Vol. 41 No. SIG 5(HPS 1) IPSJ Transactions on High Performance Computing Systems Aug. 2000

Regular Paper

A Pre-attributed Resampling Algorithm for Controlled-Precision Volume Ray-Casting

KENTARO SANO,† HIROAKI KOBAYASHI† and TADAO NAKAMURA†

Accurate volume rendering is essential for some visualization applications, e.g., medical imaging. However, the computationally expensive feature of conventional volume rendering algorithms for high-quality image generation has restricted their practical use. In this paper, we propose a pre-attributed resampling algorithm that accomplishes controlled-precision volume ray-casting at low computational costs. This algorithm changes resampling intervals based on numerical errors of the volume rendering integral so that the number of resampling points becomes minimum for a given error bound. Besides, to reduce computational costs for resampling, a simple interpolation method is applied to resampling points in regions where intensities and opacities are constant. To suppress the overhead of precision control, information on the numerical errors and the constant regions is obtained for each voxel in pre-processing, and then related to volume data as voxel attributes. The experimental results demonstrate that the proposed algorithm outperforms conventional ray-casting algorithms without precision control for accurate visualization in terms of accuracy/processing-time performance.

1. Introduction

Volume rendering is a useful and important tool to directly visualize 3-dimensional numerical data (volume) acquired through scientific computing or measurement, such as computational fluid dynamics and medical imaging. Most of recent researches in volume rendering pay attention to rendering speedups by means of efficient computation at the expense of accuracy of visualization. However, accurate visualization is essential for some applications, e.g., medical diagnoses with critical decisions, that require exact observation of volume data.

Volume is generally defined as a set of values given at 3-dimensional grid points called voxels. These voxel values and their derivatives are mapped to intensities and opacities using shading and classification transfer functions for visualization. In practical volume ray-casting, intensities and opacities are calculated at resampling points along a ray by interpolating voxel intensities and opacities pre-computed at given voxel locations because of low computational complexity [1]. However, pre-aliasing [2] errors occur in voxel intensities and opacities when non-linearity of shading and classification transfer functions leads to a large amount of energy above the Nyquist rate of a voxel resolution. Interpolation of voxel intensities and opacities with errors cannot provide correct intensities and opacities, even if the interpolation is almost ideal. Therefore, to compute accurate intensities and opacities for arbitrary transfer functions, voxel values and their derivatives should be reconstructed at resampling points, and then shaded and classified.

A voxel value and its derivative at an arbitrary position can be reconstructed by using 3-dimensional convolution of voxel values and a reconstruction filter. Poor reconstruction filters such as a trilinear interpolation filter cause noticeable errors due to post-aliasing [2)~4)]. Although the ideal filter avoids post-aliasing, it needs a tremendous computation for reconstruction because of its infinite support. Therefore, practical filters that well approximate the ideal one with a small support have been desired. A number of researchers have investigated piecewise cubic filters based on various metrics to design more efficient reconstruction filters for volume rendering [5)~7)]. Möller, et al.[5] evaluated numerical accuracy of the cubic BC-spline filter in the spatial domain using a Taylor series expansion. Their evaluation concluded that the Catmull-Rom spline is the most optimal reconstruction filter in the class of cubic BC-spline filters [6], in terms of asymptotic error behavior. They also designed a good derivative filter only with 4 filter weights: the C^1-2EF derivative filter [7].

The higher order filters provide more accurate interpolation at higher computational

† Graduate School of Information Sciences, Tohoku University

costs. Sánchez, et al.[8] proposed an adaptive interpolation method that achieves high image quality while reducing the computation time. In this method, voxels are classified into three kinds of zones: constant, linear, and non-linear. Accurate, but expensive reconstruction filters are applied only to non-linear zones. Although this technique effectively works for efficient interpolation, numerical errors in solving the volume rendering integral along a ray are not taken into account.

In most of volume rendering algorithms, the volume rendering integral to calculate an intensity of each pixel is solved by numerical quadrature along a ray. The interval of the integral is subdivided into several subintervals, where intensities and opacities are resampled. By shortening a resampling interval, an error of numerical quadrature can be reduced, while leading to higher resampling costs. However, the resampling intervals necessary to ensure a given accuracy should be different, since the errors also depend on the derivatives of the integrand. Accordingly, the number of redundant resampling operations for a given accuracy can be reduced by computing a proper length of a subinterval.

Novins[9] presented volume ray-casting algorithms to compute a subinterval length locally adequate for a given error bound by evaluating the maximum error of the numerical quadrature in each domain. The maximum error is computed with upper bounds on the derivatives of the integrand. His algorithms can achieve controlled-precision rendering at low computational costs by reducing resampling operations. In his algorithms, the derivatives are bounded under the assumption that intensity and opacity functions in the integrand are piecewisely polynomial due to trilinear or higher order polynomial interpolation of voxel intensities and opacities.

However, even if interpolation is polynomial, this assumption is not suitable for intensities and opacities calculated from interpolated voxel values and their derivatives because shading and classification transfer functions commonly have non-linearity. His algorithms do not work well on non-polynomial intensity and opacity functions. In addition, since bounds on the derivatives are computed during a rendering stage, volume ray-casting based on his algorithms has a large overhead. Moreover, Novins's algorithms do not take account of efficient interpolation by choosing a proper reconstruction method for each domain.

In this paper, we propose another volume ray-casting algorithm to control precision by changing a reconstruction method and a subinterval length. The proposed algorithm interpolates pre-computed voxel intensities and opacities in the regions where intensity and opacity are almost constant. Elsewhere, shading and classification operations are applied to an interpolated voxel value and its derivative. This algorithm calculates an adequate length of a subinterval by evaluating the maximum error of the numerical quadrature with upper bounds on a derivative of the integrand. Since intensity and opacity functions calculated from interpolated voxel values and their derivatives are not always piecewisely polynomial, this algorithm bounds the derivative without Novins's assumption.

In addition, to suppress the overhead of precision control, information on the numerical errors and the constant regions is obtained for each voxel in pre-processing, and then appended to volume data as voxel attributes. During the ray-casting stage, a reconstruction method and a subinterval length are simply obtained by looking up the voxel attributes with a small overhead. Since voxel attributes do not have to be re-computed until resampling parameters except viewing ones change, computational costs for pre-processing are not critical to render a sequence of frames for a walk-through animation.

