Advanced Interactive Preintegrated Volume Rendering with a Power Series

Byeonghun Lee and Yeong-Gil Shin

Abstract—Preintegrated volume rendering produces high-quality renderings without increased sampling rates. However, a look-up table of a conventional preintegrated volume rendering requires a dimensionality of two, which disturbs interactive renderings when the transfer function is changed. Furthermore, as the resolution of the volume data set increases, the memory space required is impractical or inefficient, especially on GPUs. In the past, several approximation methods have been proposed to reduce the complexity of both the time and memory requirement, but most of them do not correctly present thin opaque structures within slabs and ignore the self-attenuation. We propose an advanced interactive preintegrated volume rendering algorithm that achieves not only high-quality renderings comparable to the conventional ones, but also $O(n)$ time and memory space requirements even with the self-attenuation within the slabs applied. The algorithm proposed in this paper decomposes the exponential term of the ray integration equation into a power series of a finite order in the form of a linear combination to build a one-dimensional look-up table. Moreover, the proposed algorithm effectively applies the self-attenuation that is caused by fully opaque isosurfaces, by introducing an opaque prediction table. Experimental results demonstrate that the proposed algorithm offers renderings visibly identical to existing preintegrated volume renderings without degrading rendering speed.

Index Terms—Preintegration, direct volume rendering, ray casting, power series

1 Introduction

Volume rendering is a powerful tool that visualizes a three-dimensional volume data set such as a computerized tomography (CT) data set and a scientific data set. Among volume visualization methods, direct volume rendering (DVR) [1] provides interactive visualization results by adjusting a transfer function. However, high sampling rates are required to acquire high-quality renderings, leading to slow rendering performance. Some studies have accelerated the performance by skipping or leaping unnecessary spaces [2], [3]. Shear-warp factorization [4], [5] remarkably accelerated the performance by ordering the volume data set in the object space. Employing specialized hardwares [6], [7], [8], [9] also brought considerable performance improvements. However, all methods required higher sampling operations or much more memory space to visualize the data effectively.

Preintegrated volume rendering [10] remedied these problems without reference to other software or hardware approaches (see Fig. 1). Preintegrated volume rendering builds a look-up table that defines the color and opacity for each given pair of an entrance scalar value and an exit scalar value. This dramatically improves the rendering quality without increasing the sampling rates. In contrast, the conventional preintegration table consists of three parameters: the entrance scalar value, the exit scalar value, and the distance between them. This three-dimensional look-up table often takes a large amount memory space. Engel et al. reduced it to a two-dimensional table by adopting a constant distance between the entrance and the exit scalar values.

Even when the dimensionality is reduced, updating the look-up table takes a long time because it runs in the order of $O(n^3)$. Lum et al. [11] optimized it to $O(n^2)$ using a coherency of adjacent elements without sacrificing the rendering image quality. However, as the data set resolution increases, it becomes more difficult to achieve an interactive rendering speed via a two-dimensional look-up table. Moreover, it is inefficient to allocate a look-up table in memory in high resolution.

To alleviate this problem, some studies have further reduced the dimensionality of the look-up table. Engel [8] achieved a one-dimensional look-up table by setting the maximum extinction density. Kye et al. [12] proposed an approximation with arithmetic means instead of geometric means. As a result, the complexities of the update time and the memory space are reduced to $O(n)$, so the interactive rendering was realized when the transfer function changes. Nevertheless, if the transfer function has fully opaque values, these algorithms may generate much more transparent rendering results than the results of the conventional preintegrated volume rendering.

In this paper, we propose an advanced interactive preintegrated volume rendering algorithm that is as effective as an algorithm using a two-dimensional look-up table while maintaining the time and space complexity of $O(n)$. We apply a power series expansion, which is a modification of a Taylor series, to decompose the exponential terms of the ray integral into linear combinations. Because the power series is a type of polynomial, it

- *B. Lee is with the Department of Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon 440-769, Gyeonggi-Do, Korea. E-mail: intellee@skku.edu, intellee@cglab.snu.ac.kr.*
- *Y.-G. Shin is with the Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea. E-mail: yshin@cglab.snu.ac.kr.*

Manuscript received 21 June 2012; revised 27 Sept. 2012; accepted 1 Nov. 2012.
Recommended for acceptance by K. Mueller.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-06-0112.
Digital Object Identifier no. 10.1109/TVCG.2012.313.

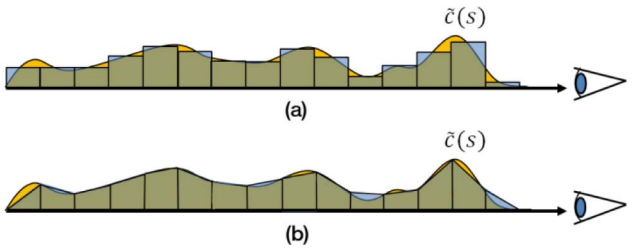

Fig. 1. Comparison of the reconstruction in (a) postclassification and (b) preintegration.

is possible to calculate the emission and absorption densities for a given slab by subtracting two accumulated values. Furthermore, we apply the propose algorithm to the preintegrated volume rendering using nonlinear gradient interpolation [13] which uses two-dimensional look-up tables.

The remainder of this paper is organized as follows: In the next section, we review previous work. Section 3 describes the background. The proposed preintegrated volume rendering is described in Section 4. Section 5 presents the experimental results and Section 6 concludes this paper.

2 Related Work

In biomedical and scientific visualizations, the rendering quality has been an active research area together with interactive or real-time rendering. Levoy [1] introduced DVR, which does not extract the surface information from a volume data set. Because a volume data set does not have emission and attenuation information, a transfer function is applied to map scalar values of the volume data set to emission and attenuation. Postclassification evaluates the transfer function after reconstruction, and this is the most widely employed technique. However, postclassification may lead to a poor rendering quality with high-frequency volume data or a high-frequency transfer function. Roettger et al. [9] proposed supersampling and accurate clipping in an effort to improve the rendering quality around high-frequency transfer functions and data sets. Lee et al. [14] introduced *virtual sampling*, which performs supersampling via curve interpolation within ray segments rather than reconstructing from the volume data set. This improved the rendering quality, but is only efficient when higher order reconstruction filter is used.

Many studies have involved higher quality reconstruction. Csebfalvi [15], [16] adopted prefiltered reconstructions that led to drastic improvements in quality. Reconstruction from the frequency domain also effectively improved the rendering images [17], [18]. However, because these methods require very heavy computation amounts, achieving interactive rendering is difficult.

Unlike the approaches mentioned above, preintegrated volume rendering generates high-quality rendering images without increasing the computation time. Preintegrated volume rendering [10] integrates the transfer function a priori and generates a look-up table for every possible entrance and exit scalar value pair. However, a conventional preintegrated volume rendering requires a two-dimensional look-up table, which makes it difficult to render interactively because the updating of the look-up table time complexity is $O(n^3)$. Lum et al. [11] reduced this complexity to $O(n^2)$ using the coherency of adjacent elements of the look-up table without sacrificing the rendering image quality. However, the updating of the look-up table is still not done at an interactive rate for a higher resolution volume data set such as medical data having a 12-bit resolution.

