Fast High-Quality Volume Ray-Casting with Virtual Samplings

Byeonghun Lee, Jihye Yun, Jinwook Seo, *Member*, *IEEE*, Byonghyo Shim, *Senior Member*, *IEEE*, Yeong-Gil Shin, and Bohyoung Kim, *Member*, *IEEE*

Abstract—Volume ray-casting with a higher order reconstruction filter and/or a higher sampling rate has been adopted in direct volume rendering frameworks to provide a smooth reconstruction of the volume scalar and/or to reduce artifacts when the combined frequency of the volume and transfer function is high. While it enables high-quality volume rendering, it cannot support interactive rendering due to its high computational cost. In this paper, we propose a fast high-quality volume ray-casting algorithm which effectively increases the sampling rate. While a ray traverses the volume, intensity values are uniformly reconstructed using a high-order convolution filter. Additional samplings, referred to as *virtual* samplings, are carried out within a ray segment from a cubic spline curve interpolating those uniformly reconstructed intensities. These virtual samplings are performed by evaluating the polynomial function of the cubic spline curve via simple arithmetic operations. The min max blocks are refined accordingly for accurate empty space skipping in the proposed method. Experimental results demonstrate that the proposed algorithm, also exploiting fast cubic texture filtering supported by programmable GPUs, offers renderings as good as a conventional ray-casting algorithm using high-order reconstruction filtering at the same sampling rate, while delivering 2.5x to 3.3x rendering speed-up.

Index Terms—Direct volume rendering, GPU, high quality, curve interpolation.

1 Introduction

Direct volume rendering (DVR) visualizes 3D volume data using a transfer function which maps voxel values (i.e., intensities) to colors and opacities, without extracting the polygonal data. By adjusting the transfer function, users can observe inner structures of the objects in the 3D volume data interactively. However, due to the humongous amount of the 3D volume data to be processed, the rendering process is computationally expensive, and thus an interactive rendering is frequently infeasible. To address this issue, many acceleration techniques, mainly concerning on the optimization of the high-cost DVR computation, have been proposed.

While these optimization techniques accelerate DVR efficiently, they do not support real-time rendering. Instead, specialized rendering hardware solutions such as VolumePro [1] have been introduced to combat the expensive computation of DVR. Although these solutions enable real-time rendering, their high cost limits the wide-spread adoption of them. In contrast, graphics processing units (GPUs) are much cheaper and more flexible than the specialized solutions. GPUs are essentially massive parallel-processing devices with many small processing units. Ever since GPUs became programmable with the introduction of shader languages, programmers have implemented various algorithms including DVR on GPUs [2].

While the aforementioned acceleration techniques or GPU programming provide interactive or real-time rendering, they often sacrifice image quality for a computational performance gain. Among the various DVR techniques, ray-casting DVR is known for its high-quality rendering. In the ray-casting DVR, a ray is cast from an image plane and then traverses the volume data while sampling (reconstructing) the intensity at a uniform (or non-uniform [3]) interval and accumulating colors and opacities evaluated via the transfer function. The reconstruction is performed by convolving the discrete volume data with a continuous filter function. A linear interpolation filter is the most widely used reconstruction filter because it produces reasonably good results with a moderate cost [4]. However, it does not provide smooth reconstructions of data and derivatives, causing artifacts so called moiré patterns. A cubic filter can remedy this problem, but it requires far more computations because it accesses a much greater number of voxels.

Along with these reconstruction errors, DVR is often associated with artifacts when the transfer function has high frequencies (a sharp increase in the opacity at some intensity). In this case, uniform sampling and post-classification (transfer function evaluation after sampling) in conventional ray-casting DVRs yield aliasing artifacts [3]. Increasing the sampling rate can mitigate this problem, but it directly affects the rendering speed. Pre-integration approach [5] reduces artifacts with a slight increase in the rendering cost. However, this approach introduces new artifacts by discretizing scalar values into a 2D lookup table.

