Kawamura T, Sakamoto N, Koyamada K. Level-of-detail rendering of a large-scale irregular volume dataset using particles. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 25(5): 905–915 Sept. 2010. DOI 10.1007/s11390-010-1071-x

Level-of-Detail Rendering of Large-Scale Irregular Volume Datasets Using Particles

Takuma Kawamura[1], Naohisa Sakamoto[2], and Koji Koyamada[2], *Member, IEEE*

[1] *Graduate School of Engineering, Kyoto University, Kyoto 6068501, Japan*

[2] *Center for the Promotion of Excellence in Higher Education, Kyoto University, Kyoto 6068501, Japan*

E-mail: {kawamura, naohisa}@viz.media.kyoto-u.ac.jp; koayamada@mbox.kudpc.kyoto-u.ac.jp

Received July 24, 2009; revised February 27, 2010.

Abstract This paper describes a level-of-detail rendering technique for large-scale irregular volume datasets. It is well known that the memory bandwidth consumed by visibility sorting becomes the limiting factor when carrying out volume rendering of such datasets. To develop a sorting-free volume rendering technique, we previously proposed a particle-based technique that generates opaque and emissive particles using a density function constant within an irregular volume cell and projects the particles onto an image plane with sub-pixels. When the density function changes significantly in an irregular volume cell, the cell boundary may become prominent, which can cause blocky noise. When the number of the sub-pixels increases, the required frame buffer tends to be large. To solve this problem, this work proposes a new particle-based volume rendering which generates particles using metropolis sampling and renders the particles using the ensemble average. To confirm the effectiveness of this method, we applied our proposed technique to several irregular volume datasets, with the result that the ensemble average outperforms the sub-pixel average in computational complexity and memory usage. In addition, the ensemble average technique allowed us to implement a level of detail in the interactive rendering of a 71-million-cell hexahedral volume dataset and a 26-million-cell quadratic tetrahedral volume dataset.

Keywords large-scale irregular volume, level-of-detail, volume rendering of unstructured meshes

1 Introduction

The development of techniques for irregular volume datasets has remained a challenge for the scientific visualization community. Such datasets consist mainly of scalar data defined on collections of irregularly ordered cells whose shapes are not necessarily orthogonal cubic. Data of this type are often generated by finite element method (FEM), a technique that is widely used in computational mechanics and is also becoming popular in computational fluid dynamics (CFD) and computational structural mechanics (CSM). Irregular volume rendering techniques can be classified into two approaches: the image-order approach and the object-order approach. Many techniques for irregular volume datasets have been developed based on both approaches. Under each approach, a tetrahedral cell is often selected as a visualization primitive, for reasons of stability and efficiency of computation.

Although substantial improvement has been made in irregular volume rendering, visibility ordering remains one of the most important techniques. As long as the volume rendering algorithm follows the density emitter model proposed by Sabella[1], visibility ordering should be considered. Sakamoto[2] focused on the particle model from which the density emitter model was derived. Koyamada[3] then developed particle-based volume rendering (PBVR), an efficient approach for rendering irregular volumes without the need for a visibility ordering process. Although the density emitter model is continuous in nature, the particle model is discrete. An obvious advantage of PBVR as compared with volume ray-casting is that the former explores important features that the latter may miss. PBVR employs importance sampling while the ray-casting approach often relies on simple sampling. Importance sampling is possible in ray-casting, but the sampling must be done at each viewing point.

A key to our success in volume rendering of large irregular volume datasets is scalability in the number of volume cells. The original PBVR procedure processed a given irregular volume dataset in a cell-by-cell manner using a particle density estimation technique. Although the original technique improved the image quality by employing a smaller particle size and increasing the number of particles, the size of the required frame

Regular Paper

buffer tended to be large since the original PBVR procedure used sub-pixel processing. To solve this problem, the current work employs the concept of ensemble average, repeating the particle projection to accumulate actual pixel values by setting the sub-pixel length to that of a single pixel.

Under the previous PBVR procedure, the density function is constant within an irregular volume cell, and the density value is represented by the centroid. When the density function changes significantly in an irregular volume cell or when the aspect ratio is high, we can see the cell boundary. This is the case when an irregular volume cell becomes large with respect to the screen size, which is frequently the case in immersive display environments. To solve this problem, we propose a particle generation technique for irregular volume datasets using metropolis sampling.

This paper is organized as follows. Section 2 describes related techniques for rendering irregular volumes. After reviewing previous work on particle density estimation equations in Section 3, Section 4 develops the new particle generation technique. Section 5 describes a new particle rendering technique that uses an ensemble set of particles. To show the effectiveness of this new rendering method, we apply it to the rendering of large-scale irregular volume datasets composed of 71 million hexahedral cells (see Fig.1) and 26 million quadratic tetrahedral cells, respectively. Section 6 gives the experimental results. Section 7 gives the discussion. Section 8 concludes the paper.

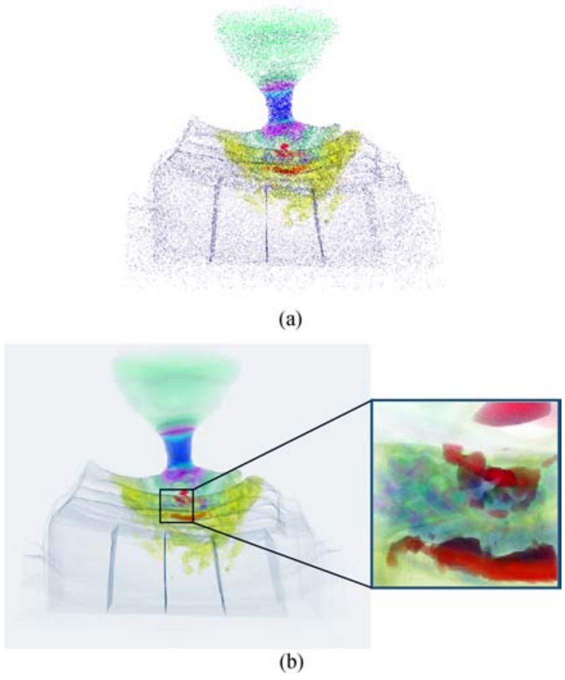

Fig.1. LOD Rendering of large-scale irregular dataset which is the result of an oral cavity airflow simulation. (a) Coarse rendering for smooth movements. (b) Fine rendering to stop movements and its close-up.

