GRID-INDEPENDENT METROPOLIS SAMPLING FOR VOLUME VISUALISATION

Satoshi Tanaka[1], Takuya Hatta[1], Frederika Rambu Ngana[1], Ayumu Saitoh[1], Naohisa Sakamoto[2], Jorji Nonaka[3], Koji Koyamada[2]

[1]Ritsumeikan University, College of Information Science and Engineering,
Noji-higashi 1-1-1, Kusatsu-shi, Shiga 525-8577, Japan
[2]Kyoto University, Academic Center for Computing and Media Studies,
Yoshida Nihonmatsu-cho, Sakyo-ku, Kyoto 606-8501, Japan
[3]KGT Inc, Tomin Shinjuku Bldg., Shinjuku 2-8-8, Shinjuku-ku, Tokyo 160-0022, Japan

stanaka@media.ritsumei.ac.jp(Satoshi Tanaka)

Abstract

We propose a method of sampling regular and irregular-grid volume data for visualisation. The method is based on the Metropolis algorithm that is a type of Monte Carlo technique. Our method enables 'importance sampling' of local regions of interest in the visualisation by generating sample points intensively in regions where a user-specified transfer function takes the peak values. The generated sample-point distribution is independent of the grid structure of the given volume data. Therefore, our method is applicable to irregular grids as well as regular grids. We demonstrate the effectiveness of our method by applying it to regular cubic grids and irregular tetrahedral grids with adaptive cell sizes. We visualise volume data by projecting the generated sample points onto the 2D image plane. We used three rendering models: an X-ray model, a simple illuminant particle model, and an illuminant particle model with light-attenuation effects. We also demonstrate that our method is suitable for parallel processing, since it realizes computation speed almost proportional to the number of processors. The grid-independency and the efficiency in the parallel processing mean that our method is suitable for visualizing large-scale volume data. The former means that the required number of sample points is proportional to the number of 2D pixels, not the number of 3D voxels. The latter means that our method can be easily accelerated on the multiple-CPU and/or GPU platforms.

Keywords: Metropolis algorithm, importance sampling, volume visualisation, grid-independent sampling, regular/irregular grid

Presenting Author's Biography

Satoshi Tanaka. In 1987, he took a Ph.D at Waseda University, Japan, majoring in theoretical physics. In 1992, he moved to Fukui University, Japan, and started research activities on visualization of high-energy-physics simulation, point rendering with Monte Carlo simulation, etc. Since 2002 he has been a professor of Ritsumeikan University, Japan. His current research fields are Monte Carlo simulation, shape modeling, computer graphics, and computer visualization. He is a member of Japan Society for Simulation Technology (JSST) and Eurographics.

ISBN 978-3-901608-32-2

Fig. 1 Sample points generated by our method for the $301 \times 324 \times 56$ lobster (regular-grid data).

1 Introduction

Direct volume rendering is widely used to extract useful information from various kinds of volume data, e.g., medical data, fluids, and energy distribution. To render (i.e., visualise) volume data, we need to generate sample points that represent the entire data set beforehand.

For cubic regular grids, the sample-point distribution is usually made uniform over the whole grid space. In rendering with object-order approaches such as splatting [1] or cell projection [2, 3], we can regard the regularly allocated voxels or cubes themselves as sample points. In an image-order approach such as ray casting[4, 5] sample points are generated at regular intervals along each ray.

For irregular grids, in which the shapes and/or sizes of volume cells are non-uniform, it is not a trivial task to define a proper sample-point distribution. The simplest way is to adopt a uniform sample-point distribution just as for the regular grids. However, the distance between neighboring sample points should be as small as the width of the smallest cell, so that every cell can be is sampled. However, by doing this, we may generate many unnecessary sample points, resulting in an increased rendering time. Another way is to generate sample points adaptively according to a voxel (grid-point) distribution. In the object-order approach, we can use voxels themselves as sample points. More intelligent approaches based on the octree subdivision of grid space are also possible [6]. In the image-order approach, we can trace rays through volume cells, and we use the ray-cell intersection points or the midpoints of ray segments in the cells as sample points [7].

As described above, a sample-point distribution is usually decided either uniformly or based on a voxel distribution. In this paper, we propose a method to distribute sample points according to a density function that is obtained by deforming the given volume data based on a user-specified 'transfer function' and then executing interpolation. Our method thus enables 'importance sampling' of local regions where the transfer function takes peak values. Such regions are usually of interest in visualizing volume data. To realize such a sample-point distribution, we use the Metropolis algorithm [8], which is known as an efficient Monte Carlo technique in chemistry, physics, and other fields. It is worth noting that the Metropolis algorithm is also suitable for parallel processing. We will demonstrate that the rendering speed increases almost linearly as the number of processors increase.

It should be noted here that the sample-point distribution realized by our method is independent of voxel distribution. This means that the sample-point distribution is independent of a given grid structure. Therefore, our method is applicable to irregular grids as well as to regular grids. In this paper, we demonstrate the effectiveness of our method by applying it to adaptive tetrahedral grids [9], in which the sizes of the tetrahedral cells are not uniform.

The above-mentioned grid-independency, as well as the efficiency in the parallel processing, means that our method is suitable for visualizing large-scale volume data. Suppose that we render volume data with size N^3. The grid-independency means that we need not traverse all the N^3 voxels, being different from the classical methods such as the ray casting [4, 5] or splatting [1]. (The classical methods can be accelerated such that the calculational time is proportional to $N^2 \log N$ [10].) In fact the number of Monte Carlo averages that are calculated in our method is identical to the number of pixels, which means that the number of required sample points is proportional to N^2. Therefore our method becomes more advantageous for large-scale data, moreover, enabling their importance sampling of the regions of interest.

Our method can generate high-density sample points intensively in the regions of interest. We use such 'adaptive high-density points' for volume rendering, which differs from the classical object-order approaches such as splatting [1] or cell projection [2, 3]. We render (visualise) the volume data effectively by projecting the adaptively distributed sample points onto the 2D image plane. Fig.1 shows sample points generated by our method. In the $301 \times 324 \times 56$ regular-grid space, a portion of the lobster is sampled intensively.

The organization of this paper is as follows. In Section 2, we explain the concept of our method. In Section 3, we apply our method to regular-grid and irregular-grid data. Section 4 is the concluding section.

