Visualization of Large-scale CFD Simulation Results Using Distributed Particle-Based Volume Rendering

Naohisa Sakamoto[1], Hiroshi Kuwano[2], Takuma Kawamura[2], Koji Koyamada[1], and Kazunori Nozaki[3]

[1] Center for the Promotion of Excellence in Higher Education, Kyoto University
[2] Graduate School of Engineering, Kyoto University
[3] Cybermedia Center, Osaka University

ABSTRACT

In this paper, we present a technique for visualizing a large-scale irregular volume dataset that is generated from an LES-based CFD simulation. Since our computational mesh was too large for a single computational node, it was divided into multiple regions; moreover, the resulting file was comprised of various irregular volume datasets. In order to cope with the multiple volume datasets, we extended our particle-based volume rendering (PBVR) technique so to fit the distributed computing environment. We applied our distributed PBVR technique to an LES-based CFD simulation, which explores dental fricative sound sources in order to confirm the effectiveness of the technique.

Keywords: predictive simulation of dental fricative sounds, particle-based volume rendering, large irregular-grid volume data.

1. INTRODUCTION

The sound source of dental fricatives is regarded as a downstream obstacle in the oral airflow field in which turbulence is dominant. In contrast, a vowel's sound is derived from vibrations of the vocal cords as a result of air flowing past them [1, 2]. In oral therapies that treat speech disorders, the modification of oral morphological features takes place as a consequence of changing the spatial positioning of the jaw. This includes maxillofacial orthodontic therapies [3], prosthetic treatments [4] and removable sports mouth guards. Alterations in oral morphology may impact the characteristics of resonance as well as the magnitudes and locations of sound sources. Computational fluid dynamics (CFD) is a powerful tool for conducting research into and developing tools for the treatment of speech disorders. In particular, dental fricatives involve a different method of sound production from that of the vowel. CFD can be used to confirm that the sound associated with a dental fricative is due to a downstream obstacle in the oral airflow field.

In order to construct the morphological geometries of an oral cavity, multiple image slices of various soft tissues, such as lips and the tongue, and hard tissues, including teeth, were acquired using a Cone Beam CT (CBCT) scanner (Figure 1 (a)). The resulting isosurfaces were subsequently extracted from a volume dataset defined as a set of slices. The surfaces were converted to a Non-Uniform Rational B-Spline (NURBS) surface. First, computational mesh cells were constructed as a hexahedron using a grid generation system. Thereafter, each mesh cell was subdivided into 8 or 27 cells, because it was necessary to adequately represent the narrow (1–3 mm) space that exists between the upper and lower teeth when a dental fricative is pronounced [5, 6]. Since the resulting mesh was composed of 72 million hexahedral cells, which is too large for a single computational node, it was divided into multiple regions so that a distributed LES-based CFD calculation could be performed. Thus, the simulation results are actually comprised of multiple sets of results.

The obstacle sound source can be explained in terms of the impact of turbulence on the obstacle's surface [7]. The sound source is located somewhere on the surface, and if the shape of surface is altered, the distribution of the source should change commensurately. It is well known that the sound generated from the interaction between the surface and the vortexes is louder than the one derived from interactions among vortexes themselves in the flow field. Compressible Navier-Stokes equations that are derived using the Direct Navier-Stokes (DNS) method can predict broadband noise. However, researchers have found that it is necessary to use an impracticably large number of meshes, as consistent with the Reynolds Number value, and an extremely fine grid. Since acoustic problems are highly time-dependent,

Corresponding Author: Prof. Naohisa Sakamoto, e-mail: naohisas@viz.media.kyoto-u.ac.jp

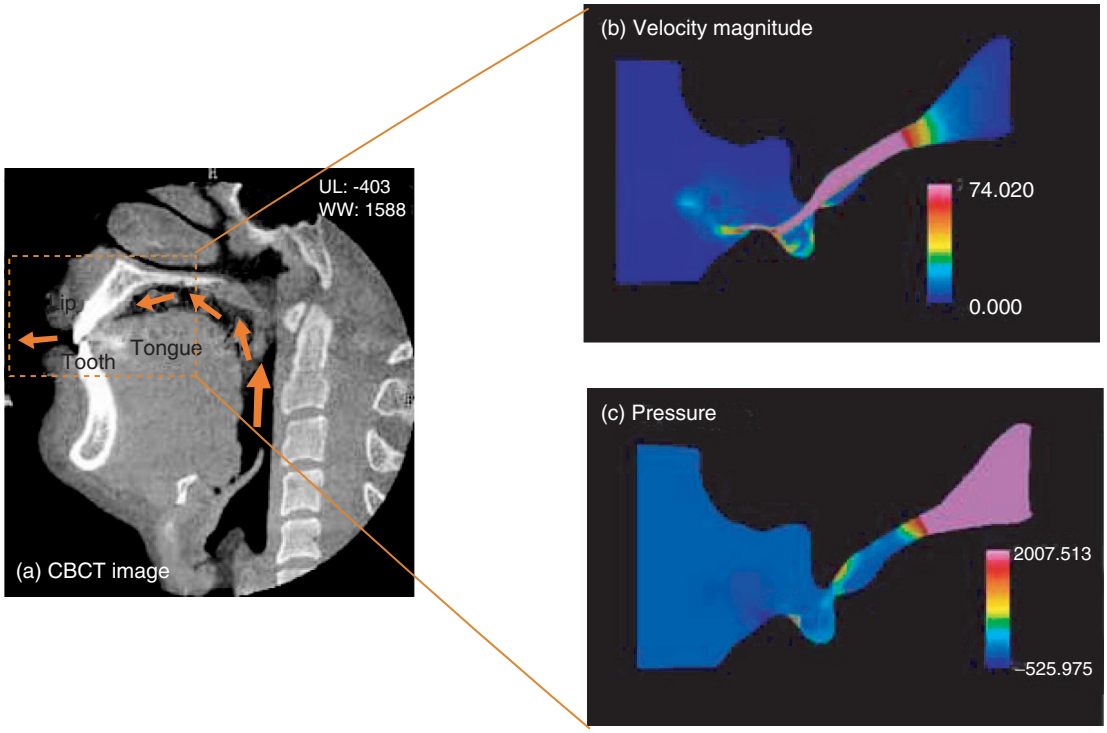

Figure 1. CBCT image and previous visualization results. (a) Mid-sagittal vocal tract image taken by CBCT; the dotted orange square indicates the targeted region of CFD simulation. (b) Velocity magnitude and (c) pressure at mid-sagittal plane of the oral cavity.

