF diminishes exponentially as the disparity in mesh sizes reduces. The actual level of inaccuracy of the implementation is described in Section 6.

5 Computational cost of boundary indexing method

Several experiments were conducted to determine the actual reduction in the computational costs of the boundary indexing method compared with the full-indexing method. Both algorithms were implemented in C++, and OpenMP was used as a parallel tool [15,16]. The calculation utilized a high-resolution, high-accuracy 3D head model with open ear canals.

Mesh size	1 mm
Physical space	2.8 m × 2.8 m × 2.8 m
Real-life simulation time	5 ms
Memory usage	326 GB
Calculation time	4.6 h
CPU	Intel(R) Xeon(R) Gold 6326 CPU @ 2.90 GHz
Memory speed	3200 MT/s

Table 1. Simulation settings and machine specifications

Table 1 presents the simulation settings and hardware configurations.

The HRTFs were calculated on a 1 mm mesh in a 2.8 m × 2.8 m × 2.8 m physical space where the human bust model was located at the center. Leveraging the duality in sound propagation, we placed the sound sources in the ear canal, and virtual microphones were positioned on a 1.1 m radius hemisphere at 30° intervals to record the head-related impulse response (HRIR). The simulation modeled a real-life period of 5 ms. The calculation utilized single-precision float-type variables and required approximately 326 GB of memory; the calculation time was approximately 4.6 hours.

Figure 2 illustrates the simulated HRTF at the horizontal plane with a 30° interval. The 0° position resembles the front of the human head, while the 90° position represents the left-hand side, and the 270° position indicates the right-hand side. The darker color indicates positions closer to the respective ear, whereas the lighter color indicates positions farther from the corresponding ear.

Several test calculations were performed to generalize the memory usage and calculation time of the boundary index method for the uniform mesh case. Smaller-scale calculations were executed and measured, and the data were used to estimate the

memory usage and calculation time of the resource-intensive simulations.

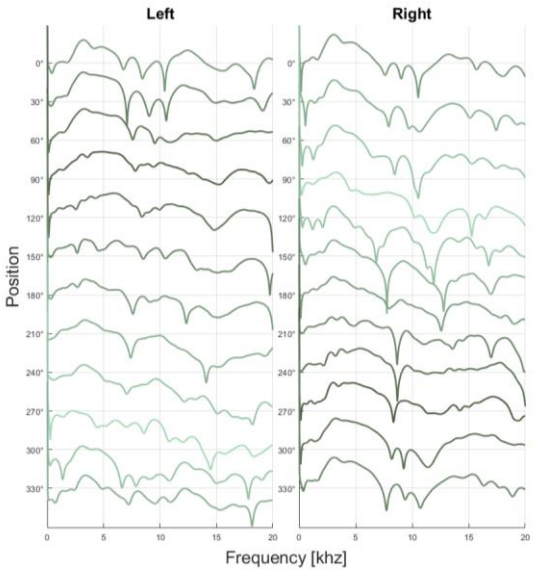

Figure 2. Simulated HRTF

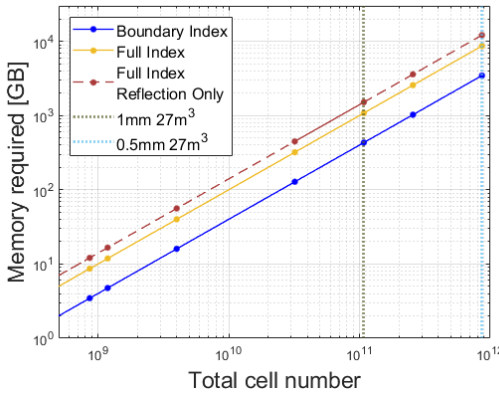

Figure 3. Memory usage of various FDTD methods

Figure 3 illustrates the theoretical memory usage of different calculation methods introduced in Section 2: the boundary indexing method (Algorithm 4), full-indexing method (Algorithm 2), and reflection-only full indexing method (Algorithm 3).

In each algorithm, the memory usage increased linearly with the total cell number. The total number of cells was exponentially determined using the simulation space and mesh size. The boundary indexing method requires less memory algorithmically; therefore, the savings in memory consumption become more evident in situations such as high-resolution HRTF calculations, where a finer mesh size and larger simulation space are preferred.