1.1 Main Contribution

In this paper, we propose a fast high-quality volume ray-casting algorithm. The proposed algorithm effectively increases the sampling rate by introducing *virtual* samplings on a cubic spline curve interpolating the intensities uniformly sampled along a ray. Further, by introducing the refinement of min max blocks we can avoid inappropriate block skipping during empty space skipping. The proposed algorithm, exploiting fast cubic texture filtering supported by programmable GPUs, delivers rendering quality comparable to a conventional ray-casting algorithm using cubic filtering at the same sampling rate, while achieving 2.5x to 3.3x rendering speed-up.

The remainder of this paper is organized as follows. In the next section we review previous works on quality improvements of volume rendering. The proposed high-quality volume ray-casting algorithm and its implementation are described in Sections 3 and 4, respectively. Section 5 presents the experimental results and Section 6 concludes this paper.

2 Related work

In biomedical and scientific visualizations, rendering quality has been an active research area together with interactive or real-time rendering. Post-classification, which evaluates the transfer function after sampling the data at a sampling point, is simple and widely used due to its conceptual and computational simplicity. However,

- *Byeonghun Lee is with Seoul National University, E-Mail: intellee@cglab.snu.ac.kr.*
- *Jihye Yun is with Seoul National University, E-Mail: jhyun@cglab.snu.ac.kr.*
- *Jinwook Seo is with Seoul National University, E-Mail: jseo@snu.ac.kr.*
- *Byonghyo Shim is with Korea University, E-Mail: bshim@korea.ac.kr.*
- *Yeong-Gil Shin is with Seoul National University, E-Mail: yshin@snu.ac.kr.*
- *Bohyoung Kim is a corresponding author with Seoul National University, E-Mail: bhkim@cse.snu.ac.kr.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

when the sampling rate is not sufficient, aliasing artifacts such as moiré patterns are often introduced.

To solve this problem, the pre-integration [5] was proposed, and has been widely used [6-8]. This method integrates the transfer function *a priori* and stores every possible integrated opacity values in an additional table. In its implementation, intensities and transfer function values are discretized and the accumulated sum is then stored in an additional 2D table instead of a mathematical integral. This pre-integrated transfer function exhibits a remarkable improvement in the image quality when the transfer function has high-frequencies; thus, it is adopted in most of today's volume rendering frameworks. However, it brings in new artifacts due to the discretization of two scalar values of the intensity and transfer function value into a 2D lookup table. In addition, a quality improvement is not expected when the transfer function has a smooth transition.

Knoll [3] proposed a peak finding approach which provides high-quality rendering when the transfer function has sharp features such as Dirac impulses. The Dirac-impulse transfer function for an iso-surface classification has been addressed neither by the post-classification nor pre-integration. Peak finding attempts to find peaks, if any exists, within a ray segment. It contributes the peak intensity instead of the integrated value. This yields viable iso-value classification to be specified within the conventional DVRs, while not requiring a significant increase in the rendering cost. However, if there is no peak or if the transfer function is smooth, it yields the same quality rendering as conventional DVRs.

The two aforementioned works are used to reduce artifacts arising from high frequencies in the transfer function. As discussed earlier, the reconstruction filter function used in sampling the data in ray-casting DVRs also affects the final rendering quality significantly. Higher-order filters provide image quality that is superior to that by lower-order filters [9]. However, they require many more accesses to volume data, yielding a considerable increase in the computational cost. Theoretically, a third-order reconstruction filter in a three-dimensional case requires 64 neighbour voxels around a sampling position for data reconstruction, compared to 8 neighbour voxels in a first-order reconstruction filter (i.e., the linear interpolation function).

These reconstructions are all performed by convolution-based filtering in the spatial domain, in which filters of narrow support are applied for an efficient reconstruction. However, convolution with a filter with wider support provides better reconstruction. Several techniques have been introduced in volume-rendering applications [10-11] in which the reconstruction was performed using convolution filters with wide support in the frequency domain. They clearly demonstrated superior quality images compared to spatial-domain methods; however, all of the techniques were developed for global reconstruction and therefore did not support efficient local resampling.