Reynolds-averaged (time-averaged) Navier-Stokes (RANS) equations are inadequate for time series analyses because they are generally time independent. The Large Eddy Simulation (LES) method is appropriate for addressing these problems. The LES method uses a Sub-Grid Scale (SGS) model to consider vortex affects for smaller grids. Because the LES uses a time-independent and space-averaged model, it is appropriate for analyzing time-dependent vortexes. However, to ensure LES accuracy, the mesh size normal to the boundary surface must be as small as possible in the turbulence boundary layer. Therefore, the LES calculation generates a rather large-scale irregular volume dataset.

Researchers have conventionally used surface-based visualization techniques to understand the velocity and pressure of dental fricative sound on a sectional slice or boundary geometry (Figures 1 (b) and (c)). Although visualization results can provide useful information, these results are limited to two-dimensional space. Before an adequate surface is determined, it is often necessary to grasp the spatial distribution of related physical quantities. Although a volume rendering technique is desirable for such purposes, it is difficult for a conventional volume rendering technique to account for a large-scale irregular volume dataset, since the technique requires the dataset to be stored in a single computational node in order to calculate the visibility of mesh cells at each viewpoint. It is apparent that such requirements are impractical for our study, since the computational space had to be divided into multiple regions so that a distributed computational resource could be utilized. Thus, Particle-based Volume Rendering (PBVR) is a good candidate for large-scale irregular volume rendering, since it requires no visibility sorting and is well suited to cell-by-cell processing, which does not require an entire dataset to be stored in a single memory space.

In this paper, we propose a distributed implementation of PBVR for visualizing a large-scale irregular volume dataset generated from a distributed CFD calculation. We apply the proposed technique to CFD simulation results generated from modeling an oral airflow field in order to confirm its effectiveness.

2. RELATED WORK

The development of techniques for rendering irregular volume datasets has remained a major challenge for visualization scholars. Such datasets consist mainly of scalar data defined on collections of irregularly-ordered cells with shapes that are not necessarily orthogonally cubic.

The irregular volume rendering technique can be image-order or object-order algorithm. The image-order algorithm requires the entire volume dataset to be stored in the main memory, including cell adjacency information. This requirement might be a disadvantage when processing a large dataset. The object-order algorithm may be promising, since its cell-by-cell approach can be easily implemented. In general, a volume rendering technique utilizes a density emitter model in which the 3D scalar field is characterized as a varying density emitter with a single level of scattering. This model was proposed by Sabella [8] and is related to a particle system in which the particles are sufficiently small and of low albedo. A conventional volume rendering technique models the density of particles, not the particles themselves, which makes the visibility sorting of volume cells indispensable in the rendering algorithm.

Since Shirley et al. [9] proposed a Projected Tetrahedra (PT) algorithm for generating a volume-rendered image by using tetrahedral cells, many irregular volume rendering techniques based on the object-order approach have been proposed. In Shirley et al.'s [9] algorithm, each tetrahedral cell is projected onto the screen in the order of visibility, from back to front, to develop a semitransparent image. The work of Williams [10] has also led to a considerable amount of research on visibility sorting techniques. Currently, in irregular volume rendering, one of the major approaches is to relax the processing requirement for visibility sorting or to develop a technique without sorting.

To address the former approach, Callahan et al. [11] have developed an integrated visibility sorting technique known as Hardware-Assisted Visibility Sorting (HAVS), in which the centroids of the cell faces are first sorted in order to roughly sort by visibility, and pixel fragments generated from rasterized faces are used along with the k-buffer in order to increase accuracy. Although HAVS achieved 1.3 fps for 1.4 million tetrahedra, it generates some artifacts when the k-buffer is not large enough. It is difficult to determine the optimum value for k. The memory space for the sorting of cell faces is about twice as much as of the number of tetrahedral cells.

Anderson et al. [12] proposed a point-based technique to render irregular volumes. They represented a tetrahedral cell as a point primitive at the center, which was followed by the rendering and compositing of the represented point primitives. Like HAVS, this technique requires the sorting of all point primitives. Although this technique has achieved 5.3 fps for 1.4 million tetrahedra and 0.3 fps for 6.3 million tetrahedra, there may be artifacts in the rendered image when point primitives are rasterized as screen-aligned squares.

Roettger et al. [13] have pointed out that the memory bandwidth required for visibility sorting becomes its limiting factor, and thus, they proposed an algorithm that requires no visibility sorting for cells of irregular volume. However, since their optical model considers only emissions, its application is limited to the visualization of gaseous phenomena. Csebfalvi et al. proposed a sorting-free volume rendering technique [14, 15] that can be categorized as an X-ray volume rendering approach. That is, their optical model considers only absorption. Zhou et al. [16] proposed a sorting-free rendering technique that is implemented with additional terms to help provide enhanced depth cues without visibility sorting. Although it has achieved 20 fps for 17.6 million tetrahedra, their optical model does not consider absorption effects.

To address these problems among current sort-free volume rendering techniques, we returned to the density emitter model in order to present a basic framework for an approach called the PBVR technique, which represents the 3D scalar fields as a set of particles and considers both emission and absorption effects. Particle density is derived from a user-specified transfer function and is utilized to estimate the number of particles to be generated in a given volume dataset. Since the particles can be considered fully opaque, no visibility sorting processing is required during rendering processes, and this is advantageous from a distributed processing perspective.

3. OVERVIEW OF PBVR

First, we use a ray casting algorithm for volume rendering to discuss on the image quality of PBVR. Using the volume ray casting algorithm, we subdivide the viewing ray into segments so that particle luminosity and density in the k-th ray segment can be regarded as constants, i.e., c_k and α_k. Brightness can be calculated as:

$$B_0 = \sum_{i=1}^{n} C_i \times \left(\alpha_i \prod_{j=1}^{i-1} (1-\alpha_j) \right). \tag{1}$$

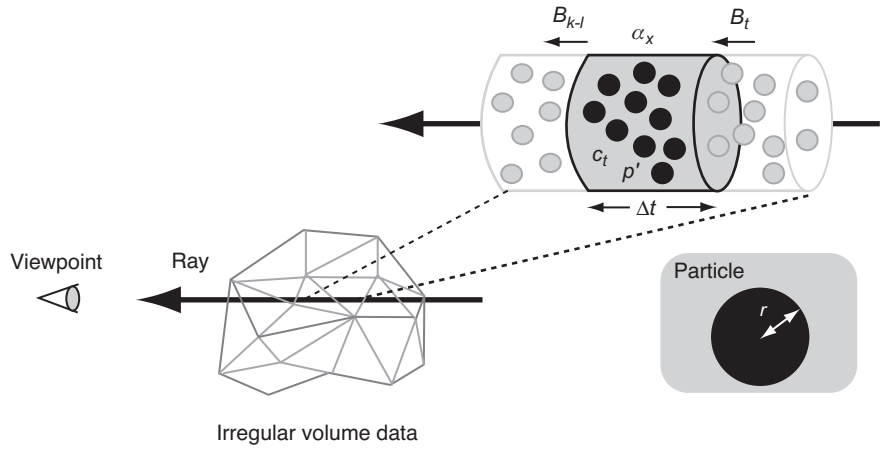

Figure 2. Density emitter model.