Table 2 lists the calculation times of the boundary index calculation under different cell sizes and simulation spaces in the volume.

	5 mm	3 mm	1 mm	0.5 mm
1 m^3	0.0006	0.0031	0.2149	3.4978
8 m^3	0.0032	<u>0.0163</u>	<u>1.1425</u>	<u>18.596</u>
27 m^3	0.0099	<u>0.0509</u>	<u>3.5608</u>	<u>57.958</u>

Table 2. Calculation time of boundary indexing FDTD method [h]

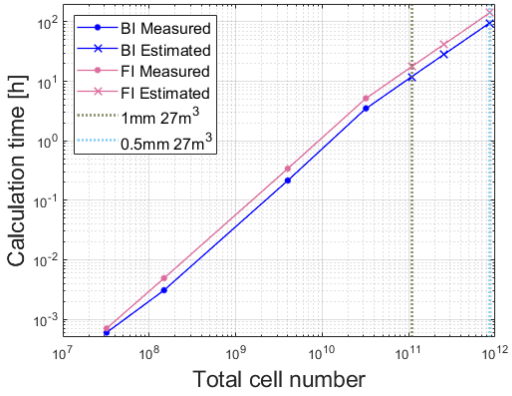

Figure 4. Calculation time of FDTD methods

The column headers represent different cell sizes, and the row headers represent different simulation spaces. The data without any underlining represent the real-life measurements, whereas the underlined values represent the estimated ones.

While the calculation time increases linearly as the total cell number increases, the predicted result may not be strictly linear because other factors, such as compiler internal settings, can also affect the calculation speed.

Among the algorithms introduced in Sections 2 and 3, Algorithm 1 cannot represent the model shapes, whereas Algorithm 2 allows sound to propagate through the model. To achieve a surface reflection-only simulation, Algorithm 3 (reflection-only full indexing (FI)) and Algorithm 4 (boundary-based indexing (BI)) were implemented. Small-scale tests were conducted, and the execution times were recorded. Calculations involving higher mesh usage were estimated through linear regression. Figure 4 illustrates the results in terms of the calculation time.

As shown in Figure 4, the calculation times of the boundary indexing method are shorter than those of the full-indexing method. It can be attributed to the fewer memory accesses in each calculation cycle.

6 Calculation settings and results of nonuniform mesh

A combination of the boundary indexing method and a nonuniform mesh was implemented in C++ combined with the former boundary index method. All the simulation settings and hardware specifications were the same as those used in the boundary-index-only calculation.

In the combined implementation with the boundary index method, we used the same 2.8 m × 2.8 m × 2.8 m physical space. The center fine mesh area was fixed at 0.4 m × 0.4 m × 0.4 m with a 1 mm cell size. The cell size of the coarser perimeter mesh area was varied from 1 to 3.

Table 3 lists the measured calculation time and memory usage for different coarse-to-fine ratios. A small increase in the coarse mesh size can result in a notable reduction in the computational cost.

Similar to the boundary index method, the calculation time and memory usage of the nonuniform-mesh FDTD increased linearly with the total cell number. In the uniform-mesh FDTD, the total number of cells is determined by the cell size and the physical space being simulated. In nonuniform mesh FDTD, the cell number is additionally influenced by the percentage of finer meshes in the simulation space and the coarse-to-fine ratio, which represents the size ratio between the coarser and finer meshes. Theoretical memory usage can be calculated from the four parameters that influence the computational costs of the nonuniform-mesh implementation.

Coarse-to-fine ratio	Calculation time [s]	Memory [GB]
×1	16289	326
×1.25	9282	190
×1.5	6024	123
×2	3076	62
×3	1327	27

Table 3. Measured computational costs of nonuniform mesh

Figure 5 illustrates some of the possible settings of the nonuniform mesh method and the corresponding theoretical memory usage. The solid lines correspond to a fine mesh size of 1 mm, whereas the dashed lines represent a 0.5 mm mesh size. In the legend, the cell size of the fine grid at the center is denoted by d_f. The fine ratio r_f indicates the percentage of the edge length of the fine mesh area in the entire simulation space with an edge length of 2.8 m. In all the nonuniform-mesh HRTF calculations performed in this study, the edge length was set to 0.4 m as the dimension of the center fine mesh, corresponding to a fine ratio of approximately 15%.