This paper is organized as follows. Section 2 provides basic concepts of the proposed algorithm based on formulated error factors of volume ray-casting. Section 3 presents techniques to estimate local errors in volume ray-casting. Section 4 describes an implementation of the proposed algorithm. Section 5 shows experimental results and discussion of the algorithm in terms of accuracy/processing-time performance. Section 6 gives conclusions and future directions.

2. Controlled-Precision Volume Ray-Casting

2.1 Volume Ray-Casting

Volume ray-casting is a basic and representative volume rendering algorithm that calculates pixel intensities by numerically solving the volume rendering integral along a ray emanated from a viewing point through the pixels[1),10)~12)]. The volume rendering integral is given by

$$I(t_a, t_b) = \int_{t_a}^{t_b} \tau(t) e^{-\int_{t_a}^{t} \sigma(s)ds} dt$$

where $\tau(t)$ and $\sigma(s)$ are an intensity function and an opacity function, respectively. Since no general solution to the volume rendering integral is known for arbitrary intensity and opacity functions, numerical quadrature is applied to it. Due to simplicity, most volume ray-casting algorithms use the open composite midpoint rule [13] that divides the interval into several subintervals, in each of which intensities and opacities are assumed to be constant with their midpoint values. The integration results of subintervals are composited to a pixel with the "over" operator [14] in the front to back order.

Two ways exist for calculation of intensities and opacities at the midpoints. The first one calculates an intensity and an opacity from a reconstructed voxel value and its derivative. We refer to this method as R(SC) because of shading and classification after reconstruction. The second one calculates an intensity and an opacity by interpolating voxel intensities and opacities. In this paper, this method is called (SC)R because of reconstruction after shading and classification. In the R(SC), shading and classification operations are necessary for each midpoint though only reconstruction process impacts accuracy of intensities and opacities. On the other hand, computational costs of the (SC)R are less than those of the R(SC), because shading and classification are removed from rendering stages. However, the (SC)R may cause noticeable errors in reconstructed intensities and opacities because of insufficient voxel resolution for arbitrary intensity and opacity functions.

2.2 Numerical Errors of Volume Ray-Casting

2.2.1 Errors of Reconstructed Intensity and Opacity

Here, we formulate intensities and opacities reconstructed by using the R(SC) and the (SC)R to explain the reason why the (SC)R cannot always calculate accurate intensities and opacities.

For simplicity but without the loss of generality, we consider 1-dimensional reconstruction. Let voxel values be $v(hi)$, where $i \in \mathbf{Z}$ and h is an interval between adjacent voxels. Opacity $\sigma_{R(SC)}(x)$, which is calculated by using the R(SC) at point x, is expressed as

$$\sigma_{R(SC)}(x) = C\left(\sum_i K(x - hi)v(hi)\right)$$

where $K(x)$ is a 1-dimensional reconstruction filter function, and $C(v)$ is a transfer function that maps voxel values to opacities. $K(x)$ is normalized so that $\sum_i K(x - hi) = 1$.

On the other hand, the (SC)R calculates an opacity at point x as

$$\sigma_{(SC)R}(x) = \sum_i K(x - hi)C(v(hi)).$$

It is obvious that $\sigma_{R(SC)}(x) \neq \sigma_{(SC)R}(x)$ when $C(v)$ does not have linearity. However, $\sigma_{R(SC)}(x)$ is equal to $\sigma_{(SC)R}(x)$ for arbitrary transfer functions when $v(x)$ is constant. This indicates that the (SC)R reconstructs accurate intensities and opacities at lower computational costs in the region where they are constant. We refer to such a region as a constant region.

2.2.2 Errors in Numerical Quadrature of Volume Rendering Integral

In this section, we formulate errors caused by the numerical quadrature of the volume rendering integral. For convenience, we define $f(t)$ to be the integrand for interval $[-t_s/2, t_s/2]$ as

$$f(t) = \tau(t) e^{-\int_{-t_s/2}^{t} \sigma(s)ds}$$

where t_s is the length of the interval. In the open composite midpoint rule, the interval is subdivided into n subintervals $[t_i, t_{i+1}]$, where the integrand is assumed to be constant with its midpoint value. Integral in each subinterval is expressed as

$$\int_{t_i}^{t_{i+1}} f(t)dt = hf(m_i) + R_i(h)$$

where $t_i = -t_s/2 + hi$, $h = \frac{t_s}{n}$, and a midpoint of subinterval $[t_i, t_{i+1}]$ is $m_i = \frac{t_i + t_{i+1}}{2}$. Remainder $R_i(h)$ is an error of the midpoint rule in $[t_i, t_{i+1}]$.

The error in the interval, $E(-t_s/2, t_s/2)$, is given by the sum of $R_i(h)$. For $\xi \in [-t_s/2, t_s/2]$, the sum can be transformed into a term of $f''(\xi)$ by using the mean-value theorem:

$$E(-t_s/2, t_s/2) = \sum_{i=0}^{n-1} R_i(h) = \frac{t_s}{24} h^2 f''(\xi).$$

When t_s is small enough for $f''(\xi)$ to approximate $f''(0)$, we get

$$E(-t_s/2, t_s/2) \simeq \frac{t_s}{24} h^2 f''(0). \quad (1)$$

This equation states that the error of the numerical integration can approximately be esti-

mated by using the second order differential coefficient of the integrand at the midpoint of the interval for small t_s.

Since $f''(t)$ is expressed with $\tau(t)$, $\sigma(t)$ and their derivatives, an absolute error bound of the numerical integration is given as

$$|E(-t_s/2, t_s/2)| \leq \frac{t_s h^2}{24}\{|\tau''(0)| + 2|\tau'(0)||\sigma(0)| + |\tau(0)|(\sigma^2(0) + |\sigma'(0)|)\}. \quad (2)$$