In the density emitter model shown in Figure 2, the opacity is described as follows:

$$\alpha_k = 1 - e^{-\rho' \Delta t}. \tag{2}$$

Here, Δt indicates length, ρ' describes the number of particles in a unit length, and $\rho' \Delta t$ expresses the number of particles in a ray segment. This model assumes that the number of particles follows a Poisson distribution, and the expression therefore represents the possibility that the segment contains some particles.

Note that the computational complexity of Equation 1 is $O(n^2)$. If we consider an intermediate brightness value B_k in the interval $[t_k, t_n]$, we obtain the following recurrence formula for the back-to-front accumulation:

$$B_{k-1} = c_k \alpha_k + (1 - \alpha_k) B_k. \tag{3}$$

This shows that the computational complexity can be reduced to $O(n)$ if the sampling points are sorted according to visibility order. Most volume rendering techniques employ back-to-front accumulation. Although visibility sorting has contributed to a performance improvement in the brightness calculation, it can unfortunately become a bottleneck when a large irregular volume dataset is rendered.

To develop a volume rendering technique that does not required visibility sorting, we represented a 3D scalar field as a set of emissive and opaque particles such that only depth comparison and no alpha blending would be required during rendering calculations. This proved to be advantageous for subsequent distributed processing. Particle density was derived from a user-specified transfer function that converts a scalar data value to an opacity data value and describes the probability that a particle is present at a given point in space.

Our particle-based volume rendering approach is comprised of three phases: particle generation, particle projection, and sub-pixel processing. The first phase involves constructing a density field and generating particles consistent with the density function. The second phase involves projecting particles onto an image plane. The third phase involves dividing a single pixel into multiple sub-pixels so that a particle is stored as precisely as possible and, in addition, calculating a final brightness value by averaging sub-pixel values [17].

In the particle generation phase, we used a particle generation method based on the Metropolis algorithm [18], which is an efficient Monte Carlo technique widely used in chemistry and physics. Since the method uses a ratio of density at the current position versus that at the candidate position, our original technique does not employ a density that represents a number of particles in a unit volume. Instead, we employed a density metric that expresses the number of particles in a unit length, and we approximated this value using opacity. The user must specify the total number of particles, but this also makes it difficult to generate an image that is comparable to the volume ray casting.

To solve this problem, we introduced particle size explicitly into the particle model. Once we were able to generate an image comparable to the volume ray casting image, it was easy to determine a

minimum sub-pixel level for which a given criterion could be satisfied with respect to the difference between the PBVR and volume ray casting images.

3.1. Particle Modeling

In order to use a density that represents the number of particles in a unit volume, in our particle model, we replace ρ' with $\pi r^2 \rho$, which is a projection area multiplied by the density of particles in a unit volume. Opacity is then expressed as:

$$\alpha = 1 - \exp(-\pi r^2 \rho \Delta t). \tag{4}$$

Here, r represents the radius of a particle.

In our particle model, we consider three attributes of particles: shape, size, and density. Particle shape is assumed to be spherical, because its projection is defined as a view-independent shape, namely, a circle. The size of the sphere is discretized using an integer number denoted by sub-pixel level, *level*, so that sub-pixel processing can be facilitated. We define the radius by dividing the pixel side length (1) by twice the level.

$$r = \frac{1}{2 \cdot level}. \tag{5}$$

Particle density can be estimated from the radius, which is an opacity value in the user-specified transfer function, and the ray segment length can be used for ray casting. From Equation 4, we have:

$$\rho = \frac{-\log(1-\alpha)}{\pi r^2 \Delta t}. \tag{6}$$

To generate an image equivalent to the volume ray casting result using PBVR, we use the above relation to estimate the particle density function. From Equation 6, we see that the number of generated particles quadruples for each doubling of the sub-pixel level.

By assuming a hardcore point process in the particle distribution, the density function ρ should have a maximum $\rho^{\max}$. Since the volume of the enclosing cube that encloses the particle is $8r^3$,

$$\rho^{\max} = \frac{1}{8r^3}. \tag{7}$$

Thus, the opacity value α has a maximum $\alpha^{\max}$. If the opacity value α is between $\alpha^{\max}$ and 1.0, the relevant density function ρ becomes a constant value, $\rho^{\max}$. Here, $\alpha^{\max} = 1 - \exp(-\pi r^2 \rho^{\max} \Delta t)$. We can calculate the particle density distribution from the opacity distribution that is derived from the user-specified transfer function. Thus, the numbers of particles, N, in the entire volume and in a volume cell are calculated as

$$N = \int_{Cell} \rho dV. \tag{8}$$

When the number of particles in Equation 8 is not an integer, the final number of particles n is determined as follows:

$$n = \begin{cases} \lfloor N \rfloor + 1 & \text{if } R \le N - \lfloor N \rfloor \\ \lfloor N \rfloor & \text{otherwise} \end{cases}, \tag{9}$$

where R is a uniform random number in [0, 1].

Particles are generated at each tetrahedral cell. In the cell, the locations of the particles are calculated stochastically in the local coordinate system, which may be one of a variety of types (e.g., barycentric).

3.2. Evaluation of Fluctuation

Our technique may suffer from fluctuations due to randomness in particle location. To evaluate the fluctuations with respect to the sub-pixel level, we calculated the brightness at a single pixel by assuming that the particle density is constant and that the particle luminosity is constant. As explained

above, opacity describes the probability that more than one particle exists in a certain ray segment. The average brightness value of a sub-pixel is calculated as:

$$B_{Ave.} = 1 \cdot \alpha + 0 \cdot (1-\alpha) = \alpha \tag{12}$$

The variance is calculated as:

$$B_{Var.} = (1-\alpha)^2 \alpha + (0-\alpha)^2 (1-\alpha) = (1-\alpha)\alpha \tag{13}$$

Thus, the deviation becomes the square root of the variance:

$$B_{Dev.} = \sqrt{(1-\alpha)\alpha} \tag{14}$$

The total brightness is calculated by averaging all sub-pixel values in a single pixel. Thus, we theoretically analyze the fluctuation of the total brightness.