Figure 6 shows the measured calculation time with a fine center mesh size of 1 mm and the estimated calculation time for a 0.5 mm center mesh size.

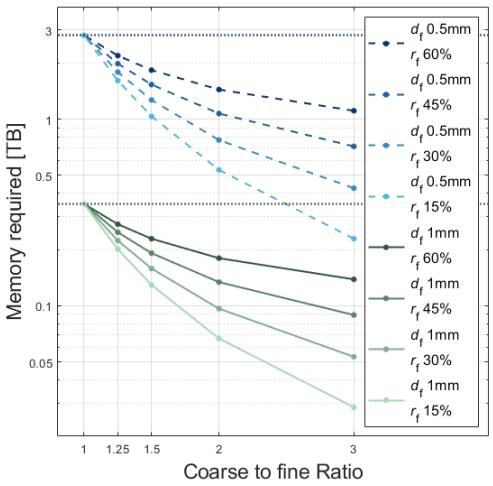

Figure 5. Memory usage of nonuniform mesh

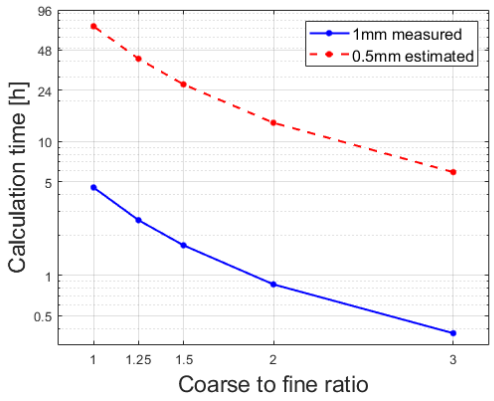

Figure 6. Calculation time of the nonuniform mesh

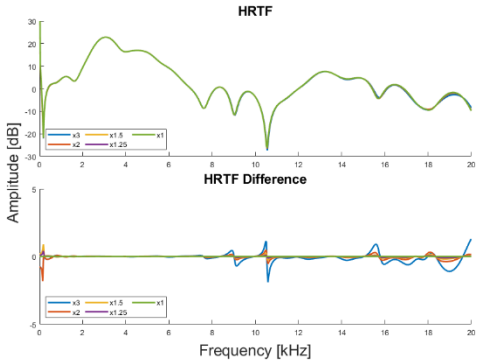

Figure 7.1 HRTF comparison at $(\theta, \varphi) = (0, 0)$ (horizontal front)

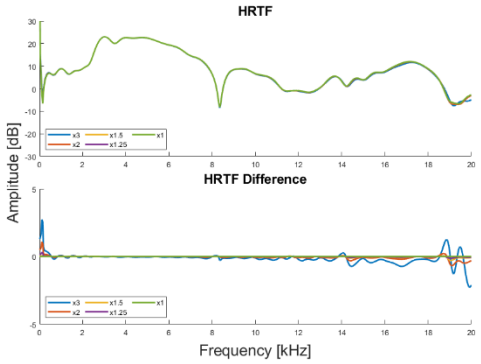

Figure 7.2 HRTF comparison at $(\theta, \varphi) = (0, 270)$ (horizontal right)

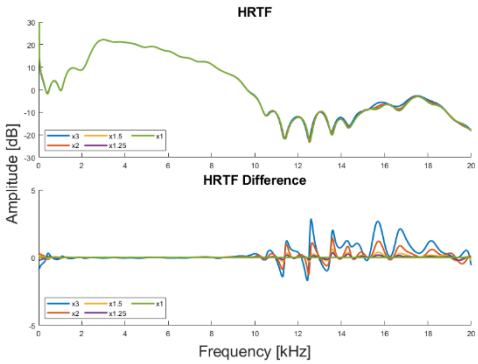

Figure 7.3 HRTF comparison at $(\theta, \varphi) = (60, 300)$ (upper-right corner)

While a nonuniform mesh significantly reduces memory usage and calculation time, a certain level of inaccuracy is present. This is partly because a small amount of reflection occurs when a soundwave propagates between fine mesh and coarse mesh. The higher numerical variance in the coarser mesh further contributed to this inaccuracy.