Related works

Csébfalvi and Szirmay-Kalos proposed "Monte Carlo volume rendering" [11]. In their method, a voxel is randomly selected, and then sample points are generated locally around the voxel. (Their way of sampling is similar to the interleaved sampling [12], which was proposed to reduce inter-pixel aliasing.) The generated sample points are projected onto the image plane. Then, the number of points projected into each pixel is simply counted, and the normalised pixel color (pixel intensity) is made proportional to the number. This method

enables importance sampling of the regions around the peaks of the transfer function, as does our method. The concept of this method is interesting, and our study was inspired by it. However, it is not a trivial job to apply the method to general irregular grids, since it assumes a uniform cubic distribution of voxels. The luminosity of sample points, i.e., the particle luminosity, is not considered, either.

Sakamoto and Koyamada also proposed a method of volume rendering based on Monte Carlo and point-based techniques [13]. In their method, fully opaque particles are generated with the hit-and-miss algorithm and projected onto the image plane. Semi-transparent rendering effects are realized by particle occlusion and sub-pixel averaging. Their method is simple and quite fast, but the statistical accuracy that influences the rendering quality still needs to be improved. Applicability of the method is also restricted to cubic regular grids.

Our method, which also uses a Monte Carlo technique, is applicable to general irregular grids as well as cubic regular grids. The number of particles contributing to pixel colors is much larger than the method of reference [13], and thus realizes higher statistical accuracy, i.e., higher-quality rendering.

2 Metropolis sampling and rendering

2.1 Transfer function and density function

For the sampling of volume data, we utilize the Metropolis algorithm [8], which can rapidly generate points according to a given continuous density function $\rho(\mathbf{x})$. Our sampling method is applicable to either regular-grid or irreglar-grid data with the same form, once a continuous $\rho(\mathbf{x})$ is defined. The functional form of $\rho(\mathbf{x})$ is decided as follows.

First, we map (deform) given discrete volume data (voxel values) to proper scalar values by using a user-specified transfer function. It is to highlight the regions of interest by amplifying our focusing voxel values and also making the unnecessary voxel values vanish.

Next, we define $\rho(\mathbf{x})$ by interpolating the scalar values obtained in the above. In this paper, we adopt the standard trilinear interpolation for regular cubic grids, and the barycentric interpolation [14, 9] for tetrahedral (irregular) grids. Any interpolation, which defines a continuous functional form of $\rho(\mathbf{x})$, is available.

Fig.2 is an example of the transfer function that highlights regions with volume data (voxel values) between 100 and 150. By tuning the peak position and the peak width, we can easily select regions to be highlighted. Fig.3 shows an example of highlighting different regions of hydrogen data. A transfer function similar to Fig.2 is used, and the peak is properly tuned to obtain the two different rendering images. The rendering is done with the generated high-density sample points and the X-ray model explained in Section 2.3.1.

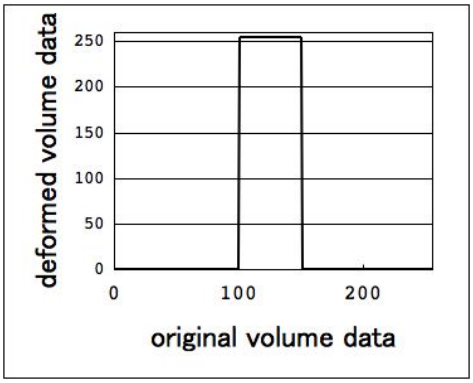


Fig. 2 Transfer function to highlight regions with volume data (voxel values) between 100 and 150.

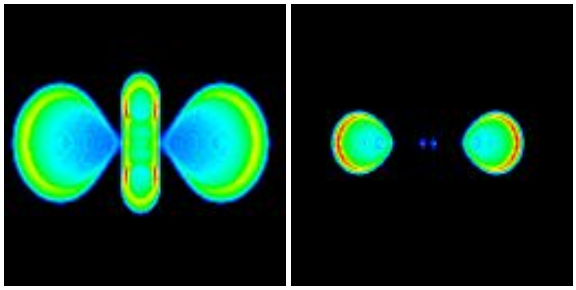


Fig. 3 Highlighting different regions of hydrogen volume data by tuning the peak of the transfer function of Fig.2.

2.2 Metropolis sampling of volume data

Once a functional form of $\rho(\mathbf{x})$ is decided, we can generate sample points, following the prescription of the Metropolis algorithm, as follows, starting at an arbitrary initial position $\mathbf{x}_0$:

- **Step 1:** Calculate $\rho(\mathbf{x}_i)$ for the current (ith generated) sample-point position, $\mathbf{x}_i$.
- **Step 2:** Generate a candidate (trial) position, $\mathbf{x}'$, which is chosen randomly inside the grid space, and calculate $\rho(\mathbf{x}')$.
- **Step 3:** Calculate the ratio of $\rho(\mathbf{x}')$ to $\rho(\mathbf{x}_i)$, i.e.,

$$W(\mathbf{x}_i \rightarrow \mathbf{x}') \equiv \rho(\mathbf{x}')/\rho(\mathbf{x}_i)\ . \qquad (1)$$

- **Step 4:** If $W(\mathbf{x}_i \rightarrow \mathbf{x}') \geq 1$, accept $\mathbf{x}'$ as an updated sample-point position, $\mathbf{x}_{i+1}$, and go back to Step 1. Otherwise, go on to Step 5.
- **Step 5:** Generate a uniform random number $R \in [0, 1]$, and determine $\mathbf{x}_{i+1}$ as below.

$$\mathbf{x}_{i+1} = \begin{cases} \mathbf{x}' & \text{if } W(\mathbf{x}_i \rightarrow \mathbf{x}') \geq R \\ \mathbf{x}_i & \text{otherwise} \end{cases} . \qquad (2)$$

Then go back to Step 1.

ISBN 978-3-901608-32-7

The description of Step 2 should be supplemented with three additional rules for generating the trial sample-point position $\mathbf{x}'$:

- **Rule 1:** The trial sample-point position $\mathbf{x}'$ is generated as follows, adding a random vector to the current sample-point position $\mathbf{x}_i$:

$$\mathbf{x}' = \mathbf{x}_i + \Delta\mathbf{x}_i \ , \qquad \Delta\mathbf{x}_i = \lambda(\eta_x^i,\ \eta_y^i,\ \eta_z^i) \ , \quad (3)$$

  where η_x^i, η_y^i, and η_z^i are independent uniform random numbers in a range $[-1, 1]$, and λ is a positive constant to decide the average sampling width. We choose $\lambda = 5$.

- **Rule 2:** If $\mathbf{x}'$ generated according to Rule 1 is outside the grid space, we abandon it. Instead we randomly select a voxel position where a non-zero scalar value of ρ is assigned, and we use the position as $\mathbf{x}'$.