First, we define a brightness value in the i-th sub-pixel as B^i. If the occurrence of every particle is independent from each other at each sub-pixel, the average and variance of the brightness value of a sub-pixel are identical:

$$\begin{aligned} B_{Ave.} &= B^i{}_{Ave.} &&= E(B^i) &&= \alpha \\ B_{Var.} &= B^i{}_{Var.} &&= Var(B^i) &&= (1-\alpha)\alpha \end{aligned} \tag{15}$$

The total brightness value is calculated by averaging brightness values across all the sub-pixels.

$$B^{total}(level) = \sum_{i=1}^{level^2} \frac{B^i}{level^2} \tag{16}$$

The variance in the total brightness can be calculated as follows:

$$\begin{aligned} B^{total}_{Var.}(level) &= Var\left(\sum_{i=1}^{level^2} \frac{B^i}{level^2} \right) \\ &= \frac{1}{level^4} Var\left(\sum_{i=1}^{level^2} B^1 \right) \\ &= \frac{1}{level^4} \left\{ \sum_{i=1}^{level^2} Var(B^i) + 2 \sum_{i,j:i<j} Cov(B^i, B^j) \right\} \end{aligned} \tag{17}$$

Since the brightness values of sub-pixels are independent from one another, the covariance term $Cov(B^i,B^j)$ can be estimated as zero. This makes the variance decrease as the number of sub-pixels, $level^2$, increases. Therefore, the deviation in total brightness can be expressed as:

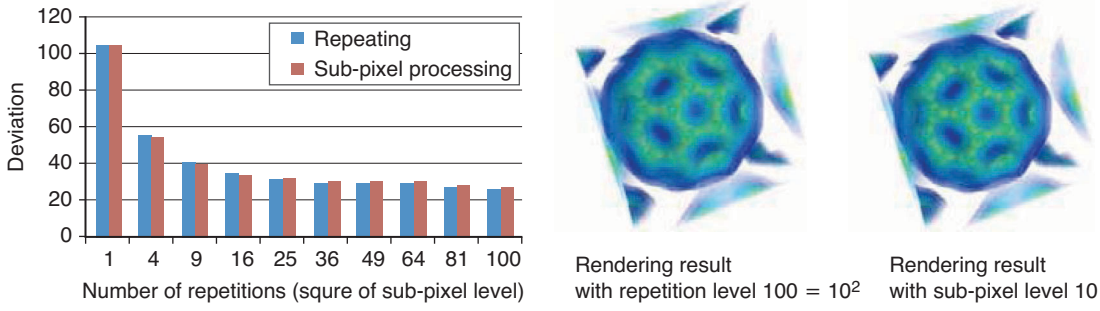

Figure 3. Deviation distribution and rendering images of the test data (1,250,235 tetrahedral cells and 262,144 nodes). The same effect can be obtained from rendering images with the sub-pixel level as well as with the repetition level as the square of the sub-pixel level.

$$B_{Dev.}^{total}(level) = \frac{B_{Dev.}}{level} = \frac{\sqrt{(1-\alpha)}\alpha}{level} \tag{18}$$

If we assume no variation in particle density between sub-pixels within a single pixel, we can achieve the same result by repeating the projection of the pixel-sized particle $level \times level$ times. This is equivalent to ensemble averaging, which is often used in CFD turbulence calculations. Figure 3 shows the deviation in total brightness at both the sub-pixel and the repetition levels. The repetition level is equivalent to the square of the sub-pixel level. Two deviation distributions are therefore matched.

4. DISTRIBUTED PBVR

In this section, we describe a distributed implementation of PBVR that is useful for handling large-scale irregular volume datasets efficiently. To allow for efficient handling, we developed a cell-by-cell particle generation technique by estimating a particle density field from a given transfer function. Each cell may be processed to generate particles that can be projected in an arbitrary order. This makes the algorithm run efficiently in a distributed computing environment.

4.1. Datasets

In this paper, we apply the proposed technique to CFD simulation results from the oral airflow field. In the simulation model, the oral cavity shape of the dental fricative was obtained by a Cone Beam CT (CBCT) scanner that can take 512 slices of 512×512 pixels in 18 seconds. Using data derived from image slices, the oral cavity can be extracted using two threshold CT values, and the required hexahedral cells can be constructed for large-eddy CFD simulation. The resulting irregular volume dataset is composed of 71 million hexahedral cells, and these are divided into the 16 datasets resulting from the distributed CFD computation. To date, only surface-based visualization has been conducted, since the currently-available volume rendering software cannot address multiple, large hexahedral volume datasets. We applied our distributed particle volume rendering method to render these 16 datasets using a streaming-based technique.

4.2. System Configuration

We implemented the distributed PBVR using a PC cluster system (see Figure 4). Our system consists of multiple processing PCs and a single-viewer PC. These PCs are networked using Gigabit Ethernet. Our proposed system is comprised of four stages: file loading, particle generation, sub-pixel transmission and image composition.

A. File loading

In our system, each irregular volume is composed of a cell connection, a node coordinate, and its associated scalar information and irregular volumes. All results can be processed as a single large data file or as multiple data files. In the former case, we constructed a mapping table from a cell identifier to a PC identifier in advance. Thus, each volume cell is assigned to one PC for processing. In the latter

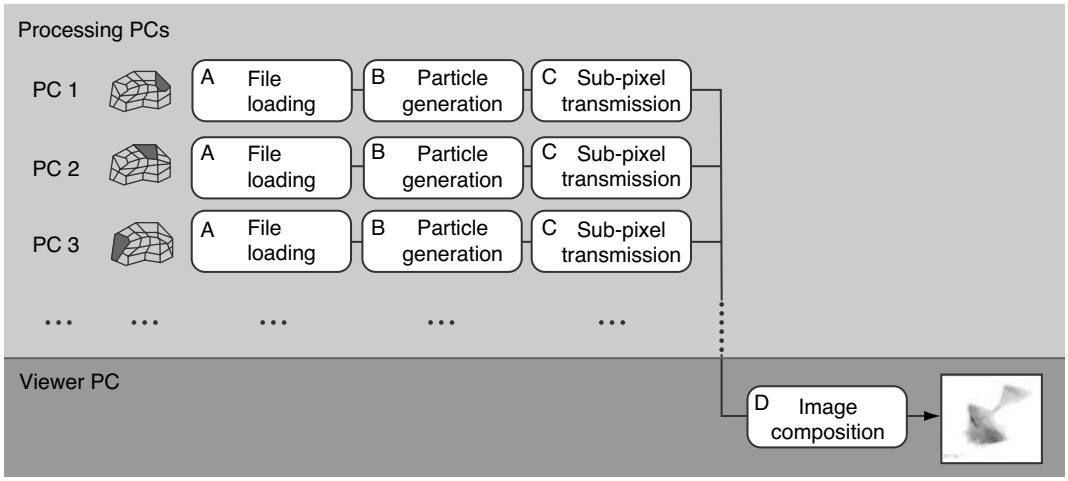

Figure 4. Distributed PBVR system.

case, we constructed a table that links a data file with a PC identifier. The mapping table is designed so that each processing PC can load the partial irregular volume in its main memory.