Some studies reduced the dimensionality of the look-up table. Engel [8] approximated the preintegration by setting the maximum extinction density to a theoretically infinite value. Kye et al. [12] improved on this work by replacing a geometric mean with an arithmetic mean and showed more accurate rendering results. Both of these methods can achieve interactive preintegrated volume rendering as the transfer function changes; however, they cannot express thin opaque structures within the slab. In addition, given that self-attenuation in a given ray segment is ignored in these methods, the rendering results may be whiter than those in conventional preintegrated volume rendering. Guetat et al. [13] considered the continuity of gradient vectors within the slab and achieved higher quality renderings. But the dimensionality of the preintegration look-up table still remains two, and four look-up tables are required for its implementation.

To achieve higher continuity in preintegrated volume rendering, a derivation of the second-order polynomial method was proposed [19]. Because a quadric polynomial approximates the original signal better than a linear one, this method results in better rendering quality. However, it requires a three-dimensional look-up table; hence, the interactivity with these renderings cannot be achieved easily as the transfer function changes. Moreover, much more memory is required to store the look-up table.

3 Background

3.1 Direct Volume Rendering

DVR performs an integration processes for each ray for a volume data set as an optical medium. In addition, the emission and absorption of the volume are evaluated as the rays pass through the volume. Ray integration is expressed as follows:

$$I = \int_0^D \tilde{c}(s(x(\lambda)))e^{-\int_0^\lambda \tau(s(x(\lambda')))d\lambda'}d\lambda. \tag{1}$$

Here, I is the integrated intensity from the observer to a maximum distance of D assuming that there is no medium after D. $x(\lambda)$ is the position along the ray of distance λ from the observer, and $s(x)$ is the scalar value at the position x. The color density $\tilde{c}(s)$ and extinction density $\tau(s)$ are evaluated via the transfer function. This integral should be approximated discretely so as to obtain its color and opacity. Dividing the ray into a constant distance, we can obtain an approximation of the color and opacity using a Riemann sum

$$I \approx \sum_{i=0}^{n} \tilde{C}_i \prod_{j=0}^{i-1}(1-\alpha_j), \tag{2}$$

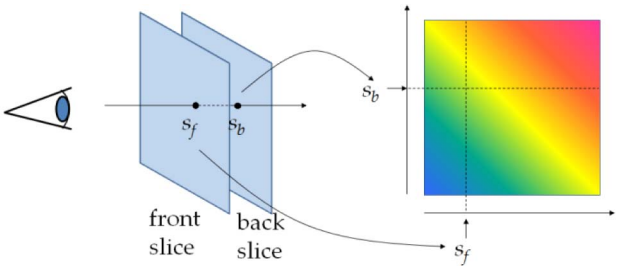

Fig. 2. Building a preintegrated look-up table for each given pair of entrance (s_f) and exit (s_b) values.

$$\alpha_i \approx 1 - e^{-\tau(s(x(id)))d}, \tag{3}$$

$$\tilde{C}_i \approx \tilde{c}(s(x(id)))d, \tag{4}$$

where $d = D/n$. This approximation is effective when both the volume data and the transfer function contain low frequencies. When high-frequency features are contained, the sampling distance d should be small to acquire high-quality renderings.

3.2 Preintegrated Volume Rendering

Preintegrated volume rendering employs a slab-by-slab rendering process instead of slice-by-slice rendering assuming a linear transition within a slab. Let f and b be the distances of the entrance and exit position of the slab, respectively. The integrated value of the slab is then determined as follows:

$$I(f,b) = \int_f^b \tilde{c}(s(x(\lambda)))e^{-\int_f^\lambda \tau(s(x(\lambda')))d\lambda'} d\lambda. \tag{5}$$

Assuming linear transitions of scalar values within a given segment of the ray, the preintegrated color and opacity are

$$\alpha_i \approx 1 - e^{-d\int_0^1 \tau((1-\omega)s_f+\omega s_b)d\omega}, \tag{6}$$

$$\tilde{C}_i \approx d\int_0^1 \tilde{c}((1-\omega)s_f+\omega s_b)e^{-d\int_0^\omega \tau((1-\omega')s_f+\omega' s_b)d\omega'} d\omega, \tag{7}$$

where s_f and s_b are the scalar values of the entrance and exit position of the slab, respectively. These equations can be generalized to the functions whose inputs are s_f, s_b, and d

$$\alpha(s_f,s_b,d) \approx 1 - e^{-\frac{d}{s_b-s_f}(T(s_b)-T(s_f))}, \tag{8}$$

$$\tilde{C}(s_f,s_b,d) \approx \int_{s_f}^{s_b} \tilde{c}(s)\frac{d}{s_b-s_f}e^{-\frac{d}{s_b-s_f}(T(s)-T(s_f))} ds, \tag{9}$$

where $T(s) = \int_0^s \tau(s')ds'$. The preintegrated color is usually approximated while ignoring self-attenuation, as follows:

$$\tilde{C}(s_f,s_b,d) \approx \int_{s_f}^{s_b} \tilde{c}(s)\frac{d}{s_b-s_f} ds = \frac{d}{s_b-s_f}(\kappa(s_b)-\kappa(s_f)). \tag{10}$$

Here, $k(x) = \int_o^x c(s)ds$. The transfer function is built during the preprocessing (see Fig. 2). This preprocessing should be done whenever the transfer function changes.

3.3 Approximation Using the One-Dimensional Preintegrated Look-Up Table

Engel [8] introduced an approximation of the one-dimensional preintegrated volume rendering scheme by setting the maximum extinction density instead of using an infinite value. The ray integration is also expressed as follows:

$$I = \int_0^D c(s(x(\lambda)))\tau(s(x(\lambda)))e^{-\int_0^\lambda \tau(s(x(\lambda')))d\lambda'} d\lambda, \tag{11}$$

where $\tilde{c}(s(x(\lambda))) = c(s(x(\lambda)))\tau(s(x(\lambda)))$. This provides high-quality renderings assuming that the volume is similar to low-density particles [20]. However, if τ is large to the point that it causes overweighting or saturation, the assumption is not valid. To prevent this, the maximum extinction density is limited by 1. This leads to more transparent renderings than those of conventional preintegrated volume rendering when high opacities are contained in the transfer function.

Another approximation of the one-dimensional preintegrated volume rendering scheme was proposed using an arithmetic mean instead of a geometric mean [12]. This provides better approximations than [8]. However, if there are thin surfaces within the slabs, it does not present opaque rendering results.

Because both of these algorithms ignore the self-attenuation within slabs, whiter rendering results may be presented when compared to the result from the conventional method.