2 Related Works

Since Shirley *et al.*[4] proposed a projected tetrahedra (PT) algorithm for generating a volume-rendered image using tetrahedral cells, many irregular volume rendering techniques have been proposed based on the object-order approach. In Shirley *et al.*'s algorithm, each tetrahedral cell is projected onto the screen in a visibility order, from back to front, to build up a semi-transparent image. The work of Williams[5] led to a considerable amount of research on visibility ordering techniques. Meredith and Ma[6] developed a technique that approximates irregular volumes as an octree structure and renders them using hardware-assisted splats. They found that when the octree nodes are ordered back to front, the difference between sorting and not sorting within octree nodes is nearly impossible to see in the rendered image. One of the major current directions in irregular volume rendering is to relax the processing requirement for the visibility sorting or to develop a technique without sorting.

Proceeding in the former direction, Callahan *et al.*[7] developed an integrated visibility ordering technique called HAVS, in which the centroids of the cell faces are first sorted in order to make a rough visibility ordering, and pixel fragments generated from rasterized faces are used along with the k-buffer in order to increase the accuracy. Although HAVS achieves 1.3 fps for 1.4 million tetrahedra, it generates some artifacts when the k-buffer is not large enough. It is also difficult to determine the optimum value for k and the memory required to store the cell faces is about twice that for the tetrahedral cells.

Anderson *et al.*[8] proposed a point-based technique to render irregular volumes. They represent a tetrahedral cell as a point primitive at the center, followed by rendering and compositing of the represented point primitives. Like HAVS, this technique requires 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.*[9] pointed out that the memory bandwidth required for visibility sorting becomes the speed-limiting factor, and they proposed an algorithm that requires no visibility sorting for cells of irregular volume. However, since their optical model only considers emissions, its application is limited to the visualization of gaseous phenomena. Csebfalvi *et al.* proposed a sorting-free volume rendering technique[10-11] that can be categorized as an X-ray volume rendering approach. Their optical model considers only absorption. Zhou *et al.* proposed a sorting-free rendering technique that is implemented with additional terms to help provide enhanced depth cues without visibility sorting[12]. Although this method has achieved 20 fps for 17.6 million tetrahedra, their optical model did not consider the absorption effect.

Sakamoto *et al.* proposed a basic PBVR technique and applied it to several volume datasets[2]. In this technique, the location of particles was limited to the regular grid, so it was difficult to generate images of equivalent quality to ray-casting. To solve this problem, Koyamada *et al.* introduced a new particle model that enables the particle density to be determined uniquely from a transfer function[3]. This introduction can make the resulting image comparable to that of ray-casting. Ding *et al.* designed a comparative experiment in which PBVR and HAVS visualized the same irregular volume datasets. In this experiment, PBVR outperformed HAVS in both performance and image quality[13].

3 Particle-Based Volume Rendering

In the volume rendering, the brightness can be calculated as

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

Here, a viewing ray is evenly subdivided into n segments, and c_i and a_i represent luminosity and opacity values at the i-th sampling point, which is a central point of the interval $[t_{i-1}, t_i]$. Usually, the opacity is specified by a visualization user through a transfer function. In the density emitter model that Sabella first proposed for volume rendering[1], the opacity a_k in the k-th ray segment is defined as:

$$\alpha_k = 1 - \exp\Big(-\int_{t_k}^{t_{k-1}} \pi r^2 \rho(\lambda)\mathrm{d}\lambda\Big). \tag{2}$$

Here, ρ and r represent the number of particles in the unit volume and the radius of a particle, respectively. Since the Poisson distribution is assumed for the number of particles in the density emitter model, the opacity describes the possibility that more than one opaque particle exists along the ray segment. If we assume that the density function is constant in the segment, and that the ray segment length can be described as $\Delta t = t_{k-1} - t_k$, we have:

$$\alpha_k = 1 - \exp(-\pi r^2 \rho_k \Delta t). \tag{3}$$

From (3), the opacity can be generally expressed as:

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

Our particle model considers three particle attributes: shape, size, and density. The particle shape is assumed to be a sphere because its projection is a circle. The size of the sphere is characterized by its diameter, which we assume equals the pixel side length divided by an integer, to facilitate the particle rendering. We call the number the sub-pixel level (L_s):

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

The particle density can be estimated using the radius, an opacity value in the user-specified transfer function, and the ray-segment length used in the ray-casting. From (4), we have:

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

A particle-based volume rendering technique is composed of two processes: particle generation and particle rendering. The first process constructs a density field and generates particles consistent with the density function. The second process projects particles onto an image plane and calculates a final brightness value using illuminated particles[2].

4 Particle Generation Using Metropolis Sampling

To generate a rendering image of equivalent quality to the volume ray-casting result, the above relation must be used to estimate the particle density function. From (6), we understand that the number of generated particles quadruples for each doubling of the sub-pixel level. 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,

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

$$\alpha^{\max} = 1 - \exp(-\pi r^2 \rho^{\max} \Delta t). \tag{8}$$

The $\rho^{\max}$ is calculated by the simple cubic lattice because the particle positions are discretized to the sub-pixel position during projection.