- **Rule 3:** We sometimes add an effect of 'random jump' to the sampling. Namely, once in dozens of sampling steps, we randomly select a voxel position where a non-zero scalar value of ρ is assigned, and we adopt it as $\mathbf{x}'$ as in Rule 2. This rule should not be applied too frequently, so that it does not influence the sample-point distribution. For example, once in every 30 sampling steps seems to be an appropriate frequency.

Let us explain the meanings of the above rules one by one.

Rule 1 describes the standard way of generating the trial sample-point position, $\mathbf{x}'$. We choose $\lambda = 5$, which means that the average sampling width is sufficiently larger than the minimal cell size. For efficiency of sampling, $\lambda \sim$ (several) $\times$ (minimum cell size) seems to be an appropriate choice, although theoretically the Metropolis algorithm should work with an arbitrary value of λ.

Rule 2 describes the 'boundary condition.' Rule 1 makes the sample-point position update by adding random vectors recursively. Then $\mathbf{x}'$ sometimes steps out of the grid space. In such an exception, we need another rule to redefine $\mathbf{x}'$. For sampling efficiency, i.e., to accelerate updating of the sample-point position, we randomly select a voxel position, where a non-zero scalar value of ρ is assigned, as $\mathbf{x}'$. This increases the ratio that $\mathbf{x}'$ is accepted, and so accelerates the sampling. Rule 2 can be executed quickly by preparing a table of voxels with non-zero scalar values beforehand as preprocessing.

Finally, Rule 3 describes the 'random jump' to guarantee sampling of the whole grid space. If the grid space has more than one disconnected regions with non-zero $\rho(\mathbf{x})$, and $\rho(\mathbf{x})$ vanishes in the intermediate regions, then the sample-point position $\mathbf{x}$ may not travel from one non-zero-$\rho(\mathbf{x})$ region to another. This is because the otherwise of Step 2 is realized in the intermediate regions where $\rho(\mathbf{x}) = 0$. Fortunately this problem can be solved by sometimes making the sample-point position jump randomly in the grid space. Rule 3 is also effective in accelerating sampling, even if there are not disconnected non-zero-$\rho(\mathbf{x})$ regions.

In Fig.4, we show an example of sampling $301 \times 324 \times 56$ lobster (regular-grid data). The 0.1M (million) sample points generated are plotted three-dimensionally. The original volume data are directly interpolated with the trilinear interpolation to define a continuous density function $\rho(\mathbf{x})$, which we use for the Metropolis sampling. The sample-point distribution shown in Fig.4 coincides with the functional form of $\rho(\mathbf{x})$, i.e., the distribution of the voxel values in the original volume data.

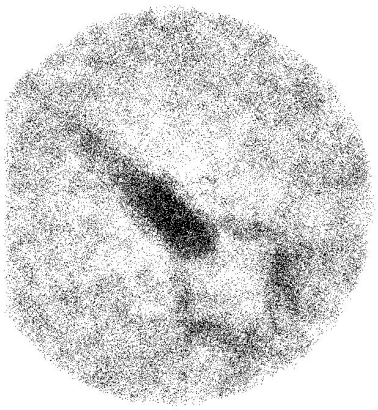


Fig. 4 Sample points generated by our method for the $301 \times 324 \times 56$ lobster (regular-grid data). The original volume data are directly interpolated to define a continuous density function $\rho(\mathbf{x})$.

Fig.1 shows a similar sample-point distribution in the case when the transfer function of Fig.5 is applied. The original volume data are deformed based on this transfer function that works as a simple low-cut filter. Then the deformed data are interpolated to define $\rho(\mathbf{x})$ for the Metropolis sampling. We can clearly see that sample points are not generated outside the lobster, although the original volume data have small voxel values there.

We show another example of the Metropolis sampling in Fig.6 (left). The 0.1M sample points generated for the $256 \times 256 \times 256$ tornado are plotted three-dimensionally. In the sampling the transfer function in the right figure is applied. We can clearly see that the core part of the tornado is intensively sampled with the effect of the transfer function.

As demonstrated in the above, we can control the sample-point distribution with the transfer function, being independent of a given grid structure.

2.3 Determination of pixel colors

A volume-rendering image can be created by projecting sample points onto the image plane, i.e., the 2D pixel space, and determining the pixel colors with a proper

ISBN 978-3-901608-32-7

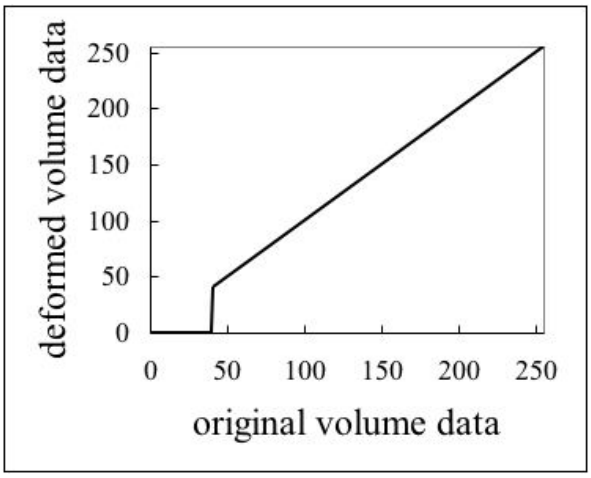


Fig. 5 The transfer function used to create Fig.1. This transfer function works as a simple low-cut filter.

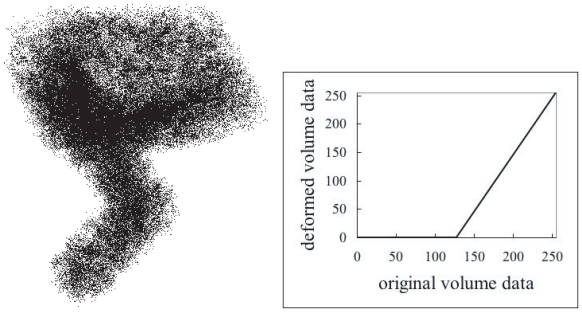


Fig. 6 Left: sample points generated by our method for the $256 \times 256 \times 256$ tornado (regular-grid data). Right: The transfer function adopted for the sampling.

prescription [1, 2, 3, 11]. In this paper, utilizing the sample points generated by our Metropolis algorithm, we investigate three rendering models to determine the pixel colors.