B. Particle generation

Particle density is estimated at each cell of the loaded partial irregular volume based on a given transfer function in each processing PC. Particles are then generated according to the estimated density field.

C. Sub-pixel transmission

The generated particles are projected onto the frame buffer with a sub-pixel resolution in each processing PC, and the stored particles in each frame buffer (active sub-pixels) are transmitted to a global frame buffer in the viewer PC. For transmission, the sub-pixel is represented in terms of its color, depth and identifier. A unique identifier is assigned to each sub-pixel location in the frame buffer. When multiple identifiers appear consecutively in a horizontal direction, we employ run-length encoding for increased efficiency. In run-length encoding, consecutive sub-pixel locations are represented using the left-most identifier, while the number of sub-pixels is expressed by run-length.

The sub-pixels are transmitted as long as they are calculated. The transmission can be asynchronous, since no visibility sorting is required for PBVR.

D. Image composition

The viewer PC receives the projected particles that are transmitted by the processing PCs. In the viewer PC, a frame buffer is allocated at the appropriate sub-pixel level, and each sub-pixel is updated using transmitted particles. The updating calculation is performed based on the Z-buffering algorithm. Since each particle has been pre-filtered at the particle generation stage, the amount of calculations for updates should be reduced.

5. EXPERIMENTAL RESULTS

In order to verify the effectiveness of our proposed technique, we used it to visualize the CFD simulation results of an oral airflow field generated by a finite element method solver, FrontFlow/Blue (Advance Soft Corporation). In this experiment, we used a viewer PC and eight processing PCs. The viewer PC featured an Intel Core 2 Duo E6750 (2.66 GHz) CPU and 2.0 GB RAM. Two of the processing PCs had Intel Core 2 Duo E4500 (2.2 GHz) CPU, and the remaining six featured Intel Core 2 Duo E6300 (1.86 GHz) CPUs and 2.0 GB RAM. Thus, the processing PCs had a total of 16 CPU cores.

5.1. Visualization Results

The CFD simulation results for exploring dental fricative sound can be rendered using our distributed PBVR. During rendering, a transfer function was designed to reduce the number of particles and enhance important regions. First, we set a simple, default linear relation from pressure and velocity to opacity. For a repetition level of 100, the pressure volume generates over 1G particles, and the velocity volume generates 36M particles. In the resulting image, the important region was occluded, and it was difficult to realize an interactive visualization. Next, we improved the transfer function to reduce the number of particles in order to realize the interactive frame rate. Using the improved transfer function, the generated pressure and velocity volumes are 15M and 17M, respectively. In Figure 5, we can easily see that there are multiple areas with high-pressure values that relate to sound sources. We can also confirm a more precise distribution of the velocity magnitude of the flow field in the oral cavity from Figure 6.

5.2. Performance Evaluation

Figure 7 (a) describes the emergence of a parallelization effect, because total processing time decreases as the number of CPU cores among the processing PCs increases. The processing time becomes 23.09 seconds when 16 CPU cores are used. Figure 7 (b) shows that the processing time increases at the image composition stage, although it decreases at the file loading and the particle projection stages, at which points the number of CPU cores increases. The processing time decreases at the sub-pixel transmission stage, as is consistent with an increase in the number of CPU cores. Since total processing time is correlated with processing time at each CPU core, the total processing time also becomes larger. To forecast performance when the number of CPU cores exceeds 16, we construct a performance model based on our experimental results:

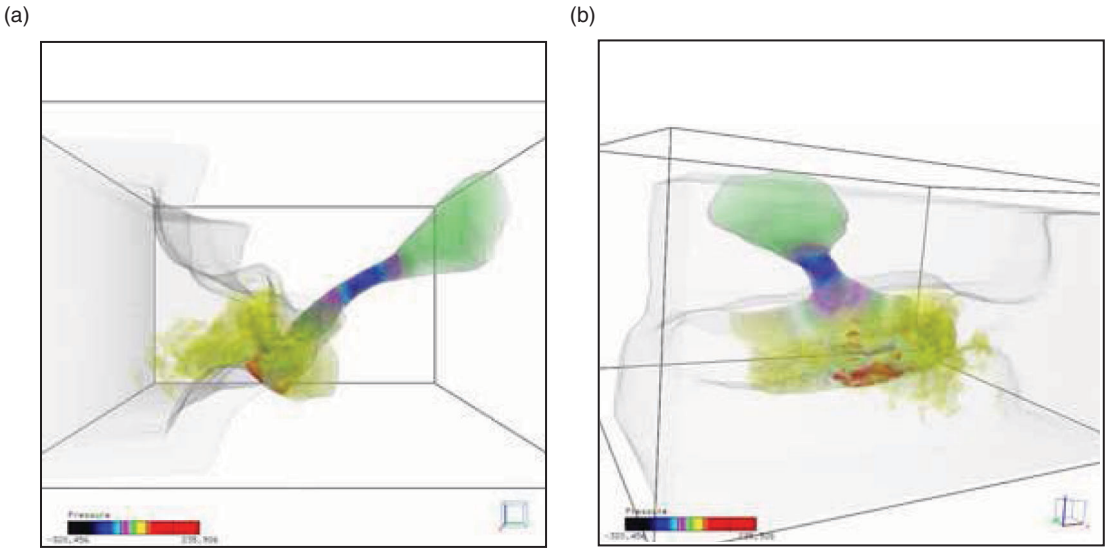

Figure 5. Visualization results for the pressure field using the proposed technique.