The particle density distribution can be calculated from the opacity distribution, which can be derived by the user-specified transfer function.

$$\rho = \begin{cases} \dfrac{-\log(1-\alpha)}{\pi r^2 \Delta t}, & (0 \leqslant \alpha \leqslant \alpha^{\max}),\\ \rho^{\max}, & (\alpha^{\max} \leqslant \alpha \leqslant 1). \end{cases} \tag{9}$$

Thus, the number of particles, N, in the entire volume or in a specific volume cell is calculated as

$$N = \int_{\text{Cell}} \rho \mathrm{d}V. \tag{10}$$

If we employ a linear interpolation function within a tetrahedral cell, the integration is equal to the volume multiplied by the density at the centroid. When the number of particles in (10) is not an integer, the final number n is determined as follows:

$$n = \begin{cases} \lfloor N \rfloor + 1, & \text{if } R \leqslant N - \lfloor N \rfloor, \\ \lfloor N \rfloor, & \text{otherwise}, \end{cases} \tag{11}$$

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

Particles are generated within each irregular volume cell (see Fig.2). In order to generate the particle density distribution according to (11), we use the metropolis method. This method generates particles one by one in a volume space, starting at an arbitrary initial particle position x_0. In the following, x_i denotes the position of the i-th particle, i.e., the position of the i-th generated particle. The particle generation is performed as follows:

Step 1. Calculate $\rho(x_i)$ for the current particle position, x_i. Generate a new particle at a candidate (trial) position, x', that is chosen randomly in the volume space, and calculate $\rho(x')$.

Step 2. Calculate the ratio of $\rho(x')$ to $\rho(x_i)$, i.e., $W(x_i \to x') \equiv \rho(x')/\rho(x_i)$.

Step 3. If $W(x_i \to x') \geqslant 1$, accept x' as an updated particle position, x_{i+1}, and go back to Step 1. Otherwise, go on to Step 4.

Step 4. Generate a uniform random number $R \in [0, 1)$, and determine x_{i+1} as follows:

$$x_{i+1} = \begin{cases} x', & \text{if } R \leqslant W(x_i \to x'), \\ x_i, & \text{otherwise}. \end{cases} \tag{12}$$

Then go back to Step 1.

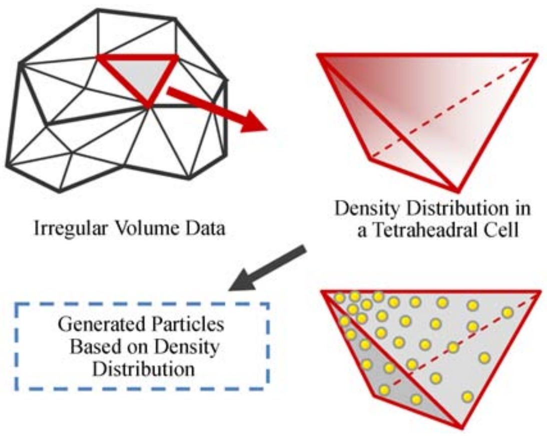
Irregular Volume Data
Density Distribution in a Tetraheadral Cell
Generated Particles Based on Density Distribution

Fig.2. Particle generation at each tetrahedral cell.

The particle generation terminates when the specified number of particles has been generated. The particles are not counted if $x_{i+1} = x_i$.

We applied interpolation functions and coordinate transformations used in the finite element method so that the particles can be generated in cells of various shapes. This method first generates a particle position in local coordinates $\boldsymbol{x}$ and then carries out the coordinate transformation to map a particle position into cell T, which is described by global coordinates $\boldsymbol{X}$.

The local coordinate of the hexahedral cell is defined as the region $[0, 1]^3$. In the case of the tetrahedral or the quadratic tetrahedral cell, those local coordinates are described by means of a barycentric coordinate with $[0, 1]^4$.

The interpolation function `Interpolation`$(\boldsymbol{x}, T)$ calculates a scalar value S using shape functions $N(\boldsymbol{x})$, which are determined by the cell shape[14]:

$$S = \sum_{i=1}^{n_p} N_i(\boldsymbol{x}) S_i. \tag{13}$$

Here, n_p is the number of nodes and S_i are the scalar values on the nodes of the cell T.

The coordinate transformations `Mapping`$(\boldsymbol{x}, T)$ from the local coordinate to the global coordinate are:

$$\boldsymbol{X} = \sum_{i=1}^{n_p} N_i(\boldsymbol{x}) \boldsymbol{X}_i^{\text{node}}. \tag{14}$$

Here, $\boldsymbol{X}_i^{\text{node}}$ are the global coordinates of cell nodes.

The function `Normal`$(\boldsymbol{x}, T)$, which calculates the gradient of $S(\boldsymbol{X})$, can be expressed as:

$$\nabla S(\boldsymbol{X}) = \boldsymbol{J}^{-1} \nabla S(\boldsymbol{x}). \tag{15}$$

Here, $\boldsymbol{J}$ is the Jacobian matrix of the coordinate transformation (14).

The scalar value S, the particle position $\boldsymbol{x}$, and the gradient vector $\nabla S(\boldsymbol{X})$ are stored in the memory space using a function `setParticle`$(S, \boldsymbol{X}, \nabla S(\boldsymbol{X}))$. The particle data (18 bytes) is composed of position (float $\times 3 = 12$ bytes), normal (byte $\times 3 = 3$ bytes) and color (byte $\times 3 = 3$ bytes) vectors. The pixel data (7 bytes) is composed of the color vector (3 bytes) and depth value (float $\times 1 = 4$ bytes) in the frame buffer. The pseudo code for `ParticleGeneration` is given as Algorithm 1.

Algorithm 1. `ParticleGeneration ()`

```
1: calculate density function $\rho$;
2: for each cell $T_i$ do
3:   Number of Particles $N_p$;
4:   $N_p$ = NumOfParticles($T_i$);
5:   Local Position $x_0$, $x'$;
6:   $x_0$ = SamplingInLocal($T_i$);
```

```
7:      Scalar Value $s_0$, $s'$;
8:      $s_0$ = interpolation($x_0$, $T_i$);
9:      Global Position X;
10:     Normal n;
11:     while j < $N_p$ do
12:       $x'$ = SamplingInLocal($T_i$);
13:       $s'$ = interpolation(x', $T_i$);
14:       Ratio of Density W = $\rho(s')/\rho(s_0)$;
15:       if W >= 1 then
16:         X = mapping($x'$, $T_i$);
17:         n = normal($x'$, $T_i$);
18:         setParticle($s'$, X, n);
19:         $x_0$ = $x'$
20:         $s_0$ = $s'$
21:         j++
22:       end if
23:       else
24:         if W >= random() then
25:           X = mapping($x'$, $T_i$);
26:           n = normal($x'$, $T_i$);
27:           setParticle($s'$, X, n);
28:           $x_0$ = $x'$
29:           $s_0$ = $s'$
30:           j++
31:         end if
32:       end else
33:     end while
34:   end for
```

In this pseudo code, NumOfParticles(T_i) is a function that calculates the number of particles for cell T_i. In order to calculate (10), the density of the center of gravity replaces the density distribution, and the number of particles is expressed as the product of the cell volume and the density of the center of gravity. The number of particles calculated as a real number is converted to an integer number, as in (11), by using the floor function floor() and the random number generator random(). The pseudo code for NumOfParticles is given as Algorithm 2.