4 Preintegrated Volume Rendering Using a Power Series

To achieve interactive rendering as the transfer function changes, updating of the look-up table should be done within the interactive time. Previous studies [8], [12] reduced the dimensionality of the look-up table efficiently but they may not be as expressive as the result using a two-dimensional look-up table [10]. Moreover, self-attenuation within the slabs is ignored. We propose an algorithm that achieves $O(n)$ complexity of both the time and memory space while preserving the rendering quality. It is comparable to the result using a two-dimensional look-up table [10] and applies self-attenuation.

4.1 Approximation Using the Taylor Series Expansion

Referring back to (6) and (7), α_i is the same as $\tilde{C}_i$ when $\tilde{c}(s)$ is always 1. In other words, $\tilde{C}_i$ is always less than or equal to α_i

$$\tilde{C}_i = K_{\tilde{C}_i}\alpha_i, \tag{12}$$

where $0 \le K_{\tilde{C}_i} \le 1$ and $K_{\tilde{C}_i}$ is equal to $\tilde{C}_i/\alpha_i$. α_i can be easily calculated using (8). However, it is difficult to calculate $\tilde{C}_i$ because of its exponential terms. We approximate (12) using the Taylor series expansion to decompose the exponential terms into linear combinations. $K_{\tilde{C}_i}$ can be approximated as follows:

$$K_{\tilde{C}_i} = \frac{\tilde{C}_i}{\alpha_i} \approx \frac{PS(\tilde{C}_i,N)}{PS(\alpha_i,N)}, \tag{13}$$

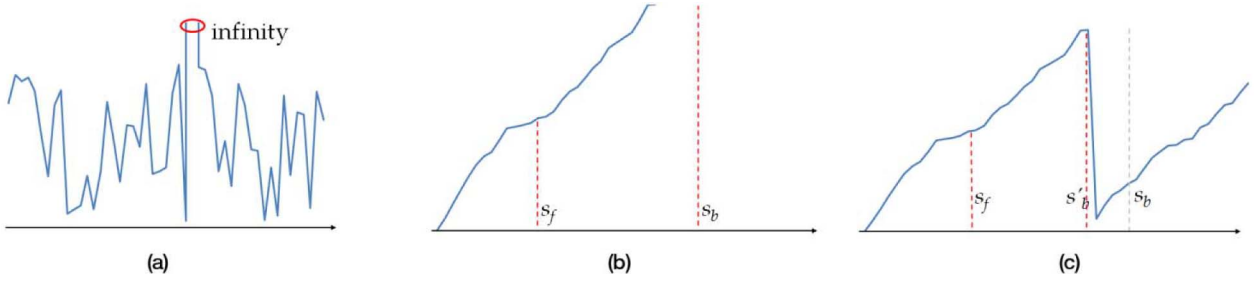

Fig. 3. Example of handling infinity: (a) an example of extinction density distribution, (b) the cumulative function of (a), and (c) is the modified cumulative function of (b).

where $PS(x, N)$ is the Nth order power series of x, which is derived from Taylor series. The higher the order of the series, the more precise the result. To convert the ray integration equation into linear combinations, we convert the ray integration equation as follows:

$$I = \int_0^D \tilde{c}(s(x(\lambda))) \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(\int_0^{\lambda} \tau(s(x(\lambda'))) d\lambda' \right)^n d\lambda. \quad (14)$$

This can be applied to (8) and (9)

$$\alpha(s_f, s_b, d) \approx \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n!} \left(\frac{T(s_b) - T(s_f)}{s_b - s_f} d \right)^n, \quad (15)$$

$$\tilde{C}(s_f, s_b, d) \approx \frac{d}{s_b - s_f} \int_{s_f}^{s_b} \tilde{c}(s) \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(\frac{T(s) - T(s_f)}{s_b - s_f} d \right)^n ds. \quad (16)$$

The preintegrated opacity is identical to that in (8) because it is a function of $(T(s_b) - T(s_f))d/(s_b - s_f)$. It is difficult to decompose the preintegrated color into a linear combination owing to the self-attenuation of the color density. To solve this problem, we use the following equation for associated colors [21]

$$\tilde{c}(s)d = c(s)\tau(s)d \approx c(s)(1 - e^{-\tau(s)d}). \quad (17)$$

Applying (17) to (9) and (16)

$$\begin{aligned} \tilde{C}(s_f, s_b, d) &\approx \int_{s_f}^{s_b} c(s) \left(1 - e^{-\frac{\tau(s)d}{s_b - s_f}} \right) e^{-\frac{d}{s_b - s_f}(T(s) - T(s_f))} ds \\ &= \int_{s_f}^{s_b} c(s) \left(e^{-\frac{T(s) - T(s_f)}{s_b - s_f} d} - e^{-\frac{T(s) + \tau(s) - T(s_f)}{s_b - s_f} d} \right) ds \\ &= \int_{s_f}^{s_b} c(s) \left(\sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(\frac{T(s) - T(s_f)}{s_b - s_f} d \right)^n \right) \\ &\quad - \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \left(\frac{T(s) + \tau(s) - T(s_f)}{s_b - s_f} d \right)^n \Bigg) ds \end{aligned} \quad (18)$$

$$= \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{n!} \left(\frac{d}{s_b - s_f} \right)^n \int_{s_f}^{s_b} c(s) \Big(\big(T(s) + \tau(s) - T(s_f) \big)^n - \big(T(s) - T(s_f) \big)^n \Big) ds. \quad (19)$$

We can obtain (18) applying the Taylor series expansion to (17). Because this equation requires an infinite number of polynomials, we approximate it using finite polynomials, as follows:

$$\begin{aligned} \tilde{C}(s_f, s_b, d) \approx \sum_{n=0}^{N} \frac{(-1)^{n+1}}{n!} \left(\frac{d}{s_b - s_f} \right)^n & (X_n(s_b, -T(s_f)) \\ & - X_n(s_f, -T(s_f))), \end{aligned} \quad (20)$$

$$X_n(x, t) = \sum_{i=0}^{n} \binom{n}{i} t^i X_{n-i}(x), \quad (21)$$

where $X_n(x) = X_n(x, 0) = \int_0^x c(s)((T(s) + \tau(s))^n - T(s)^n) ds$. Using (20) and (21), N one-dimensional preintegrated look-up tables, $X_1(x), \ldots, X_N(x)$, can approximate the conventional one-dimensional preintegrated look-up table while applying self-attenuation. Finally, $K_{\tilde{C}_i}$ is approximated to the following equation applying (15) and (19) to (13)

$$K_{\tilde{C}_i}(s_f, s_b, d) \approx \frac{\sum_{n=1}^{N} \frac{(-1)^{n+1}}{n!} \left(\frac{d}{s_b - s_f} \right)^n \big(X_n(s_b, -T(s_f)) - X_n(s_f, -T(s_f)) \big)}{\sum_{n=1}^{N} \frac{(-1)^{n+1}}{n!} \left(\frac{T(s_b) - T(s_f)}{s_b - s_f} d \right)^n}. \quad (22)$$

4.2 Exception for Infinity

When a transfer function has fully opaque values (i.e., alpha = 1), the corresponding extinction density is an infinity. This makes it difficult to accumulate values for the look-up table. One possible solution is to set a large maximum extinction density. However, this may lead to an overflow because the look-up table stores accumulated values of powers of T. It is acceptable to regard that there is at most one fully opaque scalar value within a given ray if early ray termination (ERT) [22] is employed. Hence, we integrate rays just before the position of the fully opaque value (see Fig. 3), after which postclassification is done for the fully opaque value.