Another approach to improve the reconstruction accuracy is to combine the advantages of spatial-domain and frequency-domain methods. To this end, several researchers have proposed the decomposition of the reconstruction process into discrete pre-filtering (implemented in the frequency domain) and continuous post-filtering or resampling (implemented in the spatial domain) [12-15]. This two-step framework has led to viable research on a variety of post-filters [15-17]. By exploiting the pre-filtering step, this framework provided a better approximation of the ideal reconstruction filter (i.e., the *sinc* filter).

3 Volume Ray-Casting using Virtual Samplings

3.1 Virtual Samplings from an Interpolating Cubic Spline Curve

High-quality rendering requires high-quality derivative reconstruction as well as high-quality data reconstruction. A third-order reconstruction filter guarantees the reconstruction of continuous first-order and second-order derivatives, resulting in the smooth reconstruction of implicit surface curvatures.

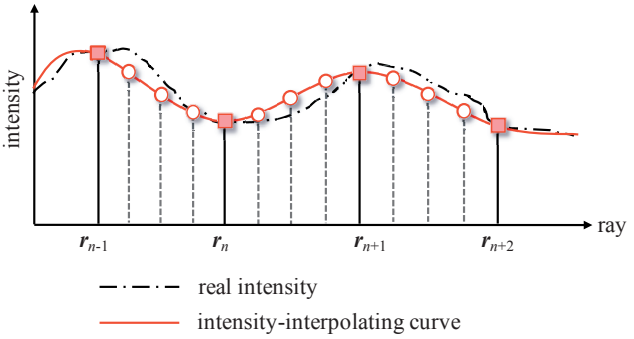


Fig. 1. Samplings at a uniform interval of unit-voxel size (solid vertical lines and red filled rectangles) and virtual samplings at a one-fourth interval from an interpolated curve (dashed vertical lines and hollow red circles)

Sigg and Hadwiger [18] proposed fast cubic filtering which evaluates a tricubic filter using only eight trilinear texture fetches. This tricubic texture filtering on GPUs provides superior image quality at a greatly improved speed over conventional tricubic convolution. However, when the combined frequency of the volume and transfer function is high, it is not very useful. Choosing a higher sampling rate is rather a viable remedy.

This section describes an approach to increase the sampling rate effectively without directly increasing the computational cost. The overall ray-casting framework is identical to the conventional framework except that more samplings occur within each ray segment in an efficient way. As a ray emitting from an image pixel traverses the volume, intensity values are sampled by the tricubic texture filtering and the transfer function is then evaluated with the sampled intensities. This sampling process is performed at a uniform unit-voxel size interval.

In a given ray segment, we derive a cubic function interpolating the two intensities sampled at the segment endpoints (Fig. 1). The intensities at any position within the ray segment are then calculated from the derived cubic function, which is referred to as '*virtual sampling.*' For ease of explanation, we refer to the type of sampling using the cubic texture filtering described earlier as '*true sampling.*' Any number of virtual samplings can be carried out within a given ray segment. To elaborate on this idea, we restrict the following description to a case in which virtual sampling is performed at one-fourth the true sampling interval, thus increasing the sampling rate by a factor of 4.

As a cubic function interpolating the *truly* sampled intensities, natural cubic splines, which are interpolation curves with C^2-continuity, can be used. However, they do not allow for 'local control,' which means that a spline curve is formulated using all of the input control points (e.g., the intensity values in our case). This makes it difficult to adopt the acceleration technique of early ray termination because all of the intensities should be sampled to construct a spline curve. Thus, we select cardinal splines which are interpolation curves defined by piecewise cubic polynomials. They support the local control property while preserving C^1-continuity across the curve sections. Among the various cardinal splines, we adopt the Catmull-Rom spline which is popularly used mainly because it is relatively easy to compute and because it guarantees that the tangents of the generated curve are continuous over multiple sections.

The Catmull-Rom spline is defined by interpolating piecewise cubic polynomials which are fitted to pass through the control points. Each polynomial section is completely specified with four consecutive control-points (e.g., four consecutive truly sampled intensities, or four red filled rectangles, as shown in Fig. 1). The

middle two control points are the section endpoints, and the other two points are used in the calculation of the endpoint slopes.