2.3.1 X-ray model

In this rendering model, the number of sample points projected into each pixel is simply counted, and the number directly determines its pixel color (pixel intensity) [11]. A normalised pixel color is made proportional to the number of points projected into the pixel. The explicit expression of the determined color for the pth pixel, $C_p^{(\mathrm{X})}$, is as follows:

$$C_p^{(\mathrm{X})} = \frac{N_p \, I_{\max}}{N_{\max}} \,, \tag{4}$$

where N_p is the number of sample points projected into the pth pixel, $N_{\max}$ is its maximum among all pixels on the image plane, and $I_{\max}$ is the maximum intensity, e.g., $I_{\max} = 255$.

In the X-ray model, an image of the density function $\rho(\mathbf{x})$ directly determines the pixel colors. Therefore, it visualises the distribution of the original volume data straightforwardly. The effects of particle luminosity are not taken into account. In Section 3, we show examples of rendering with the X-ray model

2.3.2 Simple illuminant particle model

In this rendering model, we regard the sample points as 'illuminant particles' [13]. To each particle at position $\mathbf{x}$, we assign an illuminant color (illuminant intensity) that is identical to $\rho(\mathbf{x})$. Then we project the particle into a pixel on the image plane. In each pixel, we calculate an average of the illuminant colors of the projected particles, and identify it with the pixel color. The color of the pth pixel, $C_p^{(\mathrm{A})}$, is determined as

$$C_p^{(\mathrm{A})} = \frac{\tilde{c}_p^{(\mathrm{A})} \, I_{\max}}{\tilde{c}_{\max}^{(\mathrm{A})}} \,, \qquad \tilde{c}_p^{(\mathrm{A})} = \frac{1}{N_p} \sum_{j=1}^{N_p} \rho(\mathbf{x}_j^{(p)}) \,, \tag{5}$$

where $\tilde{c_p}^{(\mathrm{A})}$ is an unnormalised color of the pth pixel, $\tilde{c}_{\max}^{(\mathrm{A})}$ is its maximum among all pixels, and $\mathbf{x}_j^{(p)}$ is a 3D position of the jth particle projected into the pth pixel.

In this rendering model, we regard the value of $\rho(\mathbf{x})$ as the particle luminosity. Since our Metropolis sampling generates high-density sample points around the peaks of $\rho(\mathbf{x})$, 3D regions with high $\rho(\mathbf{x})$ make a major contribution to the pixel colors. (It is also easy to realize shaded colors by incorporating the effects to $\rho(\mathbf{x})$ before the projection.) We will show examples of applying this rendering model in Section 3.

Mathematically, the normalized average Eq.(5) determining the pixel color $C_p^{(\mathrm{A})}$ is the average of $\rho(\mathbf{x})$ along the ray passing though the pixel. Since the averaging weight is also $\rho(\mathbf{x})$, $C_p^{(\mathrm{A})}$ corresponds to $\int \rho^2 d\ell / \int \rho d\ell$ with ℓ a coordinate along the ray. This integration can be approximated as $\int \rho^2 d\ell / \int \rho d\ell \simeq (\int \rho d\ell)^2 / \int \rho d\ell = \int \rho d\ell$, which is the simple integration of ρ along the ray. Therefore, the simple illuminant particle model should produce images similar to the ones by the conventional ray-casting rendering without opacity accumulation.

2.3.3 Illuminant particle model with light-attenuation effects

This model is similar to the above, but light-attenuation effects are taken into account. The difference appears in how to calculate averages to determine the pixel colors. The 'weighted average' is introduced such that light coming from deep inside the volume data gives little contribution to the pixel colors. This realizes an effect similar to the accumulation of opacity in the ray-casting rendering without necessity of the depth sort of the particles (sample points). The color of the pth pixel, $C_p^{(\mathrm{W})}$, is determined as

$$C_p^{(\mathrm{W})} = \frac{\tilde{c}_p^{(\mathrm{W})} \, I_{\max}}{\tilde{c}_{\max}^{(\mathrm{W})}} \,, \; \tilde{c}_p^{(\mathrm{W})} = \frac{\sum_{j=1}^{N_p} w_p(z_j^{(p)}) \rho(\mathbf{x}_j^{(p)})}{\sum_{j=1}^{N_p} w_p(z_j^{(p)})} \,, \tag{6}$$

where $z_j^{(p)}$ is the z component of $\mathbf{x}_j^{(p)}$, and we choose a proper coordinate system such that the particles are

projected in the z direction. The weight function $w_p(z)$ is defined by

$$w_p(z) = \frac{a}{(z_{\max}^{(p)} - z)^2 + a}\ , \qquad (7)$$

where a is a positive constant to control the rapidity of light attenuation, and $z_{\max}^{(p)}$ is the maximum z of the non-zero-$\rho(\mathbf{x})$ region along the z-directional line passing through the pth pixel. For each pixel, $z_{\max}^{(p)}$ is calculated and stored beforehand as preprocessing. The calculation of $z_{\max}^{(p)}$ for each pixel can be executed quite quickly by investigating the scalar values of the original voxels and then by making the proper interpolation. In Fig.7, we illustrate the prescription of this rendering model schematically. (Zhou and Garland[15] proposed a similar way of the weighted average in their point-based volume rendering. But their $z_{\max}$ is constant for all the pixels.)

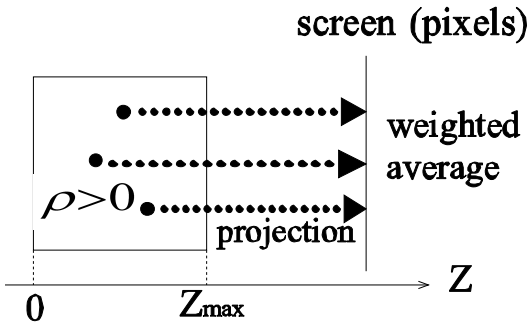


Fig. 7 Schematic illustration of the illuminant particle model with light-attenuation effects. The $z_{\max}^{(p)}$ may be different in each pixel.

The above weight function $w_p(z)$ takes its maximum value, 1, for $z = z_{\max}$, i.e., on the boundary of the non-zero-$\rho(\mathbf{x})$ region. Therefore, this rendering model is useful to visualise intensively regions near the surfaces of volume structures. In Section 3, we will compare the results of this rendering model with those of the above two models. For example, we will clearly see lobster legs on its surface of the body in Fig.8(c).

For the same reason as before, the illuminant particle model with light-attenuation effects should also produce images similar to the ones by the conventional ray-casting rendering. But the model can realize effects similar to the opacity accumulation, making surface parts of the volume structures visible clearly.