Algorithm 2. NumOfParticles (cell T_i)

```
1:  Integer Number of Particles $N_p$;
2:  Volume v = $T_i$.getVolume();
3:  Local Position g;
4:  g = $T_i$.getCenterOfGravity();
5:  Real Number of Particles $R_p$;
6:  $R_p$ = v * $\rho$(mapping(g, $T_i$));
7:    if $R_p$ - floor($R_p$) > random() then
8:      $N_p$ = floor($R_p$) + 1
9:    end if
10:   else
11:     $N_p$ = floor($R_p$)
12:   end else
13:   return $N_p$
```

In this pseudo code, getVolume() is a function for obtaining the volume of the cell, and getCenterOfGravity() obtains the center of gravity of the cell.

This algorithm can easily be parallelized because the particles are generated independently in each cell.

5 Particle Rendering Using the Ensemble Average

Using the aforementioned particle generation technique, we can generate particles in a volume cell consistent with the density function. By projecting these particles onto the image plane, we calculate brightness values for the corresponding pixels. We also perform particle occlusion with the Z-buffer algorithm during projection. This incorporates the effects of particle collisions, which prevent some particles from reaching the image plane. This method assumes that the particles are completely opaque. Thus, neither alpha blending nor visibility ordering is required.

After the projection, color mapping and shading calculations are applied to the stored particles. The color mapping converts the scalar value to the color using the transfer function, and the shading multiplies the particle color by the attenuation, which is calculated using the gradient vector and the lighting vectors. To calculate the final color value at each pixel, one of two averaging operations is applied to multiple pixel values: sub-pixel average or ensemble average.

5.1 Sub-Pixel Average

In the sub-pixel average, a pixel at the sub-pixel level is divided into various sub-domains (sub-pixels). The final brightness value B^{total} is calculated by averaging brightness values B^i across all sub-pixels (see Fig.3). The number of sub-pixels in a single pixel is termed the sub-pixel level (L_{S})

$$B^{\text{total}}(L_{\text{S}}) = \sum_{i=1}^{L_{\text{S}}^2} \frac{B^i}{L_{\text{S}}^2}. \tag{16}$$

We experimentally confirmed that the fluctuation in the brightness values is inversely proportional to the sub-pixel level by using several irregular volume datasets[3]. Performing particle occlusion for each sub-pixel causes the pixel to store the nearest particles and affects the averaging procedure that is used to determine the pixel value. When performing sub-pixel processing, the size of the required frame buffer tends to be large, consistent with the increase in the sub-pixel level.

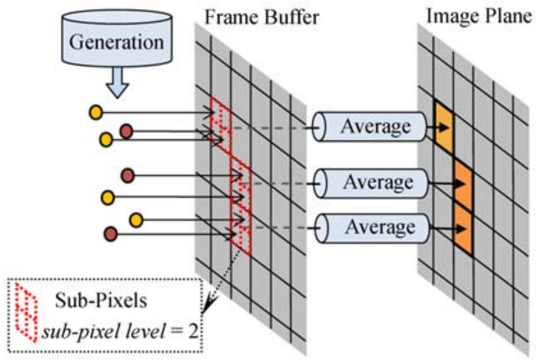

Fig.3. Sub-pixel average.

5.2 Ensemble Average

To solve the problem with respect to the size of the required frame buffer, we employ the concept of ensemble averaging and repeat the particle projection to accumulate pixel values by setting the sub-pixel length equal to that of a single pixel (see Fig.4). An ensemble is an imaginary collection of notionally identical experiments. In PBVR, the experiment involves a set of processes that include particle generation and projection. Once we set the sub-pixel length equal to 1, all sub-pixel processing is omitted. Each member of the ensemble features is nominally identical properties such as density and diameter. Members of an ensemble are, by definition, statistically independent of one another. The number of members is termed the repetition level ($L_{\rm R}$).

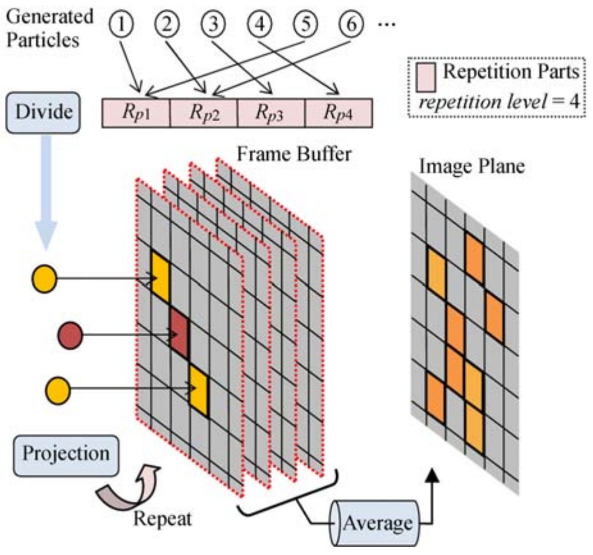

Fig.4. Ensemble average.

In the ensemble average, the final brightness value is calculated by averaging brightness values across all of the repetitions. We define a brightness value in the i-th repetition member as B^i.

$$B^{\rm total}(L_{\rm R}) = \sum_{i=1}^{L_{\rm R}} \frac{B^i}{L_{\rm R}}. \tag{17}$$

The image generation process when using the ensemble average is composed of particle division, projection and average. The particles are divided into $L_{\rm R}$ parts where each particle is cyclically located in generated order (see Fig.4). A set of divided particles is projected onto the particle buffer with the existing PBVR framework[2], and the brightness values B^i are obtained. In order to calculate the final pixel value, the brightness values are accumulated into the buffer and averaged by (17).