The following conditions are assumed:

$\boldsymbol{r}$: a given ray
$\boldsymbol{r}_n$: the n-th sampling position on $\boldsymbol{r}$
$\boldsymbol{I}_n$: an intensity sampled at $\boldsymbol{r}_n$
$\boldsymbol{b}$: a set of control points
$B_n(t)$: Catmull-Rom spline blending functions.

A set of control points is written as

$$\begin{aligned}\boldsymbol{b} &= \{b_{-1}, b_0, b_1, b_2\} \\ &= \{(\boldsymbol{r}_{n-1}, \boldsymbol{I}_{n-1}), (\boldsymbol{r}_n, \boldsymbol{I}_n), (\boldsymbol{r}_{n+1}, \boldsymbol{I}_{n+1}), (\boldsymbol{r}_{n+2}, \boldsymbol{I}_{n+2})\}. \end{aligned} \quad (1)$$

The Catmull-Rom spline polynomial function for the section between b_0 and b_1, $\boldsymbol{C}(t)$, can be regarded as one-dimensional interpolation in which b_i represents the truly sampled intensities (i.e., $\boldsymbol{I}_i$); thus, it is written as

$$\boldsymbol{C}(t) = I_{-1}\cdot B_{-1}(t) + I_0\cdot B_0(t) + I_1\cdot B_1(t) + I_2\cdot B_2(t), \qquad 0 \le t \le 1 \quad (2)$$

, where $B_{-1}(t) = 0.5(-t^3 + 2t^2 - t)$, $B_0(t) = 0.5(3t^3 - 5t^2 + 2)$,
$B_1(t) = 0.5(-3t^3 + 4t^2 + t)$, and $B_2(t) = 0.5(t^3 - t^2)$.

The virtual sampling at one-fourth the true sampling interval can be done simply by evaluating the polynomial, eq. (2), with $t = 0.25$, 0.5, and 0.75. These polynomial evaluations are merely arithmetic operations which run very fast on GPUs. Through these virtual samplings based on arithmetic operations, the sampling rate efficiently increases four times compared to when the volume is sampled at the unit-voxel size interval.

In volume rendering, derivatives (gradients) are required for shading. When performing the true sampling at a sampling position, a gradient vector at the position is calculated based on the central difference scheme. Although the central difference is one of simple gradient calculation schemes, it requires as many as six more intensity samplings. These additional samplings can be also performed by tricubic texture filtering on a GPU-programmed DVR, but this increases the rendering cost considerably.

In the proposed method, the gradient vectors at the virtual sampling positions in a ray segment are also calculated via polynomial evaluations, in which the polynomial functions being used are the newly derived Catmull-Rom splines to interpolate the two gradient vectors at the ray segment endpoints. By exploiting SIMD operations on RGBA channels supported by GPUs, the intensity and gradient vectors at a virtual sampling position are simultaneously calculated in a single arithmetic operation.

3.2 Refinement of Min Max Blocks

The proposed method works in combination with the two commonly used acceleration techniques [19] of early ray termination (ERT), in which the ray traversal is terminated when the accumulated opacity becomes sufficiently opaque, and empty space skipping (ESS), in which empty spaces are skipped using an additional empty-space encoding data structure. ERT can be easily integrated into any type of ray-casting rendering framework without specific refinements. For ESS, we choose a 3D raster structure which encodes the volume data using disjoint blocks with constant size. In this structure, each block stores the minimum and maximum scalar values contained in this block.

Min max blocks can be constructed by simply comparing the scalar values of all the voxels in the volume. To use the min max blocks in ESS, it should be guaranteed that the sampled values in a given block during ray traversal are in the range of min and max values of that block. Otherwise, blocks may be inappropriately skipped. In a case in which the min max values of a block do NOT overlap with an intensity range of the transfer function but the sampled intensities in the block during ray traversal DO overlap with the intensity range, such a block should not be skipped but should instead be traversed to contribute to the transfer function evaluation. However, using min max blocks built on original voxel values causes the ray to skip such a block, yielding an inaccurate volume rendering result.