2.4 Parallel processing

Our method is suitable for the parallel processing with multiple CPUs. The method is easily parallelised based on the simple master-slave model as follows:

1. Each slave process executes the Metropolis sampling and projection of sample points independently.
2. After the above sampling, each slave process sends the following 2D information to the master process, and the master process finalizes calculation of the pixel colors. The 2D information that is gathered by the master process is as follows:
   - In the X-ray model, N_p is gathered.
   - In the simple illuminant particle model, N_p and $\sum_{j=1}^{N_p} \rho(\mathbf{x}_j^{(p)})$ are gathered.
   - In the illuminant particle model with light-attenuation effects, $\sum_{j=1}^{N_p} w_p(z)$ and $\sum_{j=1}^{N_p} w_p(z)\rho(\mathbf{x}_j^{(p)})$ are gathered.

Note that the inter-process communication occurs only in the second step, i.e., at the end of creating an image. Therefore, most of the computation can be executed independently, which leads to efficient parallelisation. We will show our experiments on the parallel processing in Section 3.

3 Experiments

In this section, we apply our method to the volume data of regular and irregular grids, and demonstrate its effectiveness. We also compare the results of applying the three rendering models described in Section 2.3.

3.1 Application to regular-grid data

We have applied our method to the following three regular-grid volume data: $301 \times 324 \times 56$ lobster, $256\times256\times256$ tornado, and $161\times321\times129$ foot. Their rendering resolutions are 301×324 (lobster), 256×256 (tornado), and 161×321 (foot). Figs.8, 9, and 10 show the rendering results. We can see that the proper rendering images are obtained.

With the X-ray model (Figs.8 (a), 9 (a), and 10 (a)), the density function $\rho(\mathbf{x})$ directly determines pixel colors. Therefore, by using the linear transfer function (with the low-cut filter effect as in Fig.5 if necessary), we can investigate the distribution of the original volume data (voxel values) straightforwardly. The larger a voxel value is, the more it contributes to make the corresponding pixel bright. For example, in Fig.8 (a), we can see that the chest portion of the lobster is brightest, and so the voxel values should be largest there. In Fig.11, we show a colored image created with the X-ray model. The red color corresponds to largest voxel values, while the blue color to smallest voxel values.

With the simple illuminant particle model (Figs.8 (b), 9 (b), and 10 (b)), whole regions of the grid spaces are visualised more clearly. Images similar to the ray-casting rendering without opacity accumulation effects are obtained as expected.

With the illuminant particle model with light-attenuation effects (Figs.8 (c), 9 (c), and 10 (c)), portions near the surfaces of the volume structures are visualised more clearly. Note the legs in the stomach of the lobster (Fig.8 (c)), the central part of the foot bone

ISBN 978-3-901608-32-2

Tab. 1 Computational time for sampling + rendering. 'Illum (1)' indicates the simple illuminant particle model. 'Illum (2)' indicates the illuminant particle model with light-attenuation effects.

	Lobster (5M points)	Foot (10M points)	Tornado (10M points)
X-Ray	3.2 (sec)	7.4 (sec)	7.8 (sec)
Illum (1)	3.3 (sec)	7.6 (sec)	8.0 (sec)
Illum (2)	3.5 (sec)	8.0 (sec)	8.7 (sec)

(Fig.10 (c)), etc. Namely, the images similar to ray-casting rendering with opacity accumulation effects are obtained as expected.

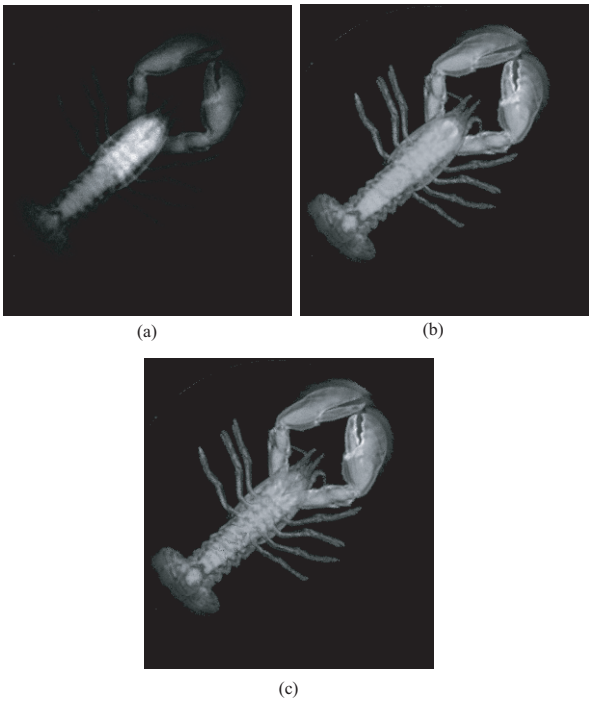


Fig. 8 Rendering of the $301 \times 324 \times 56$ lobster (regular grid) with 5M sample points: (a) X-ray model, (b) simple illuminant particle model, (c) illuminant particle model with light-attenuation effects (a = 50). In the sampling, the transfer function of Fig.5 is used.

Tab.1 summarizes the computation time for creating the images of Figs.8, 9, and 10. The time is measured in our PC with a Pentium Xeon 3.0 GHz processor and 2.0 GB memory. The computation time includes both sampling and rendering time. No hardware acceleration is used for the computation. A few or several seconds are enough to create an image of the current resolution. (We have already started researches on executing our Metropolis sampling with GPU [16]. Their preliminary results show that the sampling is a few or several times faster.)