$\tau(t)$ and $\sigma(t)$, which are 1-dimensional functions along a ray, can be expressed as $\tau(\boldsymbol{c}+\boldsymbol{d}t)$ and $\sigma(\boldsymbol{c}+\boldsymbol{d}t)$ respectively, where unit vector $\boldsymbol{d} = (d_x, d_y, d_z)$ is a direction of the ray passing through the midpoint of the interval, $\boldsymbol{c} = (x_c, y_c, z_c)$. Using ${d_x}^2 + {d_y}^2 + {d_z}^2 = 1$, $|\tau''(0)|$, $|\tau'(0)|$ and $|\sigma'(0)|$ are bounded as follows:

$$\begin{aligned}|\tau''(0)| &= |\tau_{xx}(\boldsymbol{c}){d_x}^2 + \tau_{yy}(\boldsymbol{c}){d_y}^2 \\ &\quad + \tau_{zz}(\boldsymbol{c}){d_z}^2 + 2(\tau_{xy}(\boldsymbol{c})d_x d_y \\ &\quad + \tau_{yz}(\boldsymbol{c})d_y d_z + \tau_{zx}(\boldsymbol{c})d_z d_x)| \\ &\leq |\tau_{xx}(\boldsymbol{c})| + |\tau_{yy}(\boldsymbol{c})| + |\tau_{zz}(\boldsymbol{c})| \\ &\quad + |\tau_{xy}(\boldsymbol{c})| + |\tau_{yz}(\boldsymbol{c})| + |\tau_{zx}(\boldsymbol{c})|, \end{aligned} \quad (3)$$

$$\begin{aligned}|\tau'(0)| &= |\tau_x(\boldsymbol{c})d_x + \tau_y(\boldsymbol{c})d_y + \tau_z(\boldsymbol{c})d_z| \\ &\leq |\tau_x(\boldsymbol{c})| + |\tau_y(\boldsymbol{c})| + |\tau_z(\boldsymbol{c})| \end{aligned} \quad (4)$$

and

$$\begin{aligned}|\sigma'(0)| &= |\sigma_x(\boldsymbol{c})d_x + \sigma_y(\boldsymbol{c})d_y + \sigma_z(\boldsymbol{c})d_z| \\ &\leq |\sigma_x(\boldsymbol{c})| + |\sigma_y(\boldsymbol{c})| + |\sigma_z(\boldsymbol{c})|. \end{aligned} \quad (5)$$

Equations (2)–(5) give an upper bound on the absolute errors of the numerical integration for any ray direction as:

$$|E(-t_s/2, t_s/2)| \leq \frac{t_s h^2}{24} D(\boldsymbol{c}) \quad (6)$$

where

$$\begin{aligned}D(\boldsymbol{c}) \equiv\ & |\tau_{xx}(\boldsymbol{c})| + |\tau_{yy}(\boldsymbol{c})| + |\tau_{zz}(\boldsymbol{c})| \\ & + |\tau_{xy}(\boldsymbol{c})| + |\tau_{yz}(\boldsymbol{c})| + |\tau_{zx}(\boldsymbol{c})| \\ & + 2(|\tau_x(\boldsymbol{c})| + |\tau_y(\boldsymbol{c})| + |\tau_z(\boldsymbol{c})|)|\sigma(\boldsymbol{c})| + \\ & |\tau(\boldsymbol{c})|(\sigma^2(\boldsymbol{c}) + |\sigma_x(\boldsymbol{c})| + |\sigma_y(\boldsymbol{c})| + |\sigma_z(\boldsymbol{c})|). \end{aligned}$$

Although this is a coarse bound, we can evaluate the maximum of the absolute errors for the volume rendering integral in the $\boldsymbol{c}$-neighborhood by computing $D(\boldsymbol{c})$ from $\tau(\boldsymbol{c})$, $\sigma(\boldsymbol{c})$, and their partial differential coefficients at $\boldsymbol{c}$.

2.3 Precision Control

As described in the previous subsections, the reconstruction methods and the lengths of subintervals to accomplish a given error bound at low computational costs are influenced by the local behavior of the intensity and opacity functions, and therefore should differ for each domain of the volume rendering integral. We propose an algorithm that exploits this property by resampling with an adequate reconstruction method and a subinterval for precise visualization at low computational costs.

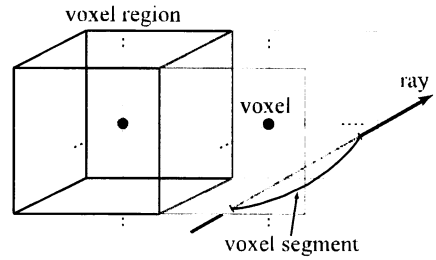


Fig. 1 Voxel region and voxel segment.

We define a voxel region as the inside of a rectangular solid centered at a voxel, which has the same size as a rectangular solid with eight neighboring voxels as vertices. We also define a voxel segment as the longest segment along a ray that is included in a voxel region. Examples of a voxel region and a voxel segment are depicted in **Fig. 1**. In the proposed algorithm, the whole interval along a ray is divided into several voxel segments, to which the numerical quadrature is applied. In each voxel segment, a reconstruction method is chosen between the R(SC) and the (SC)R. The (SC)R is used in constant voxel regions for lower reconstruction costs, while the R(SC) is used elsewhere for precise reconstruction of intensities and opacities. Besides, the length of a subinterval is determined so that the maximum error of the numerical quadrature is less than a given error bound at the lowest computational costs for each voxel segment. We refer to these processes as precision control.

To suppress the overhead of precision control, information on the constant regions and the numerical errors is obtained for each voxel region in pre-processing, and then related to volume data as voxel attributes. During a rendering stage, precision control is simply performed by looking up the voxel attributes that a ray passes through. A method for voxel attributes generation is described in Section 4.1.

3. Evaluation of Intensity and Opacity Functions

3.1 Detection of Constant Region

The order of a polynomial function can locally be obtained by evaluating wavelet coefficients of a wavelet basis with vanishing moments. For wavelet basis $\Psi(x)$ with M vanishing moments,

$$\langle \Psi(x), x^n \rangle = 0, \quad 0 \le n \le M-1. \qquad (7)$$

This states that all wavelet coefficients are 0 when the wavelet transform with a basis of M vanishing moments is applied to polynomial functions of order $M-1$. Therefore, constant regions can be detected by evaluating wavelet coefficients of Haar basis with one vanishing moment. To detect constant regions of intensity and opacity functions, their 3-dimensional wavelet transform is performed with Haar basis in the same way as Refs. 15) and 16).

3.2 Estimation of Differential Coefficients

The maximum of the absolute errors for the volume rendering integral in each voxel region can be computed based on Eq. (6). In Eq. (6), the first and second order partial differential coefficients of intensity and opacity functions have to be computed at a voxel location. We approximately estimate these partial differential coefficients by using the Haar wavelet coefficients of intensities and opacities computed at a resolution four times higher than the voxel resolution.