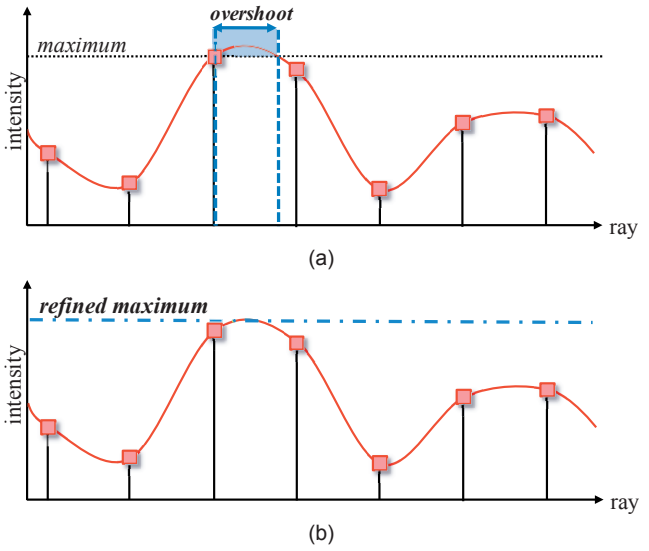


Fig. 2. (a) The case that the values evaluated from an interpolating cubic curve function exceed the maximum intensity in the original volume data. (b) Refinement of the maximum intensity.

In the proposed method, the virtual sampling is carried out using a cubic interpolation function, the Catmull-Rom spline curve. The cubic interpolation function can give rise to values outside the range of the input data ('overshoot') [4] (Fig. 2 (a)). Consequently, it is necessary to refine the min max blocks in such a way the min max values in each block are adjusted to support accurate empty space skipping (Fig. 2(b)).

The min max blocks are first created based on the original voxel intensities in the volume. As the Catmull-Rom spline is used to interpolate the truly sampled intensities, the upper and lower bounds of the virtually sampled intensities within a given block are derived using the curve polynomial function. The bound values are then used as the new min and max intensities of that block.

The bound values are derived as follows. This derivation assumes a block with the minimum and maximum values of m and M, respectively. A Catmull-Rom spline curve which is formulated from four consecutive truly sampled intensities (I_{-1}, I_0, I_1, and I_2) to pass through the middle two intensities(I_0 and I_1) is already written in eq. (2).

First, we will show that every Catmull-Rom spline curve is bounded by $\boldsymbol{C}_{extrm}$, a Catmull-Rom spline curve for an extreme case. The extreme case in which the interpolating Catmull-Rom spline curve exceeds the maximum intensity to the greatest extent is when the middle two intensities have the value of M and the other two intensities have the value of m (Fig. 3 (a)). The curve for this extreme case, $\boldsymbol{C}_{extrm}$, is written as

$$\boldsymbol{C}_{extrm}(t) = B_{-1}(t)\cdot m + B_0(t)\cdot M + B_1(t)\cdot M + B_2(t)\cdot m\ . \quad (3)$$

Without a loss of generalizability, Catmull-Rom spline curves which are available in a block with min max values of m and M (Fig. 3 (b)) can be defined as

$$\begin{aligned}\boldsymbol{C}(t) &= B_{-1}(t)\cdot I'_{-1} + B_0(t)\cdot I'_0 + B_1(t)\cdot I'_1 + B_2(t)\cdot I'_2 \\ &= B_{-1}(t)\cdot(m+\Delta I_{-1}) + B_0(t)\cdot(M+\Delta I_0) \\ &\quad + B_1(t)\cdot(M+\Delta I_1) + B_2(t)\cdot(m+\Delta I_2) \\ &= B_{-1}(t)\cdot m + B_0(t)\cdot M + B_1(t)\cdot M + B_2(t)\cdot m \\ &\quad + B_{-1}(t)\cdot\Delta I_{-1} + B_0(t)\cdot\Delta I_0 + B_1(t)\cdot\Delta I_1 + B_2(t)\cdot\Delta I_2 \end{aligned} \quad (4)$$

, where $\Delta I_n = I'_n - I_n$ (n = -1, 0, 1, and 2), and $I_{-1} = I_2 = m$ and $I_0 = I_1 = M$.