The ensemble average technique can be used to implement the level-of-detail (LOD) rendering. Only a portion of the particle set is used at a low repetition level to create smooth movements, and a high repetition level is used to see fine-features in detail when movement stops.

To evaluate the fluctuations in brightness with respect to the repetition level, we consider a constant volume dataset in which the particle density and particle luminosity are both constant ($c = 1$). As previously explained, the opacity describes the possibility that more than one particle exists along the ray segment. The brightness value B becomes 1 when a particle exists or 0 if no particle exists. Thus, the average brightness value, $B_{\rm Avg}$, can be calculated as:

$$B_{\rm Avg} = 1 \cdot \alpha + 0 \cdot (1-\alpha) = \alpha. \tag{18}$$

The variance of the brightness can be calculated as:

$$B_{\rm Var} = (1-\alpha)\alpha. \tag{19}$$

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

$$B_{\rm Dev} = \sqrt{(1-\alpha)\alpha}. \tag{20}$$

Next, we theoretically analyze the fluctuation in total brightness. If the occurrence of particles is independent for each repetition member, the average and variance of the brightness of each repetition are identical:

$$B_{\rm Avg} = E(B^i) = \alpha, \tag{21}$$

$$B_{\rm Var} = {\rm Var}(B^i) = (1-\alpha)\alpha. \tag{22}$$

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

$$B_{\mathrm{Var}}^{\mathrm{total}}(L_{\mathrm{R}}) = Var\Big(\sum_{i=1}^{L_{\mathrm{R}}} \frac{B^i}{L_{\mathrm{R}}}\Big)$$
$$= \frac{1}{L_{\mathrm{R}}^2}\Big\{\sum_{i=1}^{L_{\mathrm{R}}} Var(B^i) + 2\sum_{i,j,i<j} Cov(B^i, B^j)\Big\}. \quad (23)$$

Since the brightness of each repetition is independent from the other runs, the covariance term $Cov(B^i, B^j)$ can be estimated as zero. This makes the variance decrease with the inverse of the number (L_{R}) of repetitions. Therefore, the deviation in total brightness can be evaluated as:

$$B_{\mathrm{Dev}}^{\mathrm{total}}(L_{\mathrm{R}}) = \frac{B_{\mathrm{Dev}}}{\sqrt{L_{\mathrm{R}}}} = \sqrt{\frac{(1-\alpha)\alpha}{L_{\mathrm{R}}}}. \quad (24)$$

This equation suggests that a minimum repetition level can be calculated if we specify a certain criterion for the deviation. The larger the criterion, the smaller the required repetition level is. The repetition level is closely related to the computational complexity of PBVR, and we can therefore use this variable to achieve the required LOD control for volume rendering. (23) and (24) demonstrate that the repetition level is identical to the squared sub-pixel level, that is, $L_{\mathrm{R}} = L_{\mathrm{S}}^2$. Note that the sub-pixel level is related to the required frame buffer in addition to the computational complexity, which is not preferable to the LOD control.

6 Experimental Results

We conducted experiments to verify the effectiveness of our proposed technique in terms of performance, image quality and memory usage.

6.1 Performance

This experiment used a PC featuring an Intel Core 2 Duo E8500 (3.17 GHz) CPU with 3 GB of RAM and an nVidia GeForce 9800 GT card. The resolution of all rendered images was 512 × 512. The irregular volume datasets used for this experiment are listed in Table 1, and Fig.5 shows the results of rendering for these datasets. The maximum number of tetrahedral cells was 1.38 M.

Fig.5(a) shows "fighter" data. An irregular tetrahedral mesh which is the result of an airflow simulation over a jet fighter from a wind tunnel model developed at NASA Langley Research Center. Fig.5(b) shows "blunt" data. It is a simulation result of airflow over a flat plate with a blunt fin rising from the plate. Fig.5(c) shows "post" data. The incompressible liquid oxygen flows across a flat plate with a cylindrical post rising perpendicular to the plate (and therefore the flow). The simulation is modeling a flow internal to a rocket engine. A space shuttle launch vehicle engine has a region in which many such posts obstruct flow of liquid oxygen to promote better mixing. Fig.5(d) shows "aorta" data. This is the result of blood flow simulation in an aorta with aneurysm.

Table 1. Performance for Tetrahedral Data

Data	No. Tets.	No. Nodes	No. Particles	Sampling (ms)	Rendering (fps)
Fighter	70 125	13 832	8 878 275	1 906	14.0
Blunt	222 414	40 948	13 121 261	2 922	11.5
Post	616 050	108 300	15 110 048	3 281	11.0
Aorta	1 386 882	248 992	10 782 231	2 656	12.6

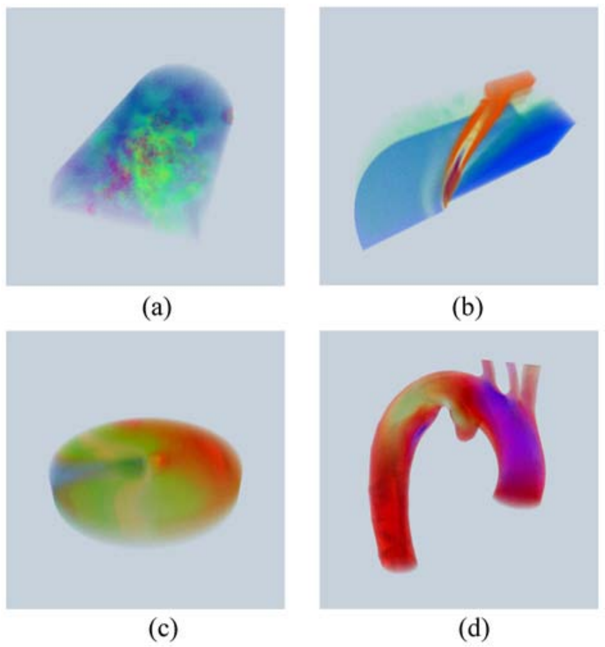

Fig.5. Rendering result for tetrahedral data. (a) Fighter. (b) Blunt. (c) Post. (d) Aorta.