Fig.12 shows the computation speeds in the parallel

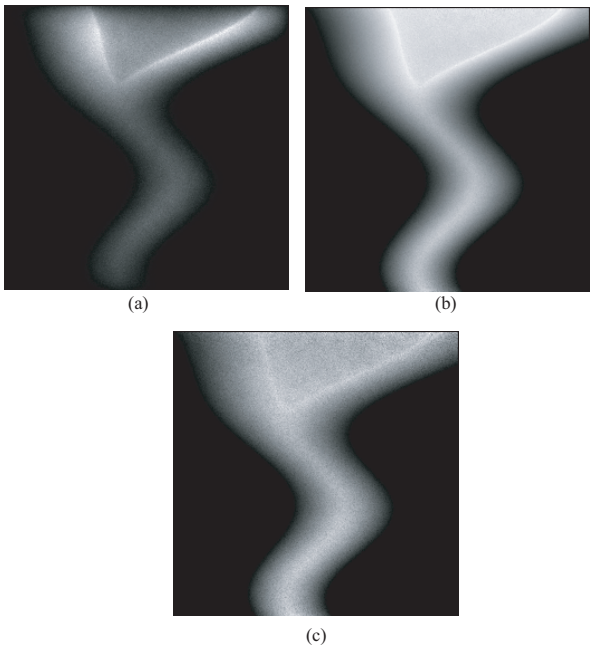
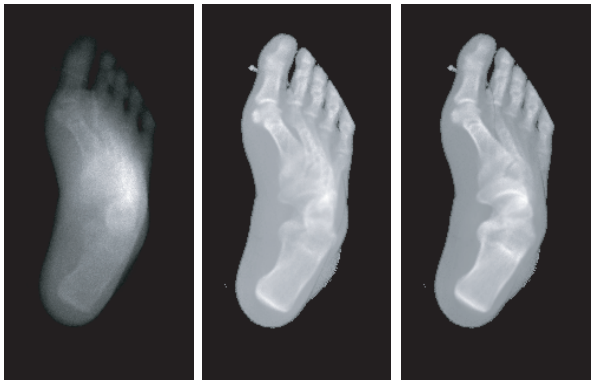


Fig. 9 Rendering of the $256 \times 256 \times 256$ tornado (regular grid) with 10M sample points: (a) X-ray model, (b) simple illuminant particle model, (c) illuminant particle model with light-attenuation effects (a = 300). In the sampling, the transfer function of Fig.6 (right) is used.

processing. The computation time is measured for creating images of the $301 \times 324 \times 56$ lobster (squares) and the $477 \times 477 \times 133$ lobster (diamonds) with 50M sample points. Inverses of the computation time, i.e., the computation speeds, are plotted for the increasing number of slave processes. The computation speed increases almost linearly, reflecting high-level independency of our Metropolis sampling. In Fig.12, we can also see that the computation speed is independent of the volume-data sizes. This is because the Metropolis algorithm makes the sample-point position update based only on local evaluation of $\rho(\mathbf{x})$. These properties, shown in Fig.12, suggest that our method is suitable for rendering large-scale volume data.

The grid-independency of our sampling method also makes it suitable for visualizing large-scale volume data. For volume data with size N^3, our method requires sample points proportional to N^2, i.e., to the image resolution. It is because the number of calculated Monte Carlo averages, which determine pixel colors, is nothing but the number of pixels on the image plane. On the other hand, the classical volume-rendering methods have to generate sample points proportional to N^3, since their sampling is based on traversing the 3D distribution of voxels. Fig.13 shows rendering of volume data with different data sizes. Fig.13 proves that we can keep the same image qualities for the increasing data size by making the number of sample points proportional to N^2. (The sampling should terminate when a sufficient number of sample points accumulates. The 'sufficient' number could be automatically decided

ISBN 978-3-901608-32-2



(a) (b) (c)

Fig. 10 Rendering of the $161 \times 321 \times 129$ foot (regular grid) with 10M sample points: (a) X-ray model, (b) simple illuminant particle model, (c) illuminant particle model with light-attenuation effects (a = 300). A transfer function similar to Fig.5 is used as a low-cut filter with the threshold value 55.

by monitoring the improving quality of the images created.)

3.2 Application to irregular-grid data

We have also applied our method to tetrahedral adaptive-grid data with different cell sizes [9]. These are irregular-grid data in the sense that their cell sizes are not uniform. The adaptive grids are created by properly eliminating parts of the voxels from original regular grids, taking care of local uniformity of the scalar values. Therefore, the voxel positions of a created adaptive grid are a subset of those in the original regular grid.

In Figs.14 and 15, we show the tetrahedral mesh structures of the adaptive-grid data to which we have applied our method. The standard barycentric method is used for interpolation to define a continuous density function $\rho(\mathbf{x})$. Our method is applicable to irregular-grid data without any difficulty as long as continuous interpolation of $\rho(\mathbf{x})$ is defined.

In Fig.16 (left), we show an example of sampling the foot data of Fig.14. In the figure, generated 0.1M sample points are plotted three-dimensionally. We can clearly see that regions around the peaks of $\rho(\mathbf{x})$ are intensively sampled independently of the original irregular-grid structure. Fig.16 (right) shows the rendering result with the simple illuminant particle model. This rendering result is quite similar to Fig.10 (b), created for the regular grid data.

In Fig.17 (a), we show an example of sampling the tooth data of Fig.15 (0.1M points). This figure shows the expected sampling result properly again. Fig.17 (b) shows the rendering result with the simple illuminant particle model (5M points). This figure properly visualises the embedded structure of the tooth. Fig.17 (c) shows the rendering result with the illuminant particle model with light-attenuation effects (a=100, 5M points). This figure visualises the asperity around the root of the tooth properly.

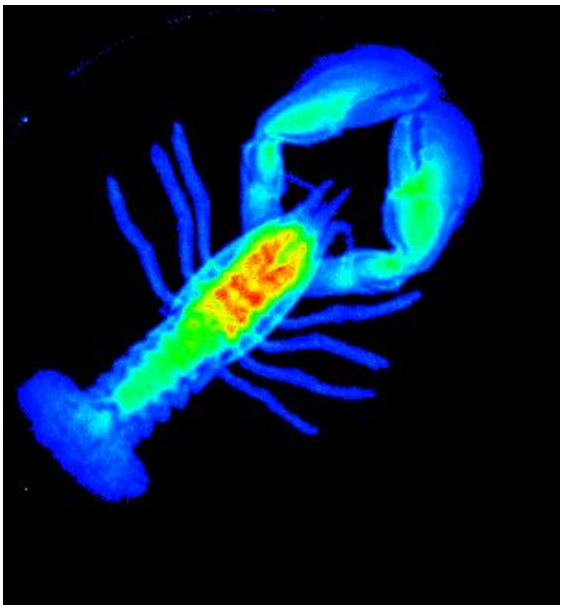


Fig. 11 Colored image of the $161 \times 321 \times 129$ lobster (regular grid) with 10M sample point and the X-ray model.