Figures 7.1–7.3 illustrate the HRTFs measured in the right ear of the same model at the same angle for different coarse-to-fine ratios, along with their differences from the results calculated with the uniform mesh, indicated by a coarse-to-fine ratio of 1. A polar coordinate system was used to represent the exact position of each HRTF. θ and φ represent the inclination and azimuth angles, respectively. A zero value in the azimuth angle corresponds to the front direction of the head model. The errors in both cases increased with the coarse-to-fine ratio. However, these inaccuracies are relatively minor and primarily occur at high frequencies.

To comprehensively assess the efficacy of the proposed FDTD-based algorithm, a thorough comparative analysis was carried out, involving actual measurements and the widely-adopted BEM-based simulation technique.

This evaluation based on the KEMAR dummy head model. Measurement results were obtained in an anechoic room using the KEMAR head. The ear canals of the KEMAR model were blocked, and custom omnidirectional microphones were placed at the ear canal entrances to record the HRIRs.

On the other hand, the simulation-based approaches were executed on a detailed 3D model of the KEMAR dummy head, generated through advanced 3D scanning technology. The BEM-based simulation

results were obtained using "mesh2HRTF", a well-established HRTF simulation toolbox implemented using the boundary element method. Meanwhile, the proposed FDTD-based method was explored under three distinctive settings:

FDTD 1:1, employing a uniform mesh with a mesh size of 1 mm.
FDTD 1:2, employing a nonuniform mesh featuring a central fine mesh size of 1 mm and a peripheral coarse mesh size of 2 mm.
FDTD 1:3, employing a nonuniform mesh featuring a central fine mesh size of 1 mm and a peripheral coarse mesh size of 3 mm.

Figure 8.1 illustrates the HRIRs obtained using these different methods, while Figure 8.2 shows their corresponding normalized power spectra.

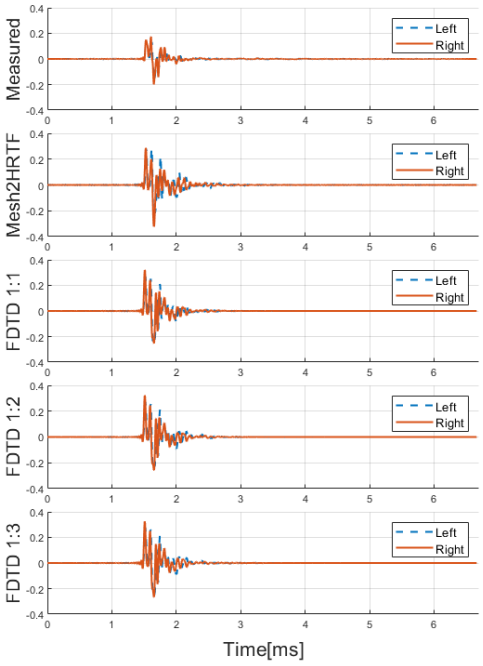

Figure 8.1 HRIRs comparison of different methods

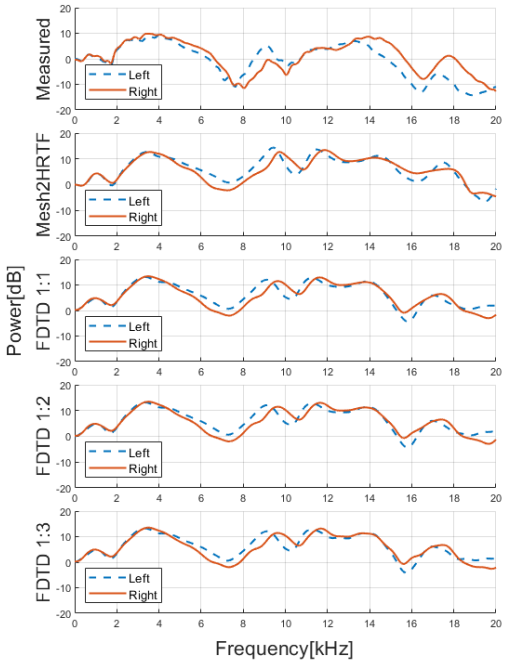

Figure 8.2 Power spectra comparison of different methods

In our evaluation, both the proposed FDTD-based method and the established BEM approach exhibit remarkable similarities in replicating HRTF characteristics. However, it should be noted that the results of both simulation methods display some amplitude discrepancies when compared with actual measurements.