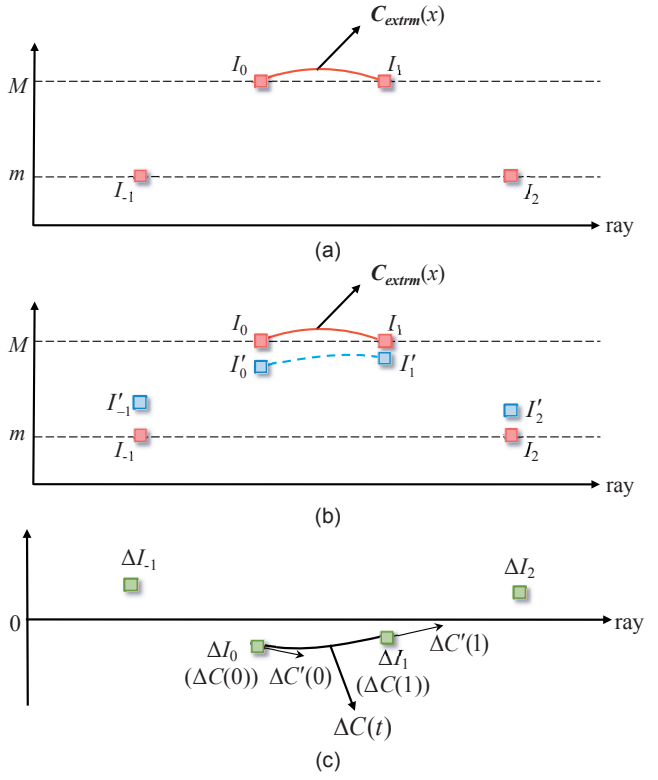


Fig. 3. Refinement of min max blocks and its justification. (a) The extreme Catmull-Rom spline curve with middle two intensities (I_0 and I_1) having the value of M and the other two intensities (I_{-1} and I_2) having the value of m. (b) Catmull-Rom spline curve in a block with min max values of m and M. (c) Catmull-Rom spline curve, $\Delta\boldsymbol{C}(t)$, specified by four intensity differences at the four true-sampling positions.

Eq. (4) is rewritten as

$$\boldsymbol{C}(t) = \boldsymbol{C}_{extrm}(t) + \Delta\boldsymbol{C}(t) \tag{5}$$

by introducing a new function defined as

$$\Delta\boldsymbol{C}(t) = B_{-1}(t)\cdot\Delta I_{-1} + B_0(t)\cdot\Delta I_0 + B_1(t)\cdot\Delta I_1 + B_2(t)\cdot\Delta I_2$$

, in which $0 \le \Delta I_{-1} \le M\text{-}m$, $m\text{-}M \le \Delta I_0 \le 0$, $m\text{-}M \le \Delta I_1 \le 0$, and $0 \le \Delta I_2 \le M\text{-}m$.

Eq. (5) indicates that the Catmull-Rom spline curves in a block with min max values of m and M can be defined as the addition of the extreme curve, $\boldsymbol{C}_{extrm}$ and another Catmull-Rom spline curve specified by the four intensity differences from the min max values at the four true-sampling positions (Fig. 3 (c)).

Now, we will prove $\Delta\boldsymbol{C}(t) \le 0$ and then $\boldsymbol{C}(t) \le \boldsymbol{C}_{extrm}(t)$. $\Delta\boldsymbol{C}(t)$ and $\Delta\boldsymbol{C}'(t)$ with $t = 0$ and 1 are evaluated as follows:

$$\begin{aligned}&\Delta\boldsymbol{C}(0) = \Delta I_0 \le 0, \qquad \Delta\boldsymbol{C}(1) = \Delta I_1 \le 0,\\ &\Delta\boldsymbol{C}'(0) = 0.5(\Delta I_1 - \Delta I_{-1}) \le 0, \quad \Delta\boldsymbol{C}'(1) = 0.5(\Delta I_2 - \Delta I_0) \ge 0\end{aligned} \tag{6}$$

, since $\Delta I_0 \le 0$, $\Delta I_1 \le 0$, $\Delta I_{-1} \ge 0$, and $\Delta I_2 \ge 0$ in eq. (5).