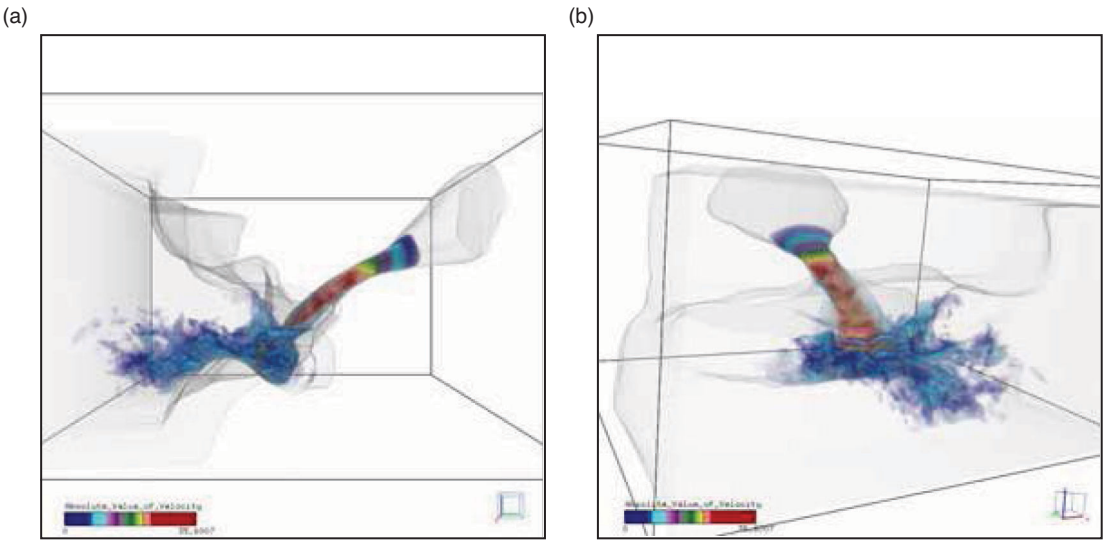

Figure 6. Visualization results for the velocity magnitude using the proposed.

$$T(n) = T_A(n) + T_B(n) + T_C(n) + T_D(n) \tag{19}$$

In Equation 19, n represents the number of CPU cores among the processing PCs.

A. File loading

Since each CPU core among the processing PCs uniformly loads partial data from the CFD simulation results (i.e., 16 datasets), the file loading time decreases as the number of CPU cores increases. Therefore, the loading time, $T_A(n)$ [sec], can be expressed as follows:

$$T_A(n) \cong \frac{T_A(1)}{n} \tag{20}$$

B. Particle generation

In the particle generation stage, processing time is inversely proportional to the number of CPU cores, since the number of volume cells that are processed in each processing PC decreases. Therefore, the projection time, $T_B(n)$ [sec], can be expressed as follows:

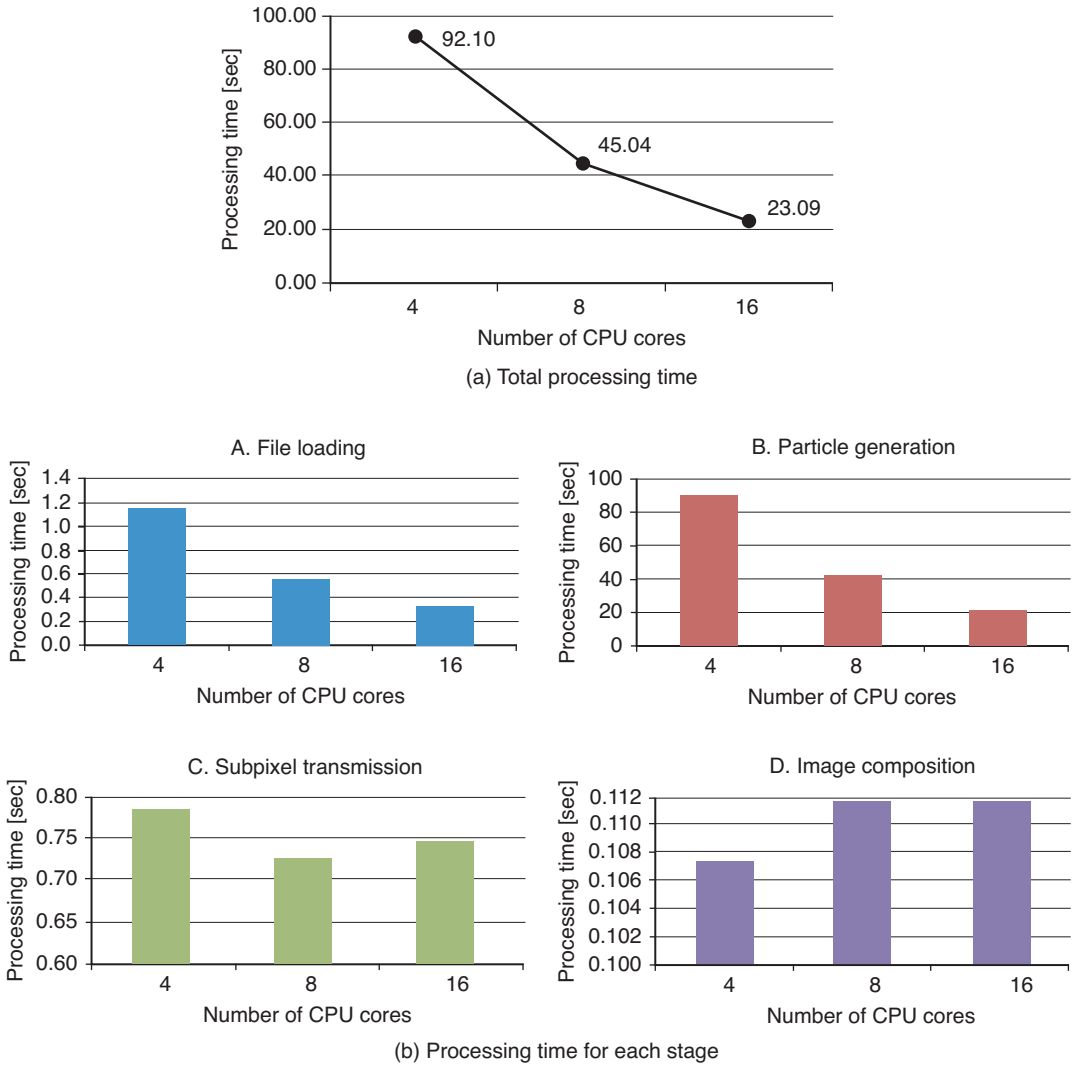

Figure 7. Relationship between the number of CPU cores and processing time.

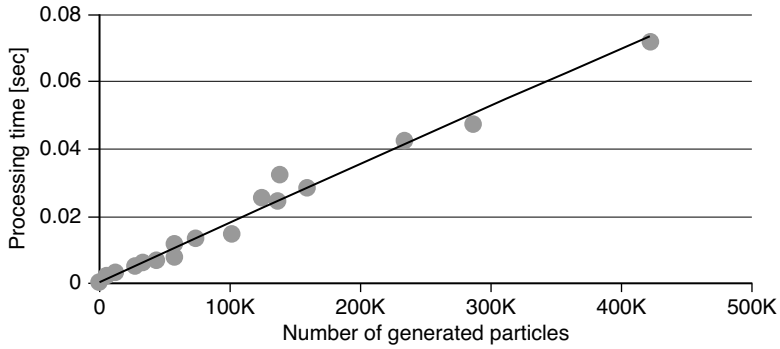

Figure 8. Projection time.