Here, we explain the estimation in 2D for simplicity, which will be extended to the 3-dimensional representation. In the 2-dimensional case, each voxel region includes 4^2 intensities and opacities at a resolution four times higher than that of voxels. As shown in **Fig. 2**, let intensities in the voxel region (a, b) be

$$c_{l,m} = \tau\left(a + \frac{2l-3}{4}h, b + \frac{2m-3}{4}h\right) \quad l, m \in \{0, 1, 2, 3\}$$

where $2h$ is a distance between adjacent voxels. Haar wavelet coefficients, which are denoted by $d^{01}_{l,m}$, $d^{10}_{l,m}$, and $d^{11}_{l,m}$, can be computed from $c_{l,m}$ by using a decomposing operation of the filter banks [17] as

$$d^{01}_{l,m} = (c_{2l+1,2m} - c_{2l,2m} + c_{2l+1,2m+1} - c_{2l,2m+1})/4 \qquad (8)$$

where $l, m \in \{0, 1\}$. For $\tau(x, y)$ that is assumed to be a real analytic function, a Taylor series expansion of $\tau(x, y)$ in the neighborhood of (a, b) and Eq. (8) give the following approximations:

$$\sum_{l,m} \frac{d^{10}_{l,m}}{h} = \frac{\partial \tau}{\partial x}(a, b) + O(h^2)$$

and

$$\sum_{m} \frac{d^{10}_{1,m} - d^{10}_{0,m}}{h^2} = \frac{\partial^2 \tau}{\partial x^2}(a, b) + O(h^2).$$

For $d^{11}_{l,m}$, we can similarly derive

$$\sum_{l,m} \frac{4d^{11}_{l,m}}{h^2} = \frac{\partial^2 \tau}{\partial x \partial y}(a, b) + O(h^2).$$

In the 3-dimensional case, similar derivations for $\tau(x, y, z)$ give the following approximations:

$$\sum_{l,m,n} \frac{d^{100}_{l,m,n}}{h} = \frac{\partial \tau}{\partial x}(a, b, c) + O(h^2),$$

$$\sum_{m,n} \frac{d^{100}_{1,m,n} - d^{100}_{0,m,n}}{h^2} = \frac{\partial^2 \tau}{\partial x^2}(a, b, c) + O(h^2)$$

and

$$\sum_{l,m,n} \frac{4d^{110}_{l,m,n}}{h^2} = \frac{\partial^2 \tau}{\partial x \partial y}(a, b, c) + O(h^2).$$

In addition, the d-symmetry indicates that the other partial differential coefficients of $\tau(x, y, z)$ at (a, b, c) can also be approximated by using $d^{010}_{l,m,n}$, $d^{001}_{l,m,n}$, $d^{011}_{l,m,n}$ and $d^{101}_{l,m,n}$. The first order partial differential coefficients of $\sigma(x, y, z)$ are also approximately estimated in the same way as $\tau(x, y, z)$.

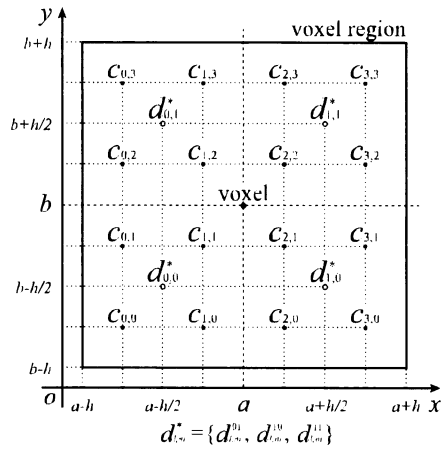


Fig. 2 Intensities at a higher resolution and their wavelet coefficients in a voxel region.

4. Implementation

4.1 Voxel Attributes

Voxel attributes are computed in pre-processing as follows. First, intensities and opacities are calculated at a resolution four times higher than that of voxels by using the R(SC). In the intensity and opacity calculation, voxel values and their derivatives resampled with proper reconstruction filters, such as cubic spline filters, are used for shading and classification in the same way as Levoy's ray-casting algorithm [1]. Then, Haar wavelet coefficients $d^*_{l,m,n}$ are computed for each of the intensities and opacities. Voxel attributes are calculated from these wavelet coefficients. After all of attributes are calculated, the wavelet coefficients, the reconstructed intensities and opacities are discarded to save memory space.

A voxel attribute consists of a reconstruction method flag (called RMF) and a resampling resolution level (called RRL). An RMF is expressed in one bit to choose a reconstruction method between the (SC)R and the R(SC). The (SC)R should be chosen in the regions where intensities and opacities are approximated to be constant in order to reduce reconstruction costs, while the R(SC) should be used elsewhere for accurate reconstruction. Since constant regions can be detected based on the vanishing moments of Haar wavelet as described in Section 3.1, the RMF of a voxel is set to the (SC)R when all the wavelet coefficients of intensities and opacities are 0 in its voxel region. Otherwise, the RMF is set to the R(SC).

An RRL indicates a degree of the maximum absolute error of the numerical quadrature for a certain length of subintervals. A floating point representation is suitable for the RRL representation to precisely determine the proper length of each subinterval. However, the small size of an RRL is desired to avoid too much memory consumption because of a large number of voxels. Therefore, an RRL is expressed in an n-bit unsigned integer that indicates a class of the maximum absolute error of the numerical quadrature. Consequently, an RRL is set to one of the numbers: $0, 1, 2, \cdots, 2^n - 1$. An appropriate bit length of an RRL will be examined in Section 5.

An RRL of each voxel is set to L when a given error bound E_b satisfies the following condition in the voxel region:

$$\frac{D(\boldsymbol{c})}{24}\frac{1}{2^L} \le E_b < \frac{D(\boldsymbol{c})}{24}\frac{1}{2^{L-1}} \qquad (9)$$

for $0 < L < 2^n - 1$ where $\boldsymbol{c}$ is the voxel location. Otherwise, an RRL is set to 0 or $(2^n - 1)$ as follows:

$$RRL = \begin{cases} 0 & \frac{D(\boldsymbol{c})}{24 \cdot 2^0} \le E_b \\ 2^n - 1 & E_b < \frac{D(\boldsymbol{c})}{24 \cdot 2^{2^n - 2}} \end{cases}. \qquad (10)$$

$\frac{D(\boldsymbol{c})}{24 \cdot 2^L}$ in Eqs. (9) and (10) denotes the maximum absolute error in the voxel segment with a length of 1 for 2^L subintervals. $D(\boldsymbol{c})$ is approximately computed by estimating the partial differential coefficients of intensities and opacities from their wavelet coefficients as shown in Section 3.2.