To do this without increasing the sampling rate, we introduce a prediction table, P. The table indexes correspond to the scalar values of the entrance positions of the slabs. In addition, each cell of the table contains the value that is closest to and greater than or equal to its scalar value whose extinction density is infinity. If no opaque value is found, an invalid flag is stored. During ray-casting, a prediction is determined by comparing $P(s_f)$ and sb. If $P(s_f) < s_b$, the exit value of the slab is redefined

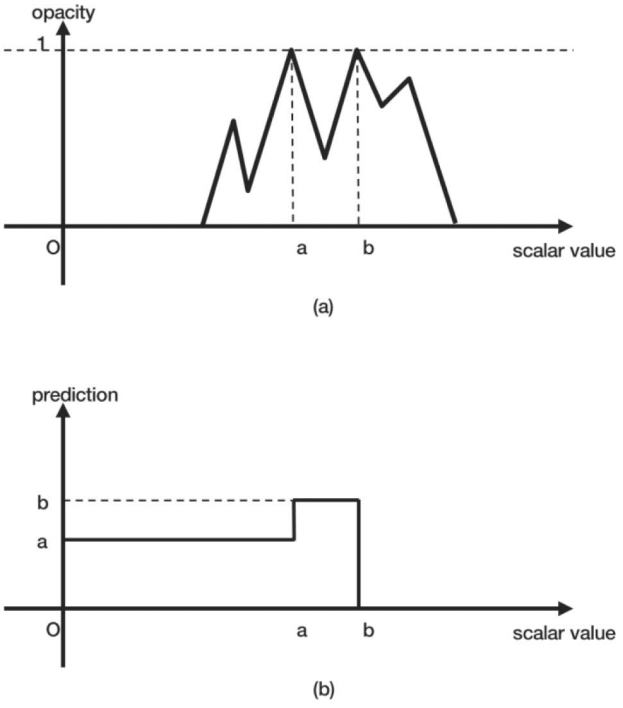

Fig. 4. Building a prediction table: (a) an example of an opacity transfer function and (b) it's prediction table.

as $P(s_f)$ and preintegration and postclassification are successively performed with the exit value of the new slab. Next, the ray is terminated because the opacity is saturated. Fig. 4 is an example of the prediction table which has two fully opaque values.

4.3 Applying to Nonlinear Gradient Interpolation Volume Rendering

The proposed algorithm is also applied to the nonlinear gradient interpolation volume rendering [13]. Guetat et al. [13] uses four two-dimensional look-up tables to achieve nonlinear gradient interpolation within a given slab. Each look-up table is constructed using the following equation:

$$\tilde{C}_k(s_f, s_b) = \frac{d}{s_b - s_f} \int_{s_f}^{s_b} \tilde{c}(s) \left(\frac{s - s_f}{s_b - s_f} \right)^k e^{-\frac{d}{s_b - s_f}(T(s) - T(s_f))} ds, \tag{23}$$

where $k = 0$, 1, 2, and 3. We expand (23) to convert into polynomials, as follows:

$$\begin{aligned}
\tilde{C}_0(s_f, s_b) &= \tilde{C}(s_f, s_b), \\
\tilde{C}_1(s_f, s_b) &= \int_{s_f}^{s_b} \frac{s - s_f}{s_b - s_f} c(s) EXP(s_f, s_b) ds, \\
\tilde{C}_2(s_f, s_b) &= \int_{s_f}^{s_b} \frac{s^2 - 2s_f s - s_f^2}{(s_b - s_f)^2} c(s) EXP(s_f, s_b) ds, \\
\tilde{C}_3(s_f, s_b) &= \int_{s_f}^{s_b} \frac{s^3 - 3s_f s^2 + 3s_f^2 s - s_f^3}{(s_b - s_f)^3} c(s) EXP(s_f, s_b) ds,
\end{aligned} \tag{24}$$

where $EXP(s_f, s_b) = e^{-\frac{T(s) - T(s_f)}{s_b - s_f} d} - e^{-\frac{T(s) + \tau(s) - T(s_f)}{s_b - s_f} d}$ which was introduced in (18). Equation (24) can be decomposed of linear combinations, as follows:

TABLE 1
Comparison of Sizes of Look-Up Tables

Data Bit Resolution	PI_{2D} (32-bit float)	PI_{1D} (32-bit float)	$PI_{PS}(1)$ (32-bit float)	$PI_{PS}(3)$ (64-bit float)
8-bit	1 MB	4 KB	4 KB	24 KB
12-bit	256 MB	64 KB	64 KB	384 KB

$$\begin{aligned}
\tilde{C}_0(s_f, s_b) &= \tilde{C}'_0(s_f, s_b), \\
\tilde{C}_1(s_f, s_b) &= \frac{1}{s_b - s_f} \left(\tilde{C}'_1(s_f, s_b) - s_f \tilde{C}'_0(s_f, s_b) \right), \\
\tilde{C}_2(s_f, s_b) &= \frac{1}{(s_b - s_f)^2} \left(\tilde{C}'_2(s_f, s_b) - 2s_f \tilde{C}'_1(s_f, s_b) \right. \\
&\quad \left. + s_f^2 \tilde{C}'_0(s_f, s_b) \right), \\
\tilde{C}_3(s_f, s_b) &= \frac{1}{(s_b - s_f)^3} \left(\tilde{C}'_3(s_f, s_b) - 3s_f \tilde{C}'_2(s_f, s_b) \right. \\
&\quad \left. + 3s_f^2 \tilde{C}'_1(s_f, s_b) - s_f^3 \tilde{C}'_0(s_f, s_b) \right),
\end{aligned} \tag{25}$$

where $\tilde{C}'_k = \int_{s_f}^{s_b} s^k c(s) EXP(s_f, s_b) ds$. Regarding $s^k c(s)$ as a single term, $c_s^k(s)$, these look-up tables can be converted to one-dimensional tables using the method in Section 4.1.

5 Experimental Results

The proposed algorithm is implemented in Direct3D 11 and HLSL on an NVIDIA GTX 680 GPU with 2.0 GB of memory. All experiments were performed on Intel i5 CPU with 8 GB RAM and the same GPU above. The reconstruction filter is a linear filter, which is most widely used. For shading, constant gradient vectors are used. ERT and empty space skipping (ESS) [22] are employed in the conventional ray-casting frameworks. Static sized blocks for ESS are employed rather than an octree structure. The rendering image resolution is 800×800 in all cases. For ease of explanation, we refer to the postclassification as 'PC,' the two-dimensional preintegrated look-up table [10] as 'PI_{2D},' the one-dimensional preintegrated look-up table [12] as 'PI_{1D},' and the algorithm proposed in this paper as '$PI_{PS}(n)$,' and n is the order of the power series.