7 Conclusions

In this study, we presented a low-cost, high-resolution HRTF numerical simulation method using a modified FDTD method that utilizes boundary indexing to perform reflection calculations, which considerably reduces memory usage without compromising the computational speed. The results of our implementation demonstrated that the method enables FDTD calculations of personalized HRTFs at a finer grid resolution, theoretically resulting in improved accuracy in higher frequency bands.

Additionally, the usage of a nonuniform mesh is also reviewed as a measure to further reduce the computational cost, enabling easier access to FDTD simulations of personalized HRTFs on relatively lower-spec hardware while still maintaining reasonable accuracy in higher frequency bands.

Compared with traditional methods that solely simulate the human head, the proposed methods allow for the inclusion of additional models, such as surrounding objects in the numerical simulations. The methods introduced herein also have a wide range of applications, such as simulating the propagation of acoustic waves in various rooms of arbitrary shapes.

To improve the accuracy of our method, further work could involve the implementation of the multistep nonuniform mesh method and employing higher-order difference calculations at the mesh border. These may help in reducing the numerical inaccuracy while retaining a relatively low memory usage.

References

[1] H. Takemoto, "Computer simulation of head-related transfer functions" *The Journal of the Acoustical Society of Japan*, vol. 73, no. 3, 166–172 (2017).

[2] H. Takemoto, P. Mokhtari, "Mechanism for generating peaks and notches of head-related transfer functions in the median plane" *Journal of the Acoustical Society of Am*erica, vol. 132, no. 6, pp. 3832–3841 (2012).

[3] P. Mokhtari, H. Takemoto, R. Nishimura, H. Kato, "Comparison of simulated and measured HRTFs: FDTD simulation using MRI head data" *Journal of the Audio Engineering Society,* 2007.

[4] P. Mokhtari, H. Takemoto, R. Nishimura, H. Kato, "Vertical normal modes of human ears: Individual variation and frequency estimation from pinna anthropometry" *Journal of the Acoustical Society of America* vol. 140, pp. 814–831 (2016).

[5] M. Nakazawa, A. Nishikata, "Development of sound localization system with tube earphone using human head model with ear canal" *IEICE Trans. Fundam.,* vol. *E88-A*, pp. 3584–3592 (2005).

[6] T. Xiao, Q. H. Liu, "Finite difference computation of head-related transfer function for human hearing" *Journal of Acoustical Society of America* vol. 113, pp. 2434–2441 (2003).

[7] Y. Kahana, P. A. Nelson, "Numerical modelling of the spatial acoustic response of the human pinna" *Journal of Sound and Vibration* vol. 292, pp. 148–178 (2006).

[8] Y. Kahana, P. A. Nelson, "Boundary element simulations of the transfer function of human heads and baffled pinnae using accurate geometric models" *Journal of Sound and Vibration* vol. 300, pp. 552–579 (2007).

[9] B. F. G. Katz, "Boundary element method calculation of individual head-related transfer function. I. Rigid model calculation" *Journal of Acoustical Society of America*. vol. 110, pp. 2440–2448 (2001).

[10] Y. Li, J. Meyer, T. Lokki, J. Cuenca, O. Atak, W. Desmet, "Benchmarking of finite-difference time-domain method and fast multipole boundary element method for room acoustics" *Applied Acoustics,* vol. 191, (2022).

[11] A. Taflove, "Review of the formulation and application of the finite-difference time-domain method for numerical modeling of electromagnetic wave interactions with arbitrary structures" *Wave Motion,* vol. 10, no. 6, pp. 547–582.

[12] A. Taflove, S. C. Hagness, "*Computational electrodynamics: the finite-difference time-domain method*" Artech House on Demand (2005).

[13] H. Jiang, H. Arai, "3D FDTD Analysis by Using Nonuniform Mesh" *ICMMT'98* (1998).

[14] A. Taflove, "The finite-difference time-domain method for numerical modeling of electromagnetic wave interactions" *Electromagnetics,* vol. 10, no. (1-2), pp. 105–126 (1990).

[15] S. Sakamoto, T. Seimiya, H. Tachibana, "Visualization of sound reflection and diffraction using finite difference time domain method" *Acoustical Science and Technology*, vol. 23 (2002).

[16] T. Sakuma, S. Sakamoto, T. Otsuru, T. *Computational Simulation in Architectural and Environmental Acoustics: Methods and Applications of Wave-Based Computation*. Springer Japan (2014).