4.2 Adaptive Resampling Based on Voxel Attributes

The proposed algorithm calculates a pixel intensity by compositing contributions of voxel segments along a ray in the front to back order. Based on an RRL, each voxel segment is subdivided into subintervals so that the maximum absolute error of the numerical quadrature in the segment is less than a given error bound. Suppose that a voxel segment with a length of t_s is subdivided into n_s subintervals. For the voxel centered at $\boldsymbol{c}$, Inequality (6) gives the maximum error of the numerical quadrature, $E_{\max} = \frac{t_s}{24} h^2 D(\boldsymbol{c})$, where $h = \frac{t_s}{n_s}$. n_s can be computed for RRL $\le 2^n - 1$ so that $E_{\max}$ is less than E_b as follows:

$$\frac{t_s}{24} h^2 D(\boldsymbol{c}) \le \frac{D(\boldsymbol{c})}{24}\frac{1}{2^{\mathrm{RRL}}}. \qquad (11)$$

Since n_s should be the smallest natural number that satisfies Inequality (11), n_s is calculated as

$$n_s = \left\lceil \sqrt{t_s^3 2^{\mathrm{RRL}}} \right\rceil. \qquad (12)$$

$\lceil x \rceil$ denotes the ceiling function. In the case of RRL=$2^n - 1$, Eq. (12) gives the largest number of subintervals though $E_{\max}$ is not always less than E_b.

An intensity and an opacity at a midpoint of each subinterval are calculated based on the method specified by the RMF of the voxel region including the subinterval. Each voxel also stores an intensity and an opacity computed at the voxel position in pre-processing. If an RMF specifies the (SC)R, an intensity and an opacity of the midpoint are calculated by interpolating pre-computed voxel intensities and opacities. If the R(SC) is specified, voxel values and their derivatives are resampled at the midpoint, and then transferred to intensities and opacities by

shading and classification.

5. Experiments

In this section, we evaluate performance of the proposed controlled-precision volume ray-casting algorithm in comparison with the conventional volume ray-casting algorithms based on either the (SC)R or the R(SC).

5.1 Experimental Environment

All of the timing results were measured through rendering programs on an Ultra SPARC-II (360 MHz) with a main memory enough to store each volume data set. Since the accuracy of the numerical quadrature is improved by shortening the constant length of a subinterval as stated in Eq. (1), we measured the accuracy of the conventional volume ray-casting algorithms by varying a constant length of a subinterval. For the proposed ray-casting algorithm, error bound E_b for an RRL was varied from 0.0 to 10.0. For the R(SC), the Catmull-Rom spline filter and the C^1-2EF derivative filter are used to reconstruct a voxel value and its derivative. For the (SC)R, the trilinear interpolation filter is used to interpolate voxel intensities and opacities. These reconstruction filters are similarly used for the conventional (SC)R and R(SC) ray-casting algorithms. Although volume data sets are represented in 8-bit integers, rendering computation is performed with double precision operations.

For experiments, we generated 8-bit gray scale images with a size of 256^2 pixels from two data sets: **Marschner** and **Jaw**. Marschner was obtained by sampling an analytical function given by Marschner, et al.[2], which has often been used to evaluate accuracy of reconstruction filters and rendering algorithms for volume data[7),8),18),19]. The function was sampled over a volume of 32^3 samples with a depth of 8 bits. This sampling rate provides severe tests for reconstruction due to a large amount of energy near the Nyquist rate. Jaw is a part of volume data that were obtained from CT slices of a human skull. In the Jaw data set, teeth and a chin are represented by 64^3 voxels with a depth of 8 bits. Both data sets are classified by using an isovalue contour surface classification function given by Levoy[1].

The accuracy of rendered images were evaluated using the root mean square (RMS) of differences in pixel intensity between a reference image and the rendered images. When N pixels are contained in one image, an RMS is defined as

$$RMS = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(I_i^{ref} - I_i)^2}$$

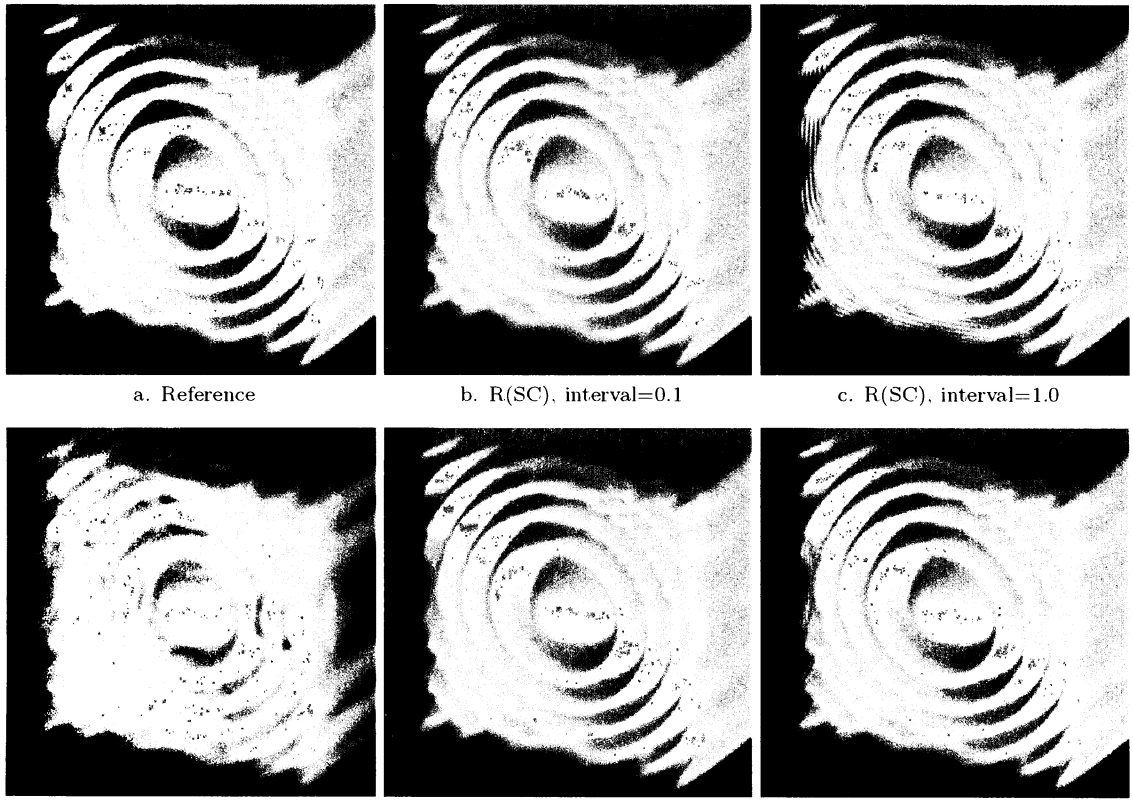
where I_i^{ref} and I_i are pixel intensities of the reference image and the rendered image, respectively. For each data set, an image rendered by using the R(SC) ray-casting algorithm with a constant interval of 0.01 was used as the reference image. The unit of an interval is a distance between adjacent voxels.