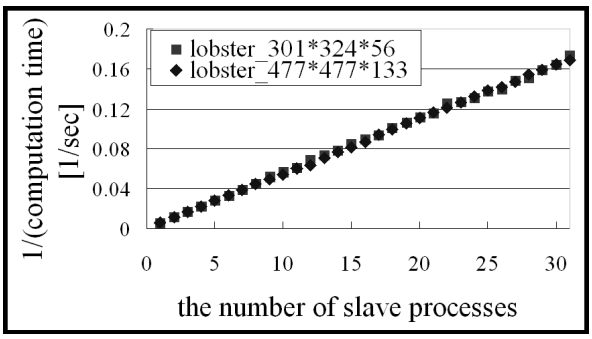
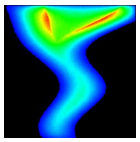


Fig. 12 Computation speeds for the increasing number of slave processes which run on Pentium Xeon 2.4 GHz processors connected with Myrinet (regular-grid data).

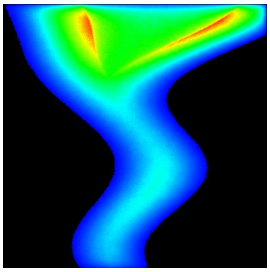
Fig.18 shows the computation speeds in the parallel processing. The computation time is measured for creating images of the foot of Fig.14 and the tooth of Fig.15 with 10M sample points. Inverses of the computation time, i.e., the computation speeds, are plotted for the increasing number of slave processes. The computation speed increases almost linearly, reflecting the high-level independency of our Metropolis sampling. The experimental result shown in Fig.18 suggests that our method is suitable for rendering large irregular-grid volume data. Since volume rendering of irregular-grid data takes a longer time than regular-grid data, it is important to accelerate the rendering speed.

The computation times required to generate images in this section is about one-order longer than those for the regular-grid data in Section 3.1. Most of the time is

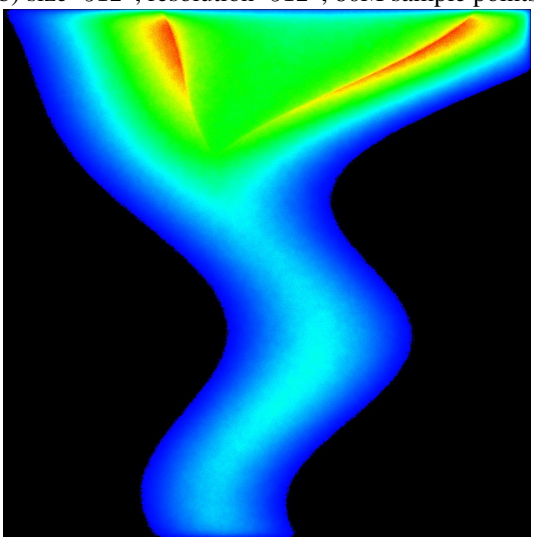
ISBN 978-3-901608-32-2



(a) size=256^3, resolution=256^2, 20M sample points.



(b) size=512^3, resolution=512^2, 80M sample points.



(c) size=1024^3, resolution=1024^2, 320M sample points.

Fig. 13 Rendering volume data with different data sizes. (The X-ray rendering model is used.) The numbers of sample points are made proportional to those of pixels. The ratio (the number of sample points)/(data size) is (a) 1.19, (b) 0.60, and (c) 0.30, respectively.

spent for evaluating $\rho(\mathbf{x})$ by investigating a tree structure to search a tetrahedral cell including a given $\mathbf{x}$, which is not an essential part of our Metropolis sampling. Acceleration of searching tetrahedral cells is not a focus of this paper. But it is, of course, an important issue for efficient rendering of irregular-grid data. It is a common target issue for every rendering method of irregular-grid data.

4 Conclusions

We have proposed a method of sampling regular/irregular-grid volume data for their visualisation. The method is based on the Metropolis algorithm which realizes importance sampling of regions of interest. Namely, the generated sample points are distributed according to a scalar field $\rho(\mathbf{x})$ that is defined by given volume data and a user-specified transfer function.

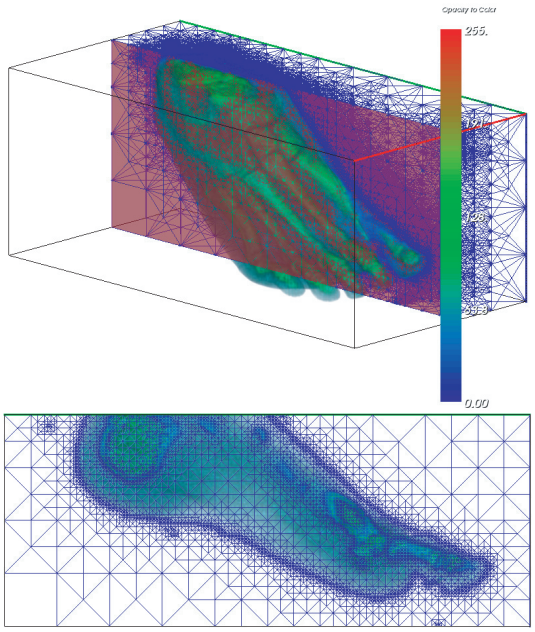


Fig. 14 Mesh structure of the adaptive grid (foot).

The sample-point distribution is independent of a grid structure of given volume data, and so our method is applicable to irregular grids as well as regular grids. We have demonstrated the effectiveness of our method by applying it to both regular cubic grids and irregular tetrahedral grids with adaptive cell sizes.

The high-density sample points generated by our method are suitable for rendering. We have successfully applied the generated sample points to the three rendering models, i.e., the X-ray model, the simple illuminant particle model, and the illuminant particle model with light-attenuation effects.

We have also demonstrated that our method is suitable for parallel processing by realizing computation speeds almost proportional to the number of processors. (Our Metropolis sampling can be accelerated by using GPU. Our preliminary results show that the sampling speed becomes a few or several times faster. Details will be reported in our future paper.)

The grid-independency and the efficiency in the parallel processing mean that our method is suitable for visualizing large-scale volume data. The former means that the required number of sample points is proportional to the number of 2D pixels, not the number of 3D voxels. The latter means that our method can be easily accelerated on the multiple-CPU and/or GPU platforms. Our future work will be done to utilize such virtues of the method for large-scale medical volume data.

The authors wish to thank Prof. Hiromi T. Tanaka, Prof. Ichiro Ohba, Yasufumi Takama, Susumu Nakata, Masafumi Oka, Akihiro Shibata, and Akinori Kimura for their valuable comments and warm encouragement.