Table 1 shows a comparison of the look-up table size for each algorithm. As shown, the proposed look-up table is practical for high-resolution data sets such as 12-bit data sets. Table 2 shows the look-up table computation times of each algorithm. An optimized computation of the two-dimensional look-up table algorithm [11] is used for a fair comparison. Although [11] provides practical computation time, it disturbs an interactive rendering when the transfer function is changed. Whereas, the look-up table computation time for PI_{PS} is enough to be neglected.

TABLE 2
Comparison of Computational Times of Each Look-Up Table (MS)

Table size	PI_{2D} (32-bit float)	PI_{1D} (32-bit float)	$PI_{PS}(1)$ (32-bit float)	$PI_{PS}(3)$ (64-bit float)
256	0.126	0.0039	0.0186	0.0386
4096	1569	0.0501	0.2211	0.5172

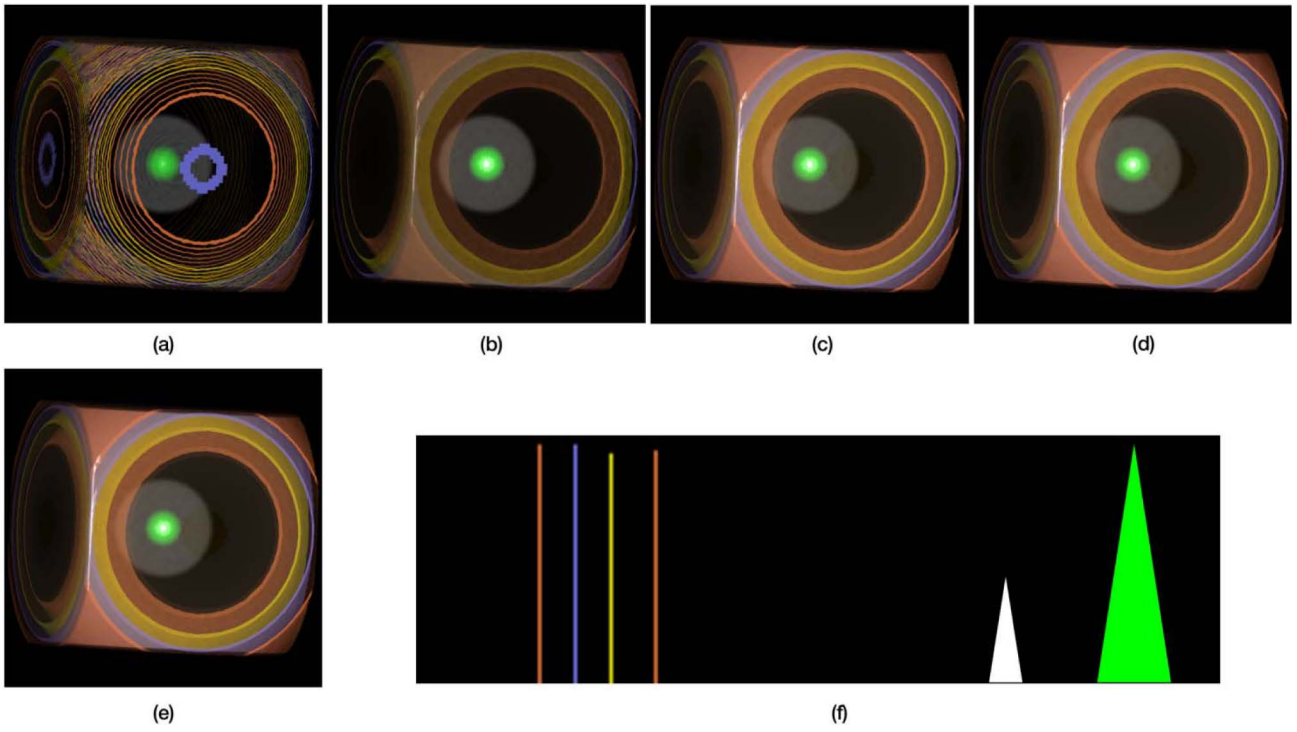

Fig. 5. Sphere data set ($64 \times 64 \times 64$, 8-bit) rendered using (a) PC, (b) PI_{1D}, (c) PI_{2D}, (d) $\mathrm{PI}_{PS}(1)$, and (e) $\mathrm{PI}_{PS}(3)$. (f) The transfer function.

5.1 Rendering Quality Comparison

We conducted experiments comparing each algorithm to show the effectiveness of the proposed algorithm.

Fig. 5 shows the rendering results using the 8-bit sphere data set [23]. Given that a transfer function having some high frequencies (not fully opaque) was used, there were striking visible differences. PC did not visualize thin surfaces, and PI_{1D} did them more transparent than the result of PI_{2D}. The proposed algorithm gave results comparable to those with $\mathrm{PI}_{PS}(1)$ and $\mathrm{PI}_{PS}(3)$. Furthermore, the rendering result of $\mathrm{PI}_{PS}(1)$ was visibly identical to $\mathrm{PI}_{PS}(3)$. We can infer from this that the first order power series is enough to approximate PI_{2D}. Thus, $\mathrm{PI}_{PS}(1)$ is used in the remainder of the paper.

Fig. 6 demonstrates the rendering results with the 12-bit head data set. A trapezoidal shape transfer function, which is most widely used for medical data set visualization, was used. PI_{1D} resulted in a whiter image than PI_{2D}'s image because self-attenuation was ignored. Given that the transfer function has fully opaque values, opaque prediction is necessary to apply self-attenuation effectively. As a result, the opaque value was effectively predicted and the proposed algorithm visualized opaque surface as PI_{2D} did.

We conducted another experiment with the 12-bit chest data set. Fig. 7 shows the rendering results adapting PC, $\mathrm{PI}_{1D}, \mathrm{PI}_{2D}$, and $\mathrm{PI}_{PS}(1)$. The transfer function for the experiment has a sharp opaque scalar value near the scalar value of the skin. PC was performed with the $4\times$ sampling rate. Even though the sampling rate was high, PC did not render thin surfaces effectively. Fig. 7d shows the rendering results from the proposed algorithm. The proposed algorithm rendered opaque surfaces as accurately as the result of PI_{2D} shown in Fig. 7c.