Table 1 also shows performance results in terms of the number of generated particles, particle generation time (sampling time), and rendering speed by setting the repetition level to 144. In this technique, performance is related to the number of irregular volume cells and to the number of generated particles. The latter depends on the density function, which is described consistently with the transfer function.

6.2 Image Quality and Fluctuation

This and later experiments used a commodity PC featuring an Intel Core 2 Duo E6750 (2.66 GHz) CPU with 2.0 GB RAM and an nVidia GeForce 8500 GT card.

To evaluate the image quality of this new PBVR technique, the error values between the PBVR image and the volume ray-casting image were calculated. The error value is defined as the average distance between two images in RGB space, and only active pixels are considered in calculating the error averages. Fig.6 shows the distribution of error value with respect to repetition level, demonstrating that the error values are very close to each other. This means that the ensemble average generates almost the same image as the sub-pixel average.

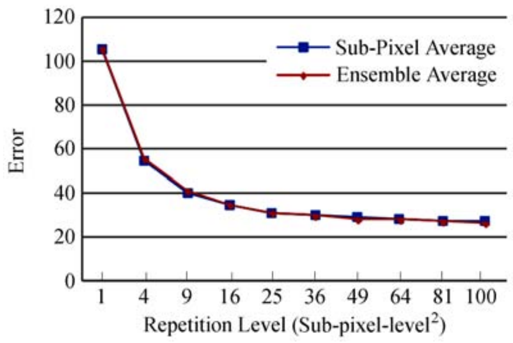

Fig.6. Error distribution in sub-pixel and ensemble averages.

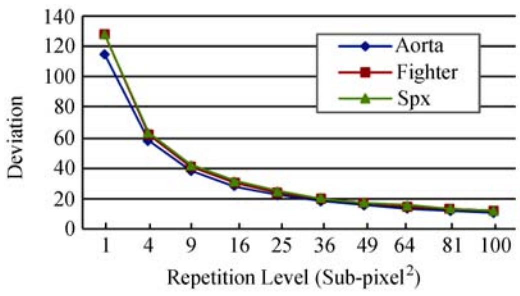

Fig.7. Fluctuation of the ensemble average.

Subsection 5.2 demonstrated that the fluctuation in brightness values is inversely proportional to the square root of the repetition level by using a simple case in which the particle density is constant. For actual volume datasets that feature a complex distribution of particle densities, it is preferable to investigate whether the fluctuation is distributed with respect to the repetition level by analyzing the generated images. In this experiment, the PBVR rendering calculations were performed twenty times at each repetition level by changing the seed of the random number generator. Two types of images were calculated, the average and the deviation. The average image was constructed by averaging twenty images, and the deviation image was constructed by taking the absolute difference between the average and one of the twenty images. Fig.7 shows that the average pixel values of the deviation images decrease with repetition level, consistent with the rendering results for these datasets. These results show that the deviation curves fit well to an inverse square root curve.

6.3 Application to Large-Scale Irregular Volume Datasets

To visualize large-scale CFD and CSM datasets, we develop a distributed implementation of PBVR that is useful for efficiently handling large irregular volume datasets. To allow efficient handling, we employed a cell-by-cell particle generation technique described in Section 5. 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.

The CFD dataset "Oral" is composed of hexahedral cells, and represents the oral airflow simulation result. In the simulation model, the oral cavity shape of the dental fricative was obtained by a Cone Beam CT (CBCT) scanner. Using volume datasets derived from image slices, the oral cavity can be extracted using two threshold CT values, and the required hexahedral cells can be constructed for a large eddy CFD simulation. The Oral dataset is composed of 71 449 236 hexahedral cells with 74 452 754 nodes, and it is divided into 16 datasets as outputs generated from the distributed CFD computation.

The irregular volume dataset of the CSM simulation "Pump" is a result of the analysis of the elastic deformation caused by its own weight with one hundred million degrees of freedom using a PC cluster. The Pump dataset is composed of 26 289 770 quadratic tetrahedral cells with 36 728 129 nodes, and it is divided into 32 datasets as outputs generated from the distributed CSM computation.

Only surface-based visualization has been conducted to date, since it is currently difficult for an available volume rendering software to deal with a large-scale irregular volume dataset such as Oral or Pump dataset. This distributed PBVR was therefore applied to render these datasets. Fig.8 shows the rendering result of the Oral dataset. Fig.8(a) shows the absolute value of the airflow vector and Fig.8(b) shows the pressure. The volume and its boundary surface are simultaneously rendered. The inner boundary surfaces of the portion of the volume dataset can be clearly seen.

In order to verify the effectiveness of the ensemble average, we compared the memory usage and rendering speed of the sub-pixel average and proposed ensemble average methods for the Oral volume dataset. The screen resolution is 512×512. Table 2 reports the results, including the frame buffer size (MB) and rendering speed (fps) for each averaging technique. The table also shows the number of generated particles (M) and its memory usage (MB) for both average techniques. The ensemble average process is faster when the repetition level is larger than 8. In addition, the sub-pixel average process cannot render the Oral dataset in this environment when the repetition level exceeds 64. From Table 2, we can verify that the ensemble average saved more memory usage than the sub-pixel average.

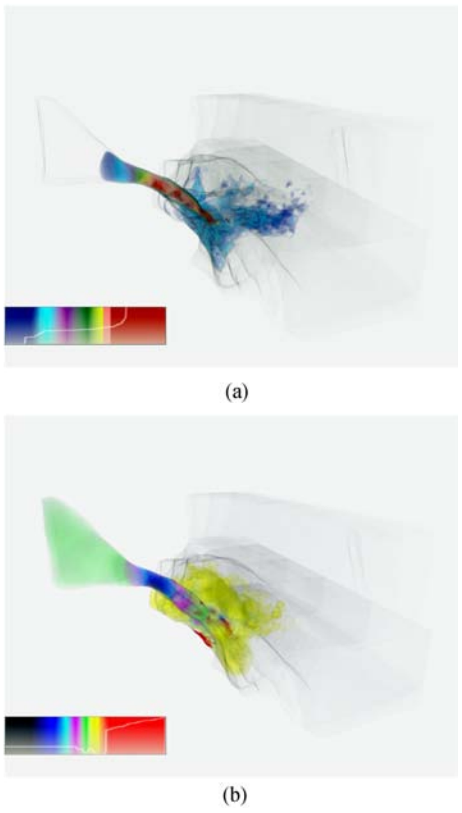

Fig.8. Rendering result of the Oral dataset and its transfer function. (a) Absolute value of the velocity of the airflow. (b) Pressure of the airflow.