5.2 Rendered Images

Figures 3 and **4** show rendered images of Marschner and Jaw, respectively. In both figures, the top-left images are the reference images of the two data sets. Figure 3 a preserves the shape of the waves defined by the Marschner's function. This demonstrates that the spectrum of this function is almost band-limited overall the volume, and can acceptably be sampled at 32^3 points without aliasing. In Fig. 4 a, detailed shapes of teeth are distinctly rendered due to the higher order reconstruction filters and the fine resampling intervals for the numerical quadrature.

For both data sets, the top-middle and top-right images are rendered by using the R(SC) algorithm with intervals of 0.1 and 1.0. While the images rendered with an interval of 0.1 almost look the same as the reference images, striped patterns emerge in the images rendered with an interval of 1.0. Since the R(SC) algorithm calculates quasi-accurate intensities and opacities, these artifacts are caused by errors of the numerical quadrature for the volume rendering integral.

The bottom-left images in Figs. 3 and 4 are rendered by using the (SC)R algorithm with an interval of 0.1. For Marschner, the rendered image suffers from the extreme Moiré patterns that destroy the overall shapes of the waves despite a fine interval of 0.1. The Moiré patterns are caused due to errors in interpolating voxel intensities and opacities. For Jaw, intensities and opacities are reconstructed better than those of Marschner by using the (SC)R algorithm. Consequently, the overall shape of Jaw can be perceived in the rendered image as shown in Fig. 4 d. However, severe artifacts appear on the surface of the lower jaw, and especially the teeth with striped patterns and dark spots. Besides, the images as a whole are blurred in comparison with the other images.



a. Reference b. R(SC), interval=0.1 c. R(SC), interval=1.0

d. (SC)R, interval=0.1 e. Proposed, E_b=0.0 f. Proposed, E_b=10.0

Fig. 3 Rendered images of Marschner.

These artifacts demonstrate that the resolution of the voxels is not sufficient to keep the high frequency spectrum of the intensity and opacity functions.

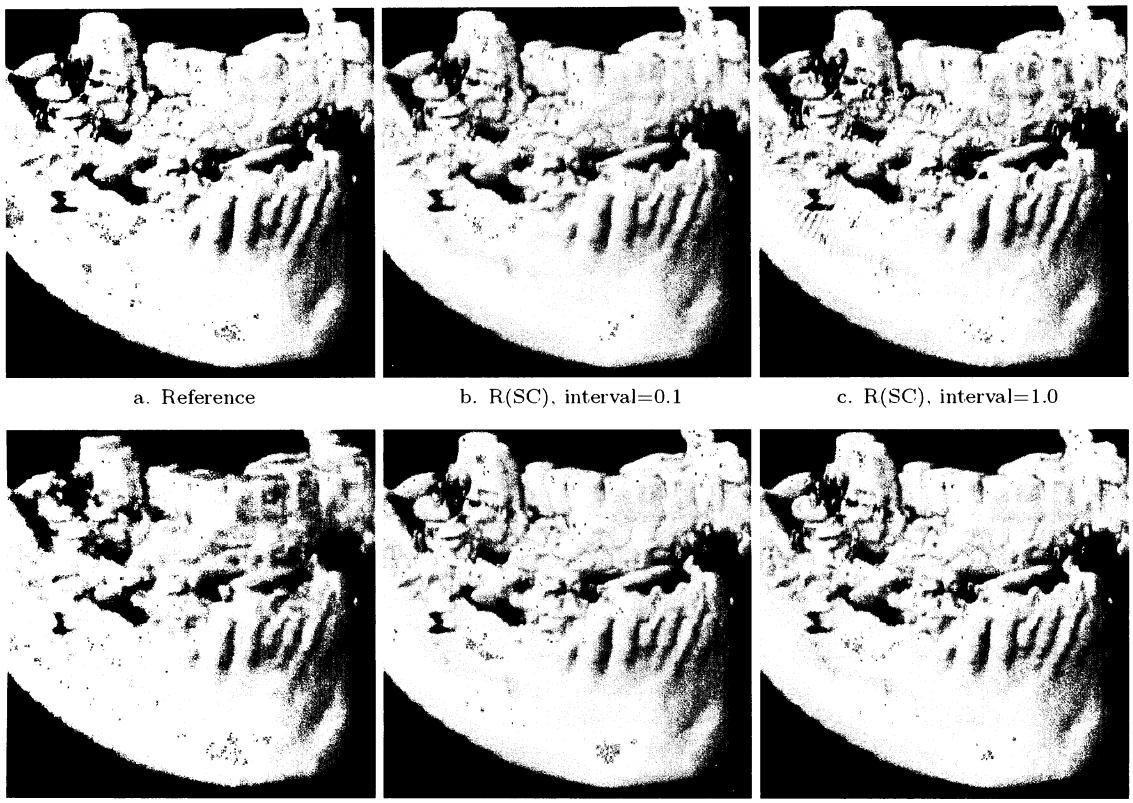
Images generated by the proposed algorithm with an RRL of 3 bits are shown in the bottom-middle and bottom-right. For both data sets, the bottom-middle images rendered with an error bound of 0.0 are not visibly different from the reference images. Besides, much better quality than the images of the (SC)R algorithm is achieved for an error bound of 10.0, even though there are slight noises overall the images. These results manifest that almost no visible blurring and errors of the numerical quadrature occur in the case of error bounds less than 10.0.