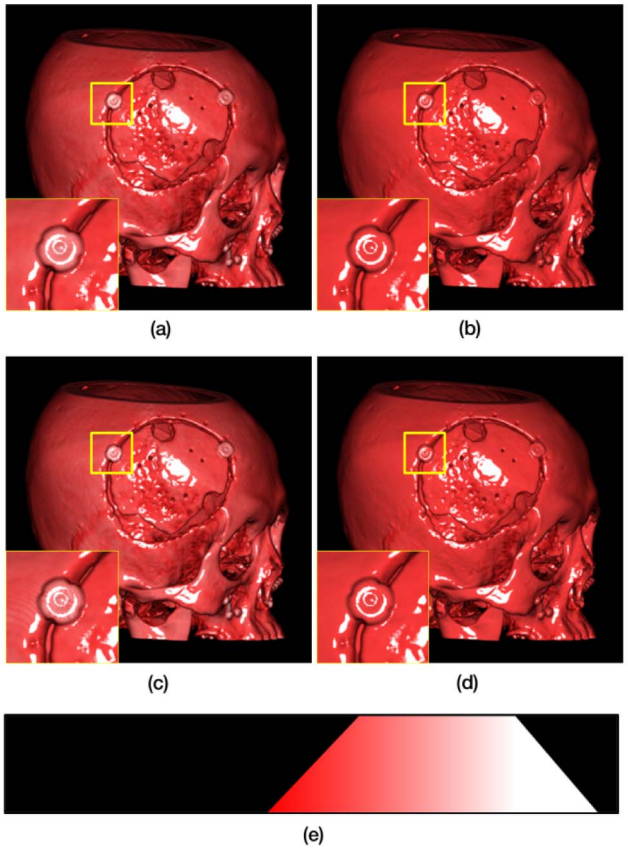

Fig. 6. Head data set ($512 \times 512 \times 300$, 12-bit) rendered using (a) PI_{1D}, (b) PI_{2D}, (c) $\mathrm{PI}_{PS}(1)$ without opaque prediction, and (d) $\mathrm{PI}_{PS}(1)$ with opaque prediction. (e) The transfer function.

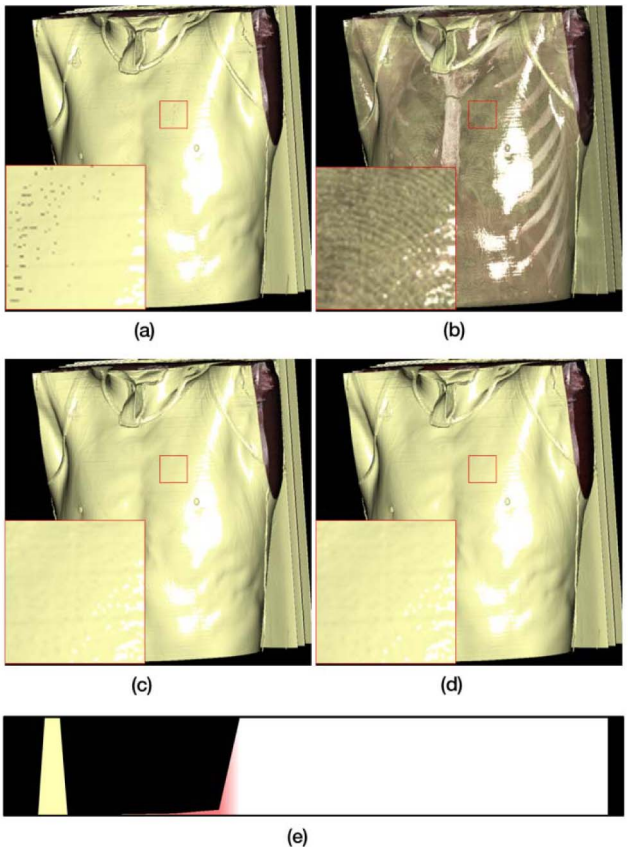

Fig. 7. Chest data set ($512 \times 512 \times 551$, 12-bit) rendered using (a) PC with the $4\times$ sampling rates, (b) PI_{1D}, (c) PI_{2D}, and (d) PI_{PS} (1) with opaque prediction. (e) The transfer function.

We also conducted an experiment with a case having multiple opaque isosurfaces (i.e., multiple Dirac impulses in the transfer function). Fig. 8 shows the rendering results using the polynomial function data set [24]. Although there are many thin opaque isosurfaces, PI_{1D} did not accurately visualize the surfaces. The proposed algorithm effectively visualized multiple surfaces from the given transfer function.

The proposed algorithm can be adapted to the algorithm in [13]. For ease of explanation, we refer to [13] as 'NLPI_{2D},' and our modification as 'NLPI_{PS}.' Same as above, we employed the first-order power series. Fig. 9 shows the comparison results. Differently from the previous experiments, this experiment requires higher precision arithmetic operations since the look-up tables are built from the power of scalar value s. Thus, we employed a double floating data type *structured buffer*, which is a kind of resource type in Direct3D 11 and double precision arithmetic. Nevertheless, the proposed algorithm generated slightly different rendering result. To adopt the proposed algorithm, we expanded the original equations to (25). During this procedure, errors were accumulated because equation (25) sums approximated values.

For a quantitative error analysis, we computed the peak signal-to-noise ratios (PSNR) for each rendering result between PI_{2D} and $\mathrm{PI}_{PS}(1)$. An analysis was done for each color channel. As shown in Table 3, most of the PSNRs were greater than 40 dB, and the minimum was larger than 30 dB. Noting that in general it is difficult to observe the difference between two images with a PSNR greater than 30 dB [25], it is clear see that the proposed algorithm is comparable to PI_{2D} from an objective perspective as well.

5.2 Rendering Speed Comparison

We tested the rendering speeds of each algorithm (PC, PI_{1D}, PI_{2D}, $\mathrm{PI}_{PS}(1)$) from the viewpoints of 360 directions. Table 4 shows the overall performance of the experiments. The proposed algorithm resulted in 1.1-1.2 times faster rendering speed than PI_{2D}. In the chest data set case, the rendering speeds of PC and PI_{1D} were slower than that of