C. Sub-pixel transmission

This stage includes particle projection, run-length encoding and transmission. By assuming that the processing times for these procedures are T_{proj} [sec], T_{enc} [sec] and T_{trans} [sec], respectively, the total processing time in this stage, T_C [sec], is as follows:

$$T_C(n) = T_{proj}(n) + T_{enc}(n) + T_{trans}(n) \tag{22}$$

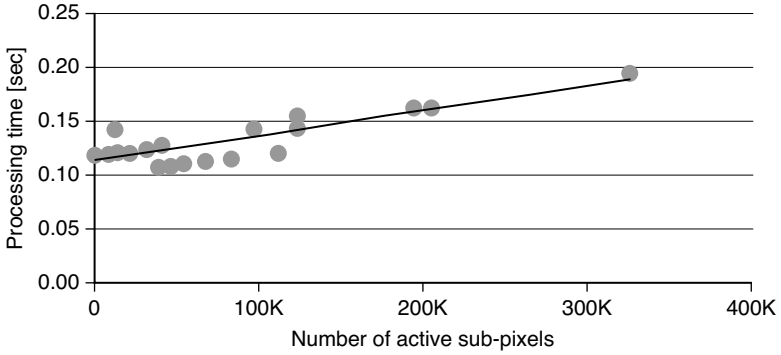

Figure 9. Encoding time.

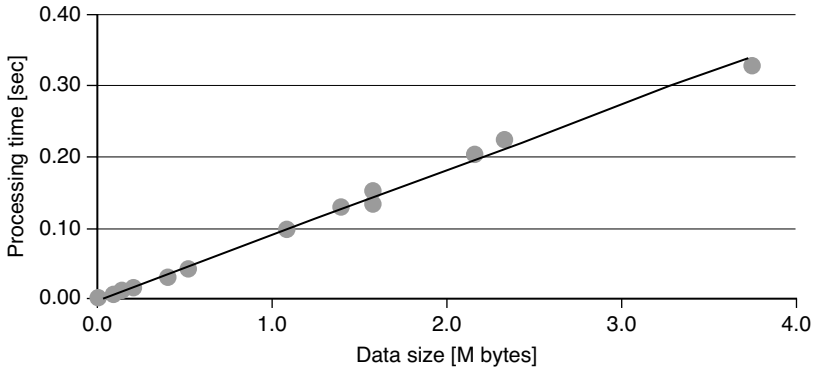

Figure 10. Transmission time.

We have measured the particle projection time for irregular volume in our experiment by utilizing the number of generated particles. Figure 8 shows the obtained result. From this figure, we confirm that the processing time increases proportionately as the number of generated particles increases. Since the number of generated particles on a processing PC also decreases as the number of CPU cores increases, the particle projection time, T_{proj}, can be expressed as follows:

$$T_{proj}(n) = \frac{P}{n} \cdot t_{proj}, \tag{23}$$

where P is the overall amount of generated particles among the processing PCs, and t_{proj} is the projection time for a particle.

Encoding time depends on the number of particles that are projected onto the sub-pixels (i.e., the number of active sub-pixels). Figure 9 shows the relationship between encoding time and the number of active sub-pixels in our experiment. Encoding time can be considered proportional to the number of active sub-pixels in the graphs depicted in this figure. If we assume that the time necessary to check whether a sub-pixel is active is T_{check}, that the number of active sub-pixels is $a(n)$, and that encoding time for an active sub-pixel is t_{enc}, then the run-length encoding time T_{enc} can be expressed as follows:

$$T_{enc}(n) = a(n) \cdot t_{enc} + T_{check} \tag{24}$$

In order to evaluate transmission time for the encoded active sub-pixels, we measured the required transmission time and the transmitted data size; the obtained results are shown in Figure 10. From this figure, we confirm that transmission time increases proportionately as the transmitted data size increases. Letting $D(n)$ bytes be the total amount of the transmitted data and t_{trans} be transmission time for one byte of data, we can represent transmission time, T_{trans}, as follows:

$$T_{trans}(n) = D(n) \cdot t_{trans} \tag{25}$$

Based on Equations 23, 24 and 25, processing time for this stage, T_C, is as follows:

$$T_C(n) = \frac{P}{n} \cdot t_{proj} - a(n) \cdot t_{enc} + D(n) \cdot t_{trans} + \boldsymbol{T}_{check} \tag{26}$$

D. Image composition

At this stage, the viewer PC receives data transmitted from the processing PCs and creates a copy in its main memory. These copied fragments are then used to update the frame buffer. In our implementation, the process is serialized at each CPU core. We measured the updating time during this stage; the results are shown in Figure 11. From this figure, the total image compositing time, T_D (n) [sec], can be expressed as follows by letting t_{update} be updating time for one byte of data and T_{disp} be the processing time for displaying the updated frame buffer:

$$T_D(n) = D(n) \cdot t_{update} + T_{disp} \tag{27}$$

Therefore, the performance model is:

$$T(n) = \frac{T_A(1) + T_B(1)}{n} - \frac{P}{n} \cdot t_{proj} + a(n) \cdot t_{enc} + D(\boldsymbol{n}) \cdot (t_{trans} + t_{update}) + T_{check} + T_{disp} \tag{28}$$

We used the performance model presented in Equation 28 in order to verify the parallelization effect of our proposed system. Based on experimental results with four CPU cores, we obtain $T_A(1) = 4T_A(4) = 9.80$ [sec] and $T_b(1) = 4T_B(4) = 448.19$ [sec]. We also measured the active sub-pixels and the transmitted data size when utilizing different number of CPU cores in order to estimate $a(n)$ and $D(n)$; the results are shown in Tables 1 and 2. From these results, the simple regression formulas for $a(n)$ and $D(n)$ are as follows:

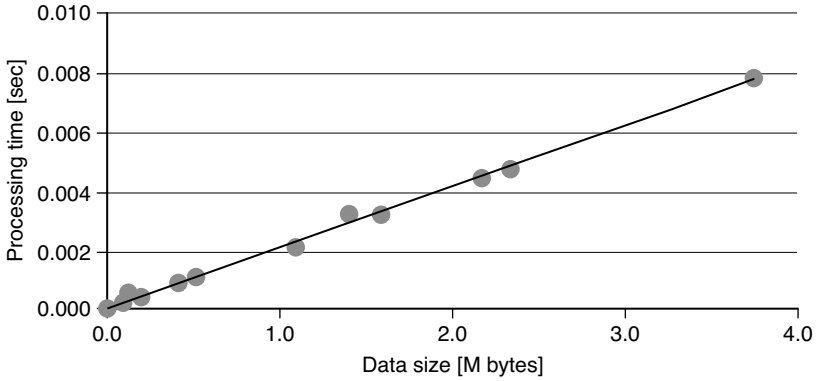

Figure 11. Updating time.