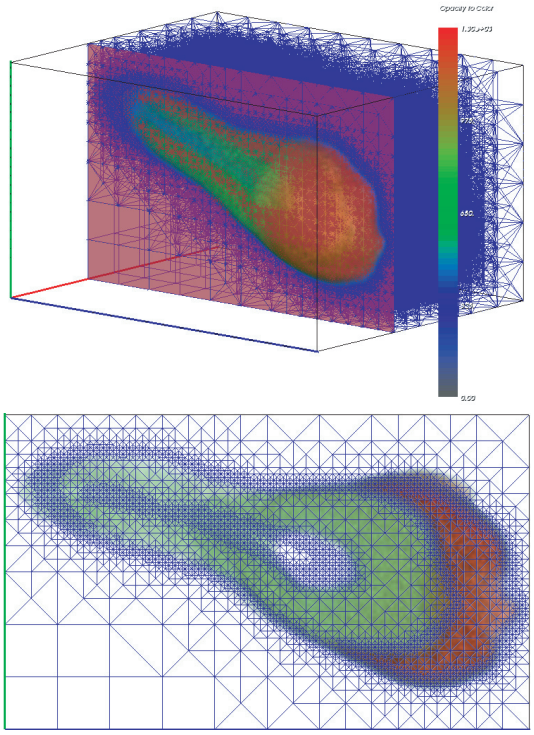


Fig. 15 Mesh structure of the adaptive grid (tooth).

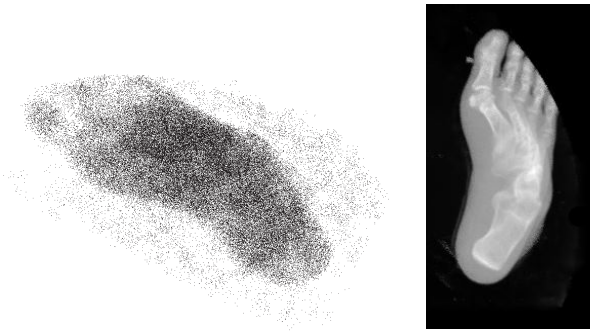


Fig. 16 Sampling and rendering results for the adaptive-grid data of Fig.14 (foot). The original volume data are directly interpolated with the barycentric method. Left: Generated sample points. Right: rendering with the simple illuminant particle model (10M points).

5 References

[1] L. Westover. Footprint Evaluation for Volume Rendering. In *Proceedings of SIGGRAPH '97*: 367–376, 1990.

[2] N. Max, P. Hanrahan and R. Crawfis. Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions. *Computer Graphics*, 24(5): 27–33, 1990.

[3] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. In *Proceedings of the 1990 Workshop on Volume Visualization*: 63–69, 1990.

[4] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics & Applications*, 8(3): 29–37, 1988.

[5] R. Drebin, L. Carpenter, P. Hanrahan. Volume Rendering. *Proceedings of SIGGRAPH '88*: 65–74, 1988.

[6] J. Wilhelms. Pursuing Interactive Visualization of Irregular Grids. *The Visual Computer*, 9: 450–458,1993.

[7] K. Koyamada and T. Miyazawa. Volume Rendering Method for Finite Element Method Results — Air-Flow Visualization in a Clean Room —. *Trans. IPS. Japan*, 32(5): 560–569, 1991.

[8] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chem. Physics*, 21: 1087–1092, 1953.

[9] H. T. Tanaka, Y. Takama and H. Wakabayashi. Accuracy-Based Sampling and Reconstruction with Adaptive Grid for Parallel Hierarchical Tetrahedrization. In *IEEE Volume Graphics 2003*: 79–86, 2003.

[10] T. Malzbender. Fourier Volume Rendering. *ACM Transactions on Graphics*, 12(3): 233–250, 1993.

[11] B. Csébfalvi and L. Szirmay-Kalos. Monte Carlo Volume Rendering. In *Proceedings of IEEE Visualization 2003*: 449–456, 2003.

[12] A. Keller and W. Heidrich. Interleaved Sampling. In *Proceedings of EUROGRAPHICS Workshop on Rendering 1998*: 269–276, 1998.

[13] N. Sakamoto and K. Koyamada. Particle Generation from User-Specified Transfer Function for Point-Based Volume Rendering. In *IEEE Visualization Proceedings Compendium*: 125-126, 2005.

[14] K. Koyamada. Volume Visualization for the Unstructured Grid Data. In *SPIE Symposium on Electronic Imaging Conference Proceedings*, 1259: 14–25, 1990.

[15] Yuan Zhou and Michael Garland. Interactive Point-Based Rendering of Higher-Order Tetrahedral Data. In *Proceedings of IEEE Visualization 2006*: 1229-1236, 2006.

[16] Matt Pharr and Randima Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*, Addison-Wesley, 2005.

ISBN 978-3-901608-38-2

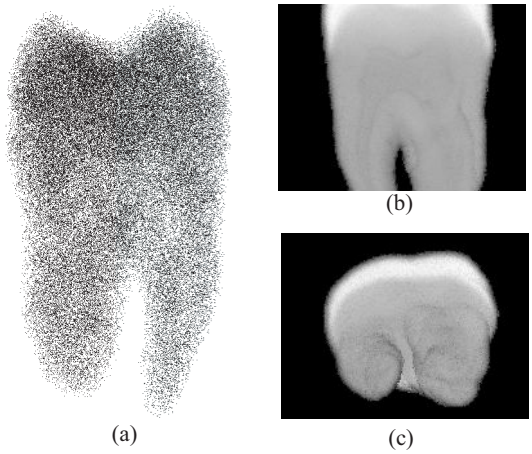


Fig. 17 Sampling and rendering results for the adaptive-grid data of Fig.15 (tooth). A transfer function similar to Fig.5 is used. (a) Generated sample points. (b) Rendering with the simple illuminant particle model (5M points) (c) Rendering with the illuminant particle model with light-attenuation effects (a=100, 5M points).

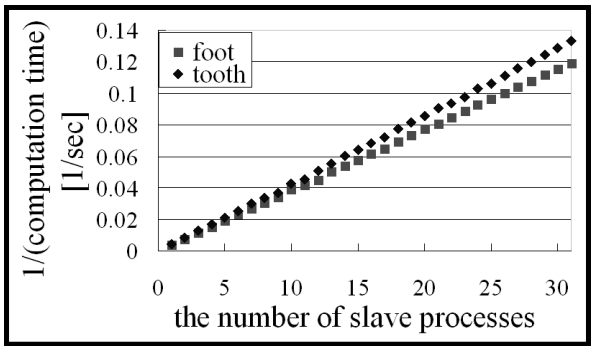


Fig. 18 Computation speeds for the increasing number of slave processes which run on Pentium Xeon 2.4 GHz processors connected with Myrinet (irregular-grid data).