In order to evaluate the effectiveness of the particle generation technique, we compared the image quality and particle generation time of the metropolis sampling-based technique to those of the uniform sampling-based one. Fig.9 shows the close-up images of the Oral dataset generated by both techniques. The number of generated particles and generation times are shown in Table 3. We can confirm that the image quality is improved by the metropolis sampling-based technique since some blocky noise exists in uniform sampling. In the supplemental video material (see http://www.youtube.com/watch?v=uH_FsuGYme0), our PBVR shows that the pressure field has a local maximum close to the frontal teeth in the oral cavity using our tiled display wall.

Table 2. Performance for Sub-Pixel and Ensemble Averages

Subpixel Level	2	4	6	8	10	12	14
Frame Buffer	7.00	28.00	63.00	112.00	175.00	252.00	343.00
fps	172.86	52.90	21.45	8.22	NA	NA	NA
Repetition Level	4	16	36	64	100	144	196
Frame Buffer	2.50	2.50	2.50	2.50	2.50	2.50	2.50
fps	150.29	41.50	18.93	10.10	6.45	4.01	2.69
No. Particles (M)	0.17	0.69	1.55	2.76	4.32	6.22	8.46
Particle Mem. (MB)	2.96	11.83	26.66	47.44	74.17	106.71	145.21

Table 3. Particle Generation Time and the Number of Particles for Metropolis Sampling and Uniform Sampling

	Metropolis	Uniform
Sampling Time (s)	25.75	24.07
No. Particles (M)	31.89	

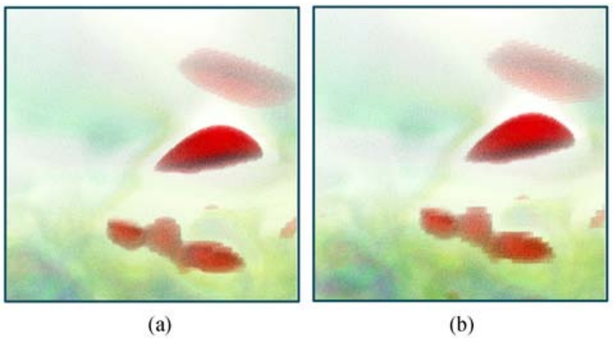

Fig.9. Close-up images of the Oral pressure dataset. (a) Metropolis sampling. (b) Uniform sampling.

Fig.10 shows the rendering results of the Pump dataset as well as the LOD rendering results. Fig.10(a) is rendered with the repetition level 2 and Fig.10(b) is rendered with the repetition level 100. A hundred sets of particles are generated with the repetition level 100 in advance, and only two sets are used as the ensemble members in order to render them with the repetition levels 2. Table 4 shows the number of generated particles and its sampling time. Table 5 shows the rendering speed at the repetition levels 2 and 100. The supplemental video material (see http://www.youtube.com/watch?v=uH_FsuGYme0) shows the LOD rendering of the Pump and Oral datasets.

Table 4. Particle Generation Time and the Number of Particles for the Pump Dataset

Repetition Level	Sampling Time (s)	No. Particles
100	47.65	13 097 195

Table 5. Performance for LOD Rendering of the Pump Dataset

	Coarse Rendering	Fine Rendering
Repetition Level	2	100
fps	93.73	1.58

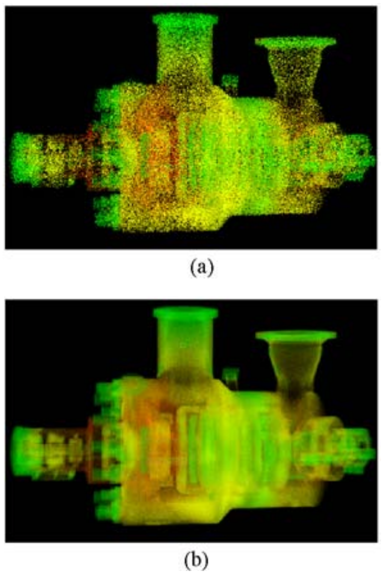

Fig.10. LOD control for rendering of the Pump dataset. (a) Repetition level 2 to move. (b) Repetition level 100 to stop.

7 Discussion

Sub-pixel processing requires additional memory space for large frame buffers. A sub-pixel level of 10 requires 100 times as many frame buffers in order to store sub-pixel values. By using ensemble averages of particles, the PBVR becomes more efficient with respect to memory usage and exhibits a small increase in rendering time in comparison with the sub-pixel processing PBVR when the repetition level is relatively small as shown in Table 3.

The performance of the proposed technique is influenced by the number of particles. The number of particles changes as the visualization parameters are modified. If a transfer function or the pixel size in the global coordinates changes, the density function will also need to be changed, and the number of particles may exceed the system capacity. In this case, we can adopt a streaming approach to render irregular volumes; not all particles are stored in the main memory, but instead are generated in each irregular volume cell and projected onto the image plane, in both this approach and the PT technique. (In addition, we can employ the ensemble average to develop a sort-free PT technique.)

In the density emitter model, a viewing ray is approximated as a cylinder and the particle luminosity remains constant. However, it is natural to use a cone whose top becomes a viewing point and to use a particle whose luminosity is inversely proportional to the squared distance from the viewing point. Since the number of particles in the ray segment volume is proportional to the square of the distance in this case, the total particle luminosity is independent of the distance. Thus, the cylinder viewing ray with constant luminosity particles is equivalent to the cone approach where luminosity decreases with distance. This equivalence will hold when the viewing point is far away enough from the nearest point of the volume. The PBVR keeps the radius of the particle constant in the ray space. However, when the PBVR zooms up in the volume space, the resulting image will become transparent and more particles need to be generated. This is because the particle density has been underestimated near the viewpoint and will become a negative factor for the interactive rendering. We plan to develop a new method of constructing the particle radius on the image plane according to the distance from the image plane. It can be easily implemented by rendering the particles in a common camera projection process.