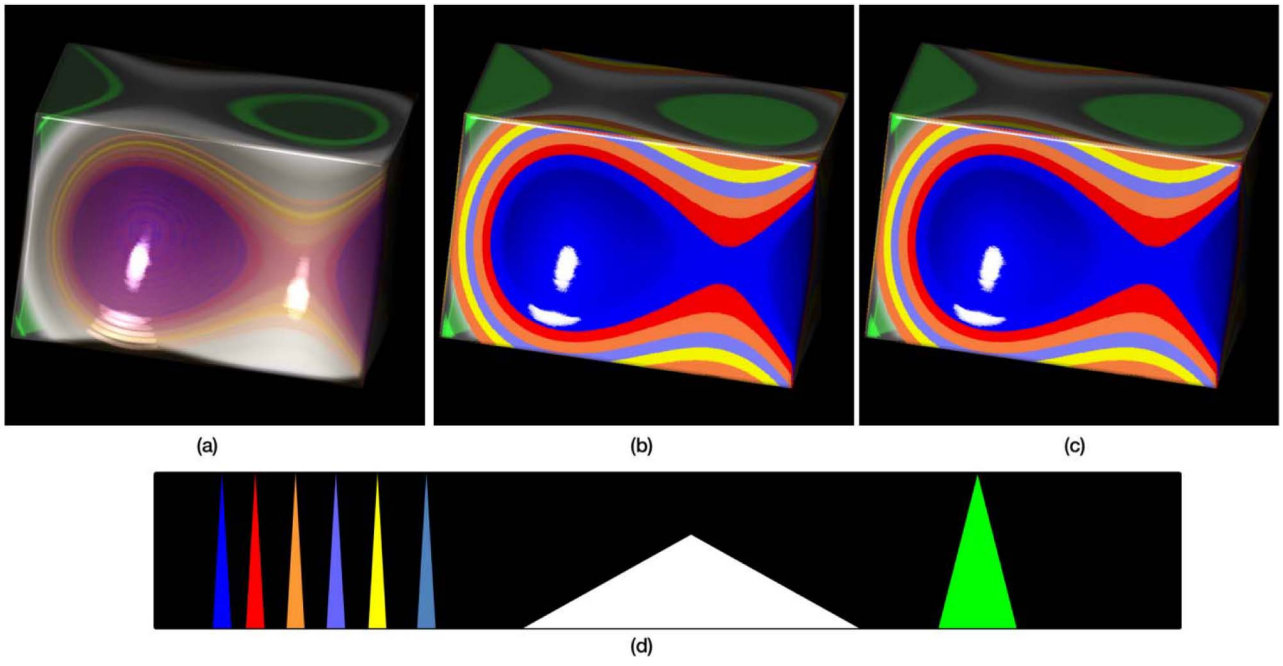

Fig. 8. Polynomial function data set ($64 \times 64 \times 64$, 8-bit) rendered using (a) PI_{1D}, (b) PI_{2D}, and (c) $\mathrm{PI}_{PS}(1)$ with opaque prediction. (d) The transfer function.

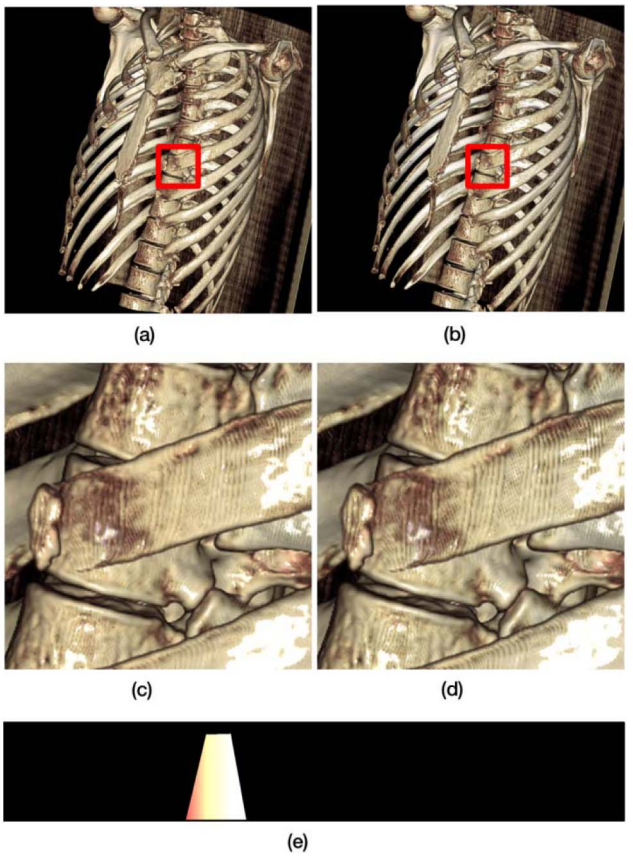

Fig. 9. Ribs data set ($512 \times 512 \times 551$, 12-bit) rendering results using (a) NLPI_{2D} and (b) NLPI_{PS}. (c) and (d) are magnified images of (a) and (b), respectively. (e) The transfer function.

the proposed algorithm (e.g., the chest data set) because they did not render opaque surfaces, which led to an increase in the sampling count.

Opaque prediction made a little speed degradation due to additional control logic for predicting whether or not opaque values exist within a given slab. In the case of the Head data set (see Figs. 6c and 6d), it was about 1.1 times faster if the opaque prediction was not applied.

In the case of NLPI_{PS}, there was significant speed degradation. This is due to double precision floating-point data and arithmetic, which is usually slower than single precision floating-point arithmetic, especially on GPUs.

6 Conclusion

We proposed an advanced preintegrated volume rendering using a power series that maintains an interactive level of rendering performance as the transfer function changes and provides a nearly identical result as the conventional scheme using a two-dimensional look-up table. The proposed algorithm uses a look-up table of size of $O(n)$ while adopting a power series, which is as expressive as the conventional two-dimensional preintegrated look-up table. Moreover, updating the look-up table time complexity becomes $O(n)$. These features allow interactive and high-quality preintegrated volume renderings. Most rendering results from the proposed algorithm gave visibly identical results compared to those of the conventional two-dimensional preintegrated look-up table with any type of transfer function and data set.

TABLE 3
PSNRs of Each Rendering Result between PI_{2D} and the Proposed Algorithm (dB)

Dataset	Red	Green	Blue	Average
sphere (Figs. 5 (c) and (d))	41.90	44.53	45.42	43.95
head (Figs. 6 (b) and (d))	52.80	49.10	49.23	50.38
chest (Figs. 7 (c) and (d))	46.69	45.29	45.29	45.76
function (Figs. 8 (b) and (c))	65.71	63.07	66.21	65.00
ribs (Figs. 9 (a) and (b))	30.66	31.89	32.04	31.53

The proposed algorithm was also effective with higher resolution data sets such as 12-bit data sets. As a result, the PSNRs between the conventional two-dimensional preintegrated volume rendering and the proposed algorithm were high enough that it was difficult to distinguish between the two resulting images from each algorithm.

The main drawback of the proposed algorithm is the requirement of opaque prediction. Even if there is no opaque value in the transfer function, the main loop of ray casting should always make a prediction whether or not the opaque values exist within the given slab. As a result, it was about 1.1 times faster if the opaque prediction was not applied. However, the proposed algorithm using opaque prediction is fast enough to be practical. When the proposed algorithm was applied to [13] in Section 4.3, the performance for both rendering quality and speed was not as effective as the results using constant gradient vectors. Especially, the rendering speed was only the half. Unlike the conventional volume rendering frameworks, double precision arithmetic was used for look-up tables, leading to a decrease in the rendering speed, to relax precision errors. However, because this is an inherently hardware manufacturing issue, we expect this issue to be resolved as hardware performance increases. Fortunately, hardware vendors are making an effort to improve the double precision arithmetic performance.

In the future, we are planning to improve both the rendering quality and speed of NLPI_{PS}. A multi-dimensional transfer function is worth studying, as this requires

TABLE 4
Overall Rendering Speed Performance in Frames Per Second

Dataset	PC	PI_{1D}	PI_{2D}	PI_{PS}
sphere (Fig. 5)	300.65	233.44	277.43	290.28
head (Fig. 6)	192.66	186.84	158.09	185.53 (202.11 w/o prediction)
chest (Fig. 7)	34.99	21.23	51.91	61.54
ribs (Fig. 9)	n/a	n/a	38.08 (NLPI_{2D})	19.43 (NLPI_{PS})

much more memory space to adopt preintegrated volume rendering.