5.3 Processing Time and Accuracy

Figure 5 shows a relationship between the rendering time and the RMS of the Marschner data set for each algorithm. "E_b" and "int" in this figure indicate the error bound for the proposed algorithm and the resampling intervals for the conventional R(SC) and (SC)R algorithms, respectively.

This figure demonstrates the tradeoff between the rendering time and the RMS for the R(SC) algorithm. The shorter resampling interval provides the smaller RMS, while the rendering time becomes longer. When the resampling interval is 0.1, an RMS is 0.2 and the rendering time is over 500 sec. The rendered image with an RMS of 0.2 has no visible difference in image quality from the reference image as shown in Fig. 3. For a resampling interval of 1.0, the rendering time is reduced to 55 sec, although an RMS increases to 7.

On the other hand, the (SC)R algorithm has no tradeoff between the rendering time and the RMS. The RMS for the (SC)R algorithm does not decrease even though the rendering time is increased by the shorter resampling intervals. This is because errors in interpolated intensities and opacities are independent of the resampling intervals. Furthermore, such noticeable artifacts as the Moiré patterns result in a considerably large RMS of over 25. Therefore, the (SC)R algorithm is not suitable for accurate vi-



a. Reference b. R(SC), interval=0.1 c. R(SC), interval=1.0

d. (SC)R, interval=0.1 e. Proposed, E_b=0.0 f. Proposed, E_b=10.0

Fig. 4 Rendered images of Jaw.

sualization, although fast visualization can be achieved.

The proposed algorithm also has a tradeoff between rendering time and the RMS, which is shown as the solid line for a 3-bit RRL. The smaller error bound results in a smaller RMS and a shorter rendering time. The RMS for an error bound ranging from 0.0 to 10.0 is much less than that of the (SC)R algorithm. Especially, the RMS less than 1.0 is equivalent to the rendering quality of the reference image. Comparing timing results that achieve the same RMS, the rendering time of the proposed algorithm is always less than that of the R(SC) algorithm for any error bound as shown in Fig. 5. Especially, the rendering time is almost a half of that of the R(SC) algorithm for an error bound of 0.0. This demonstrates that precision control effectively works for accurate visualization at lower computational costs.

Figure 6 indicates that the results of the Jaw data set has almost the same tendency as those of the Marschner data set. However, the advantage of the proposed algorithm is much greater than that for Marschner since Jaw contains many redundant regions where precision control effectively works. This result demonstrates that the proposed algorithm is also effective for such a class of volume data sets in real applications.

In other experiments, the rendering time and the RMS in the case of a 1-bit or 2-bit RRL had almost the same trends as those of a 3-bit RRL except their minimum RMSs. Since the smaller number of RRL bits limits a choice of shorter subintervals, the minimum RMS for an error bound of 0.0 increases as the number of RRL bits is reduced. Therefore, the number of RRL bits should be given based on required accuracy. We empirically found that at least 3 bits are necessary for precision control to achieve rendering quality visibly equivalent to the reference image.

5.4 Memory Requirement

Table 1 shows memory requirements of each algorithm in bytes/voxel. Each element of the voxel attributes consists of a 1-bit RMF and a 3-bit RRL. For the proposed algorithm, each

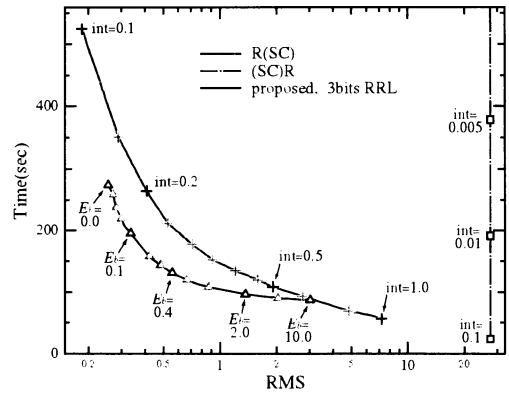


Fig. 5 RMS vs. Rendering Time for Marschner.

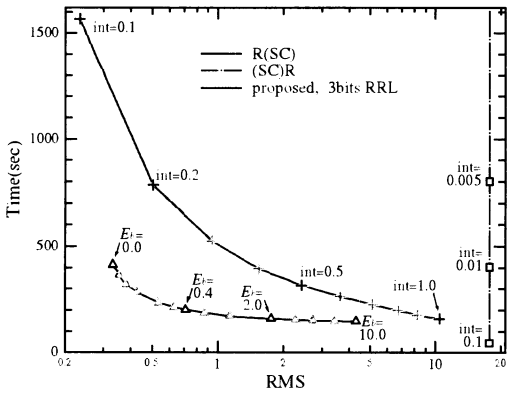


Fig. 6 RMS vs. Rendering Time for Jaw.

voxel stores all of a voxel value, an intensity and an opacity despite its RMF because reconstruction in a voxel region concerns the adjacent voxels. For example, trilinear interpolation in a voxel region with an RMF of the (SC)R requires intensities and opacities of voxels around the region, even if these adjacent voxels have an RMF of the R(SC). However, the memory requirement of the proposed algorithm is still less than the double size of the memory space required for the (SC)R algorithm.

5.5 Pre-processing Time

Table 2 shows the pre-processing time to generate voxel attributes. For both data sets, the processing time to calculate intensities and opacities is dominant because all of resampling points with a resolution 4^3 times higher than that of voxels involve interpolation with expensive reconstruction filters, shading and classification. On the other hand, the table indicates

Table 1 Memory requirement (B/voxel).

	R(SC)	(SC)R	proposed
voxel value	1.0	0.0	1.0
intensity	0.0	1.0	1.0
opacity	0.0	1.0	1.0
attribute	0.0	0.0	0.5
total	1.0	2.0	3.5

Table 2 Pre-processing time (sec).

	Marschner	Jaw
intensities and opacities calculation	89.6	744.6
wavelet transform and attributes calculation	21.9	189.5
total	111.5	934.1

that the processing time for wavelet transform and attributes calculation is not critical.