Eq. (6) indicates that there exists, if any, local minima of $\Delta\boldsymbol{C}(t)$ between $t = 0$ and $t = 1$ because the Catmull-Rom spline, $\Delta\boldsymbol{C}(t)$, is a cubic polynomial. Therefore, it follows that $\Delta\boldsymbol{C}(t) \le 0$; thus, $\boldsymbol{C}(t) \le \boldsymbol{C}_{extrm}(t)$, implying that $\boldsymbol{C}(t)$ is bounded by $\boldsymbol{C}_{extrm}(t)$.

Next, we accordingly adjust the original max value of M to a new maximum M_{new}, which is derived from the bounding curve $\boldsymbol{C}_{extrm}(t)$. $\boldsymbol{C}_{extrm}(t)$ in eq. (3) can be rewritten as

$$\begin{aligned}\boldsymbol{C}_{extrm}(t) &= B_{-1}(t)\cdot m + B_0(t)\cdot M + B_1(t)\cdot M + B_2(t)\cdot m\\ &= 0.5(-t^3 + 2t^2 - t)\cdot m + 0.5(3t^3 - 5t^2 + 2)\cdot M\\ &\quad + 0.5(-3t^3 + 4t^2 + t)\cdot M + 0.5(t^3 - t^2)\cdot m\\ &= -0.5(M-m)\cdot t^2 + 0.5(M-m)\cdot t + M\\ &= -0.5(M-m)\cdot t(t-1) + M.\end{aligned} \tag{7}$$

Eq. (7) is a second-order function with the maximum value at $t = 0.5$, given that $M - m > 0$. The maximum value of eq. (7) is calculated as

$$\boldsymbol{C}_{extrm}(0.5) = M + (M - m)/8. \tag{8}$$

We finally set the value of $M + (M - m)/8$ to the new maximum M_{new}. The max value of a block with min max values of m and M is then adjusted to M_{new}.

Likewise, the new minimum, m_{new}, is calculated from another extreme case in which the middle two intensities have the value of m and the other two have the value of M. m_{new} is determined to have the value of $m - (M - m)/8$, and the min value of the block is then adjusted to m_{new}.

4 Implementation

The proposed algorithm can be integrated easily into conventional ray-casting DVR frameworks. The generation of min max blocks and their refinement are simultaneously carried out in a pre-processing step. In the ray casting, ESS is performed using the refined min max blocks.

Conventional ray-casting is typically implemented on commodity GPUs in a single pass, in which the final pixel color is determined through a single loop in the fragment program. To execute the proposed virtual samplings, another loop is required that derives a Catmull-Rom spline curve interpolating the truly sampled intensities and then evaluates the derived curve function at three virtual sampling positions within a ray segment. The true samplings are done by tricubic B-spline texture filtering [18]. As mentioned earlier, in the virtual sampling, the intensity and gradient vector are simultaneously evaluated from corresponding Catmull-Rom splines using a SIMD operation on RGBA channels supported by GPUs.

In ray-casting DVRs, opacities evaluated from the transfer function need to be adjusted when the sampling step size changes. With the assumption that an evaluated opacity at a sampling position with a given sampling step size is α, if we reduce the sampling step size to $1/n$ by increasing the sampling rate by a factor of n, an adjusted opacity, α_n, with this reduced step size is calculated to satisfy the following equation:

$$\alpha = \textstyle\sum_{i=0}^{n-1}(1-\alpha_n)^i\alpha_n. \tag{9}$$

As it is complicated to solve α_n in eq. (9) for any positive integer n, we restrict n to have the power of 2. For this case, α_n ($n = 2^m$) can be calculated easily via recursion. As an example,

$$\begin{aligned}&\alpha = \alpha_1 = \alpha_2 + (1-\alpha_2)\alpha_2 = 2\alpha_2 - \alpha_2^2\\ &\alpha_2 = 1 - \sqrt{1-\alpha_1} \qquad (\because \alpha_2 \ge 0).\end{aligned} \tag{10}$$

Considering that the sampling step size for α_{2^m} is half the step size for $\alpha_{2^{m-1}}$, eq. (10) is directly generalized to a recurrent relation written as

$$\alpha_{2^m} = 1 - \sqrt{1-\alpha_{2^{m-1}}} \qquad (m \ge 1). \tag{11}$$

The proposed method requires 4 consecutive truly sampled values to evaluate virtual samplings within a ray segment. In its implementation, while proceeding one ray step, the method conducts only one new true sampling because three true sampling values for the previous ray segment can be used for the current ray segment.