PSEUDO CODE

```
//CPU side
Array tf; // transfer function
Array tauSum;   // cumulative table of tau
RGBA_Array ps;   // power series table

BuildTauSum(tauSum)
{
  // build accumulated table of tau
  for each element of tauSum
    tauSum[n+1] = tauSum[n] + tau[n];
  ....
}

BuildPowerSeries(ps, tauSum, order)
{
  // build power series table
  for each element of ps
  {
    t = tauSum[n]^order – tauSum[n-1]^order;
    ps[n+1] = ps[n] + color[n]*t;
  }
  ....
}

// GPU side, the 1st order is used for an example
RayCasting()
{
  Ray_setup(ray);
  output = 0;    // output pixel color

  for (ray is between startPosition and endPostion)
  {
    sf = get_value(ray);
    sb = get_value(ray + ray_step);
    opaque = PredictOpaque(sf, sb);
    // fetch from the power series table
    k_sf = ps[sf];          // RGBA color
    k_sb = ps[sb];          // RGBA color
    color = (k_sb – k_sf)/(sb – sf);   // 1st order
    alpha = 1 – exp(-color.a);         // a is used as tau
    color = color/color.a * alpha;   // equation (22)
    Composite(output, color);
    ....  // check ERT & opaque
    ray = ray + ray_step;
  }

  if(opaque)
  {
    // apply the opaque value
    color = tf[opaque_position];
    Composite(output, color);
  }
}
```

ACKNOWLEDGMENTS

This work was supported in part by the Technology Innovation Program the Industrial Strategic Technology Development Program funded by the MKE of Korea (No. 0421-20120054).

REFERENCES

[1] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications,* vol. 8, no. 3, pp. 29-37, May 1988.
[2] R. Yagel and Z. Shi, "Accelerating Volume Animation by Space-Leaping," *Proc. IEEE Visualization Conf.,* pp. 62-69, 1993.
[3] M. Wan, A. Sadiq, and A. Kaufman, "Fast and Reliable Space Leaping for Interactive Volume Rendering," *Proc. IEEE Visualization Conf.,* pp. 195-202, 2002.
[4] P. Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," *Proc. ACM SIGGRAPH,* pp. 451-458, 1994.
[5] J.P. Schulze, R. Niemeier, and U. Lang, "The Perspective Shear-Warp Algorithm in a Virtual Environment," *Proc. IEEE Visualization Conf.,* pp. 207-213, 2001.
[6] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The VolumePro Real-Time Ray-Casting System," *Proc. ACM SIGGRAPH,* 1999. pp. 251-260,
[7] S. Roettger, M. Kraus, and T. Ertl, "Hardware-Accelerated Volume and Isosurface Rendering Based on Cell-Projection," *Proc. IEEE Visualization Conf.,* pp. 109-116, 2000.
[8] K. Engel, "Advanced Volume Rendering Techniques," *Proc. IEEE Visualization Conf.,* 2003.
[9] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser, "Smart Hardware-Accelerated Volume Rendering," *Proc. Symp. Data Visualization,* pp. 231-238, 2003.
[10] K. Engel, M. Kraus, and T. Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading," *Proc. ACM SIGGRAPH/EUROGRAPHICS Workshop Graphics Hardware,* pp. 9-16, 2001.
[11] E.B. Lum, B. Wilson, and K.-l. Ma, "High-Quality Lighting and Efficient Pre-Integration for Volume Rendering," *Proc. Joint Eurographics-IEEE TVCG Symp. Visualization,* pp. 25-34, 2004.
[12] H. Kye, B. Shin, and Y.G. Shin, "Interactive Classification for Pre-Integrated Volume Rendering of High-Precision Volume Data," *Graphical Models,* vol. 70, pp. 125-132, 2008.
[13] A. Guetat, A. Ancel, S. Marchesin, and J.-M. Dischler, "Pre-Integrated Volume Rendering with Non-Linear Gradient Interpolation," *IEEE Trans. Visualization Computer Graphics,* vol. 16, no. 6, pp. 1487-1494, Nov./Dec. 2010.
[14] B. Lee, J. Yun, J. Seo, B. Shim, Y.-G. Shin, and B. Kim, "Fast High-Quality Volume Ray Casting with Virtual Samplings," *IEEE Trans. Visualization Computer Graphics,* vol. 16, no. 6, pp. 1525-1532, Nov./Dec. 2010.
[15] B. Csebfalvi, "Prefiltered Gaussian Reconstruction for High-Quality Rendering of Volumetric Data Sampled on a Body-Centered Cubic Grid," *Proc. IEEE 16th Visualization Conf.,* pp. 311-318, 2005.
[16] B. Csebfalvi, "An Evaluation of Prefileted B-Spline Reconstruction for Quasi-Interpolation on the Body-Centered Cubic Lattices," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 3, pp. 499-512, 2010.
[17] A. Li, K. Mueller, and T. Ernst, "Methods for Efficient, High Quality Volume Resampling in the Frequency Domain," *Proc. IEEE Visualization Conf.,* pp. 3-10, 2004.
[18] M. Artner, T. Möller, I. Viola, and M.E. Gröller, "High-Quality Volume Rendering with Resampling in the Frequency Domain," *Proc. EUROGRAPHICS/IEEE VGTC Symp. Visualization,* pp. 85-92, 2005.
[19] J.-F. E. Hajjar, S. Marchesin, J.-M. Dischler, and C. Mongenet, "Second Order Pre-Integrated Volume Rendering," *Proc. IEEE Pacific Visualization Symp.,* pp. 9-16, 2008.
[20] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 99-108, June 1995.
[21] J.F. Blinn, "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces," *Proc. ACM SIGGRAPH,* pp. 21-29, 1982.
[22] M. Levoy, "Efficient Ray Tracing of Volume Data," *ACM Trans. Graphics,* vol. 9, pp. 245-261, 1990.
[23] *The Volume Library,* http://www.volvis.org, 2013.
[24] *Data,* http://www.vis.uni-stuttgart.de/~engel/pre-integrated/data.html, 2013.
[25] C.-C. Lee, H.-C. Wu, C.-S. Tsai, and Y.-P. Chu, "Adaptive Lossless Steganographic Scheme with Centralized Difference Expansion," *Pattern Recognition,* vol. 41, no. 6, pp. 2097-2106, 2008.

Byeonghun Lee received the BS degree in computer science from Sungkyunkwan University, Korea, and the MS and PhD degrees in computer science and engineering from the Seoul National University, Korea, in 2007, and 2013, respectively. He is currently a postdoctoral researcher in the Department of Computer Engineering, Sungkyunkwan University, Korea. His research interests include high-quality three-dimensional volume visualization and GPU-based three-dimensional reconstruction.

Yeong-Gil Shin received the BS and MS degrees in computer science from Seoul National University, Korea, and the PhD degree in computer science from the University of Southern California, Los Angeles, in 1990. He is currently a professor in the Department of Computer Science and Engineering and the director of the Computer Graphics and Image Processing Laboratory, Seoul National University. His research interests include computer graphics, volume visualization, medical imaging, and image processing.