The total time of pre-processing is almost equal to the difference in rendering time between the R(SC) algorithm and the proposed one. Therefore, the pre-processing, which is the overhead of precision control, is not so large. Since voxel attributes do not have to be recomputed until shading and classification parameters are changed, computational costs of pre-processing become relatively lower in the case where only viewing parameters are varied. In addition, it is possible to implement a more efficient routine for interpolation of voxel values and their derivatives by exploiting its regularity. This remains as a future study.

6. Conclusions

In this paper, we have proposed the pre-attributed resampling algorithm for controlled-precision volume ray-casting that is designed to efficiently synthesize high-quality images. The proposed algorithm exploits uneven distribution of computational costs necessary to achieve a given accuracy in each domain, by resampling with an adequate reconstruction method and a proper subinterval. The precision control is based on local information about the constant regions and the errors of the numerical quadrature. To suppress the overhead of precision control, such information is obtained and stored as voxel attributes in pre-processing.

Through experimental results, we have evaluated the accuracy/processing-time performance of the proposed algorithm in comparison with the conventional volume ray-casting algorithms based on either the (SC)R or the R(SC). The experimental results clarified the properties of

the two conventional algorithms:

- The rendering time for the R(SC) algorithm dramatically increases as an RMS decreases.
- The (SC)R algorithm generates images with noticeable errors at the lowest costs.

On the other hand, the experimental results indicated that the proposed algorithm achieved accurate visualization equivalent to the R(SC) algorithm at much lower computational costs. The proposed algorithm is suited for highly accurate visualization at reasonable costs because the algorithm outperforms the R(SC) algorithm for accurate visualization in terms of accuracy/processing-time performance.

Computational costs for generation of voxel attributes and the data size of voxel attributes are the drawbacks of the proposed algorithm. In future research we are going to refine our algorithm by taking into account efficient computation and a smart data structure for voxel attributes.

Acknowledgments The authors would like to thank the anonymous reviewers for their constructive comments. Jaw data set was generated from volume data provided by North Carolina Memorial Hospital. This research was partially supported by Grant-in-Aid, the Ministry of Education, Grant No.10558038.

References

1) Levoy, M.: Display of Surfaces from Volume Data, *IEEE Computer Graphics and Applications*, Vol.8, No.3, pp.29–37 (1988).
2) Marschner, S.R. and Lobb, R.J.: An Evaluation of Reconstruction Filters for Volume Rendering, *Proc. 1994 Symposium on Visualization*, pp.100–107 (1994).
3) Wolberg, G.: *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA (1990).
4) Bentum, M.J. and Malzbender, T.: Frequency Analysis of Gradient Estimators in Volume Rendering, *IEEE Trans. Visualization and Computer Graphics*, Vol.2, No.3, pp.242–254 (1996).
5) Möller, T., Machiraju, R., Mueller, K. and Yagel, R.: Evaluation and Design of Filters Using Taylor Series Expansion, *IEEE Trans. Visualization and Computer Graphics*, Vol.3, No.2, pp.184–199 (1997).
6) Mitchell, D.P. and Netravali, A.N.: Reconstruction Filters in Computer Graphics, *Computer Graphics*, Vol.22, No.4, pp.221–228 (1988).
7) Möller, T., Mueller, K., Kurzion, Y., Machiraju, R. and Yagel, R.: Design Of Accurate And Smooth Filters For Function And Derivative Reconstruction, *Proc. 1998 Symposium on Volume Visualization*, pp.143–151 (1998).
8) Sánchez, R. and Carvajal, M.: Wavelet Based Adaptive Interpolation For Volume Rendering, *Proc. 1998 Symposium on Volume Visualization*, pp.127–134 (1998).
9) Novins, K.: Controlled Precision Volume Rendering, *Proc. 1992 Workshop on Volume Visualization*, pp.83–89 (1992).
10) Upson, C. and Keeler, M.: V-BUFFER: Visible Volume Rendering, *Computer Graphics*, Vol.22, No.4, pp.59–64 (1988).
11) Drebin, R.A., Carpenter, L. and Hanrahan, P.: Volume Rendering, *Computer Graphics*, Vol.22, No.4, pp.65–74 (1988).
12) Sabella, P.: A Rendering Algorithm for Visualizing 3D Scalar Fields, *Computer Graphics*, Vol.22, No.4, pp.51–58 (1988).
13) Pozrikidis, C.: *Numerical Computation in Science and Engineering*, Oxford University Press, New York (1998).
14) Porter, T. and Duff, T.: Compositing Digital Images, *Proc. SIGGRAPH'84*, pp.253–259 (1984).
15) Muraki, S.: Volume Data and Wavelet Transforms, *IEEE Computer Graphics and Applications*, Vol.13, No.4, pp.50–56 (1993).
16) Westermann, R.: A Multiresolution Framework for Volume Rendering, *Proc. 1994 Symposium on Volume Visualization*, pp.51–58 (1994).
17) Mallat, S.G.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.11, No.7, pp.674–693 (1989).
18) Möller, T., Machiraju, R., Mueller, K. and Yagel, R.: A Comparison Of Normal Estimation Schemes, *Proc. 1997 Symposium on Visualization*, pp.19–26 (1997).
19) Möller, T., Machiraju, R., Mueller, K. and Yagel, R.: Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering, *Proc. 1996 Symposium on Volume Visualization*, pp.71–78 (1996).

(Received February 6, 2000)
(Accepted June 2, 2000)

Kentaro Sano is currently a research associate of Graduate School of Information Sciences, Tohoku University. His research interests include volume visualization and parallel processing. He received the M.E. and D.E. degrees in Information Science from Tohoku University in 1997 and 2000, respectively. He is a member of the IPS of Japan.

Hiroaki Kobayashi is currently an associate professor of Department of Computer and Mathematical Sciences, Graduate School of Information Sciences, Tohoku University. In 1995, and from 1997, he has been also a visiting associate professor of Computer Systems Laboratory/Department of Electrical Engineering, Stanford University. His research interests include high-speed microprocessor architecture, parallel processing systems and applications, computer graphics and volume visualization. He received the B.E. degree in Communication Engineering, and the M.E. and D.E. degrees in Information Engineering from Tohoku University in 1983, 1985 and 1988, respectively. He is a member of the IEEE Computer Society, the ACM and the IPS of Japan.

Tadao Nakamura is currently a professor of Graduate School of Information Sciences, Tohoku University. Also since 1994, he has been a visiting professor of Computer Systems Laboratory at Stanford University. His research interests include computer architecture.