8 Conclusion

This work presents a metropolis sampling-based technique for generating particles in an irregular volume cell and an ensemble average technique for conserving the frame buffer memory usage in order to improve the PBVR process. The previous PBVR approach had problems with respect to image quality and memory usage when rendering a large-scale irregular volume dataset. The former issue is that the cell boundary may become prominent when the density function changes significantly in an irregular volume cell. The latter problem is that the required frame buffer tends to be large when the sub-pixel level increases. In addition, the ensemble average can in turn be utilized to control the LOD of volume rendering. We do not intend to replace the ray-casting technique with our proposed technique. Instead, we envisage it as being useful for previewing large irregular volume datasets with minimal preprocessing.

This technique scales well to large irregular volume datasets with reasonable memory requirements. The only restriction is that there should be sufficient memory to store at least the information of a single irregular volume cell. Each cell may be processed to generate particles that can be projected in an arbitrary order. Simulations using huge irregular volumes are usually run in parallel on clusters of high-bandwidth supercomputers or PCs. The resulting datasets can be so massive that they require parallel computing resources of similar magnitude to effectively visualize them. A streaming-based approach is one promising solution for meeting such a requirement, and this technique is a natural fit for such an approach.

The transfer function exploration is important for the volume rendering. When we modify the opacity function, the particles need to be regenerated from scratch. It may take a considerable computation time. To address this issue, we will improve the performance of the particle generation process by using two techniques. The first one is to employ the parallel processing in the process. Our particle generation will be efficiently parallelized since it can be independently processed at each cell. The second one is to reduce the number of particles by modifying the opacity function so that its feature can be preserved.

Acknowledgement We thank the anonymous reviewers for their valuable comments that have helped to improve the manuscript.

References

[1] Sabella P. A rendering algorithm for visualizing 3D scalar field. *Computer Graphics*, 1998, 22(4): 51-58.
[2] Sakamoto N, Nonaka J, Koyamada K, Tanaka S. Particle-based volume rendering. In *Proc. Asia-Pacific Symposium on Visualization*, Sydney, Australia, Feb. 5-7, 2007, pp.141-144.
[3] Koyamada K, Sakamoto N, Tanaka S. A particle modeling for rendering irregular volumes. In *Proc. International Conference on Computer Modeling and Simulation*, Cambridge, UK, April 1-3, 2008, pp.372-377.
[4] Shirley P, Tuchman A, A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 1990, 24(5): 63-70.
[5] Williams P. Visibility-ordering of meshed polyhedra. *ACM Transaction on Graphics*, 1992, 11(2): 103-126.
[6] Meredith J, Ma K L. Multiresolution view-dependent splat-based volume rendering of large irregular data. In *Proc. IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, San Diego, USA, Oct. 22-23, 2001, pp.93-99.
[7] Callahan S, Ikits M, Comba J, Silva C. Hardware-assisted visibility ordering for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2005, 11(3): 285-295.
[8] Anderson E W, Callahan S P, Scheidegger C E, Schreiner J, Silva C T. Hardware-assisted point-based volume rendering of tetrahedral meshes. In *Proc. Brazilian Symposium on Computer Graphics and Image Processing*, Belo Horizonte, Brazil, Oct. 7-10, 2007, pp.163-170.
[9] Roetter S, Ertl T. Cell projection of convex polyhedra. In *Proc. IEEE TVCG Workshop on Volume Graphics*, Tokyo, Japan, 2003, pp.103-107.
[10] Csebfalvi B, Szirmay-Kalos L. Monte carlo volume rendering. In *Proc. IEEE Visualization*, Seattle, USA, Oct. 19-24, 2003, pp.449-456.
[11] Csebfalvi B. Interactive transfer function control for Monte Carlo volume rendering. In *Proc. IEEE Symposium on Volume Visualization and Graphics*, Austin, USA, Oct. 11-12, 2004, pp.33-38.
[12] Zhou Y, Garland M. Interactive point-based rendering of higher-order tetrahedral data. *IEEE Transactions on Visualization and Computer Graphics*, 2006, 12(5): 1229-1236.
[13] Ding Z, Kawamura T, Sakamoto N, Koyamada K. GPU acceleration of improved particle-based volume rendering for irregular-grid volume data. In *Proc. International Conference on System Simulation and Scientific Computing*, Chengdu, China, Oct. 17-19, 2008, pp.685-692.
[14] Gallapher R S (ed.). Computer Visualization. CRC Press, 1995.

Takuma Kawamura graduated from Future University-Hakodate in 2006 with a B.S. degree in system information science. He received the M.S. degree in Graduate School of Engineering from Kyoto University in 2008. He is currently a Ph.D. candidate in Graduate School of Engineering, Kyoto University.

Naohisa Sakamoto received the Master's degree from Ryukoku University in 2001 and the Ph.D. degree in Graduate School of Engineering from Kyoto University in 2007. From 2001 to 2002, he worked for KGT (Kubota Graphics Technology) Inc. He was a research scientist at Center for the Promotion of Excellence in Higher Education (CPEHE) at Kyoto University from 2003 to 2008. Since April 2008, he is an associate professor with CPEHE at Kyoto University. His research interests include scientific visualization.

Koji Koyamada is a professor in electrical engineering at Kyoto University. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from Kyoto University in 1983, 1985, and 1994 respectively. He joined IBM Japan in 1985 and worked for research on scientific computation and scientific visualization. He was an assistant professor of Iwate Prefectural University from 1998 to 2001. He became an associate professor and professor of Kyoto University in 2001 and 2003 respectively. He is a member of the Visualization Society of Japan, the IEEE Computer Society, and the Information Processing Society of Japan, Japan Society for Simulation Technology.