Table 1: Number of active sub-pixels

Num. of CPU cores	4	8	16
$a(n)$	1.44×10^5	0.73×10^5	0.39×10^5

Table 2: Total amount of transmitted data [bytes]

Num. of CPU cores	4	8	16
$D(n)$	7.05×10^6	7.28×10^5	8.06×10^6

$$a(n) = 3.43 \times 10^3 + \frac{5.64 \times 10^5}{n} \tag{29}$$

$$D(n) = 6.66 \times 10^6 + 8.59 \times 10^4 n \tag{30}$$

As per Figures 8, 9, 10 and 11, we also use linear regression to find that $t_{proj} = 1.62 \times 10^{-7}$, $t_{enc} = 2.37 \times 10^{-7}$, $t_{trans} = 8.36 \times 10^{-8}$ and $t_{update} = 1.98 \times 10^{-9}$.

Therefore, we estimate the performance model as follows:

$$T(n) = \frac{458.24}{n} + 7.35 \times 10^{-3} n + C, \tag{31}$$

where $C = T_{check} + T_{disp} + 0.57$. The number of particles P described in Equation 28 was 7.36×10^5 in our experiment.

If the parallelization effect holds in our system, the derivative with respect to the number of CPU cores, n, should be negative:

$$\frac{\partial T(n)}{\partial n} = -\frac{458.24}{n^2} + 7.35 \times 10^{-3} < 0 \tag{32}$$

The maximum n that satisfies the Equation 32 equals 249. This means that total performance time decreases when the number of CPU cores exceeds 250, since the increase in processing time at the sub-pixel transmission and image composition stages contributes more to the total performance than the decrease impacts the file loading and particle projection stages. In the future, we hope to parallelize the data transmission and image composition processes in the viewer PC.

6. CONCLUSIONS

We have developed a distributed implementation of PBVR for visualizing a large-scale irregular volume dataset generated from a distributed CFD simulation. We first applied the developed technique to the CFD simulation results, where such a large-scale dataset has, until now, never been visualized by a volume rendering technique. Using the experimental results, we constructed a performance model as a function of the number of CPU cores and found that the overall performance time improves as the number of CPU cores increases up to 250. Using our proposed technique, we have also verified that the pressure field has local maxima close to the frontal teeth in the oral cavity.

In future studies, we plan to develop a high-resolution rendering system using a tiled display wall (TDW) and the distributed PBVR to visualize a large-scale volume dataset. In addition, we intend to apply the GPU technique for accelerating the volume rendering process.

REFERENCES

[1] R. D. Kent, C. Read, The acoustic analysis of speech, 2nd edition, Sibgular Thomson Learning, Canada, pp. 38–43, 2002.

[2] C. H. Shadle, Articulatory-acoustic relationships in fricative consonants, in W. J. Hardcastle, A. Marchal (eds.), Speech Production and Speech Modeling, Kluwer Academic Publishers, pp. 187 209, 1990.

[3] S. Y. A. Lee, L. T. Whitebill, V. Ciocca, N. Samman, Acoustic and Perceptual Analysis of the Sibilant Sound /s/ Before and After Orthognathic Surgery, J. Oral Maxillofac Surg., 60, pp. 364–373, 2002.

[4] S. Hamlet, V. D. Geoffrey, D. M. Bartlett, Effect of a dental prosthesis on speaker – Specific characteristics of voice, J. Speech Hear. Res., Vol. 19, pp. 639–650, 1976.

[5] E. Pound, Controlling anomalies of vertical dimension and speech, J Prosthet Dent. 36(2), pp.124–135, 1976.

[6] K. Nozaki, H. Tamagawa, S. Shimojo, Prediction of dental fricative sound by lighthill-curle equation, In silico Dentistry, Japan, pp, 155–158, 2008.

[7] K. N. Stevens, Acoustic Phonetics, MIT Press, Cambridge Massachusetts, pp. 379–412, 1988.

[8] P. Sabella, A Rendering Algorithm for Visualizing 3D Scalar Field, Computer Graphics, Vol. 22, No. 4, pp. 51–58, 1988.

[9] P. Shirley, and A. Tuchman, A Polygonal Approximation to Direct Scalar Volume Rendering, In Proc. of San Diego Workshop on Volume Visualization, pp.63–70, 1990.

[10] P. Williams, Visibility-ordering of Meshed Polyhedra, ACM Transaction on Graphics, Vol. 11, No. 2, pp. 103–126, 1992.

[11] S. Callahan, M. Ikits, J. Comba, and C. Silva, Hardware-Assisted Visibility Ordering for Unstructured Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, Vol.11, No.3, pp.285–295, 2005.

[12] E. W. Anderson, S. P. Callahan, C. E. Scheidegger, J. Schreiner, and C. T. Silva, Hardware-Assisted Point-Based Volume Rendering of Tetrahedral Meshes, Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), pp.163–172, 2007.

[13] S. Roettger, and T. Ertl, Cell Projection of Convex Polyhedra, In Proc. of Volume Graphics 2003, pp. 103–107, 2003.

[14] B. Csébfalvi, and L. Szirmay-Kalos, Monte Carlo Volume Rendering, In Proc. of IEEE Visualization 2003, pp. 449–456, 2003.

[15] B. Csébfalvi, Interactive Transfer Function Control for Monte Carlo Volume Rendering, In Proc. of IEEE Symposium on Volume Visualization and Graphics 2004, pp. 33–38, 2004.

[16] Y. Zhou and M. Garland, Interactive Point-Based Rendering of Higher-Order Tetrahedral Data, IEEE Transaction of Visualization and Computer Graphics, Vol.12, No.5, pp.1229–1236, 2006.

[17] N. Sakamoto, J. Nonaka, K. Koyamada, and S. Tanaka, Particle-based Volume Rendering, In Proc. of Asia-Pacific Symposium on Visualization 2007, pp.141–144, 2007.

[18] S. Tanaka, T. Hatta, F. R. Ngana, A. Saitoh, N. Sakamoto, J. Nonaka, K. Koyamada, Grid-independent Metropolis sampling for volume visualization, The 6th EUROSIM Congress on Modeling and Simulation (EUROSIM 2007), CD-ROM (310), 2007.