5 Experimental Results

We tested the proposed method on an Intel i7 desktop system with a 2.67 GHz quad-core processor and 12 GB of memory. The system was also equipped with a NVIDIA GTX 280 GPU with 1 GB of memory. HLSL Shader Model 4.0 with DirectX 10.0 was used.

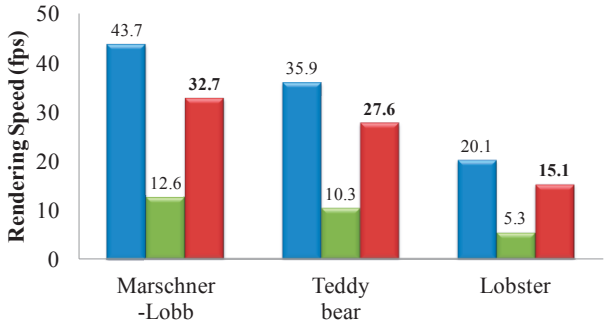


Fig. 4. Computational performance results for three datasets. Three bars for each dataset represent the three compared methods, Cubic x1, Cubic x4, and our method, from left to right.

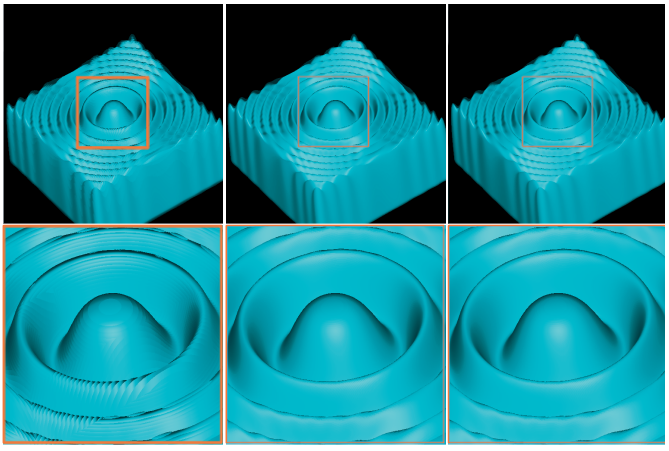
5.1 Performance Improvement by Virtual Samplings

To demonstrate the effectiveness of the proposed method, we comparatively evaluated the rendering performance of three types of ray-casting DVR: cubic sampling at a unit-voxel size interval (referred as 'Cubic x1'), cubic sampling with a sampling rate 4x higher than the previous rate (referred as 'Cubic x4'), and the proposed method with cubic sampling at a unit-voxel size interval and three virtual samplings in a ray segment (referred as 'Cubic x1 + VS x3'). In Cubic x4, all of the samplings are true samplings by tricubic texture filtering and the gradients are calculated based on the central difference scheme at all of the sampling positions. The proposed method performs as many samplings as Cubic x4.

The test datasets include Marschner-Lobb data [20], a teddy bear [21], and a lobster [22]. Fig. 4 presents the rendering speed in fps for these three datasets, averaged over multiple tests. Fig.s 5, 6, and 7 show the rendered images of the datasets rendered by the three methods. The image resolution is 512×512 in all cases.

As shown in all the figures and in the speed graph, Cubic x1 exhibits the most amount of artifact while providing the fastest rendering speed. The proposed method offers rendering as good as Cubic x4 but with a faster rendering speed by 2.5x to 3.2x. It is trivial that Cubic x1 shows the fastest but most degraded rendering considering that its sampling rate is only one fourth that of the other two methods. In contrast, though our method has a sampling rate identical to that of Cubic x4, it renders images of similar quality at a greatly improved rendering speed. This is attributed to the fact that the virtual samplings introduced in our method are carried out simply via arithmetic evaluations.

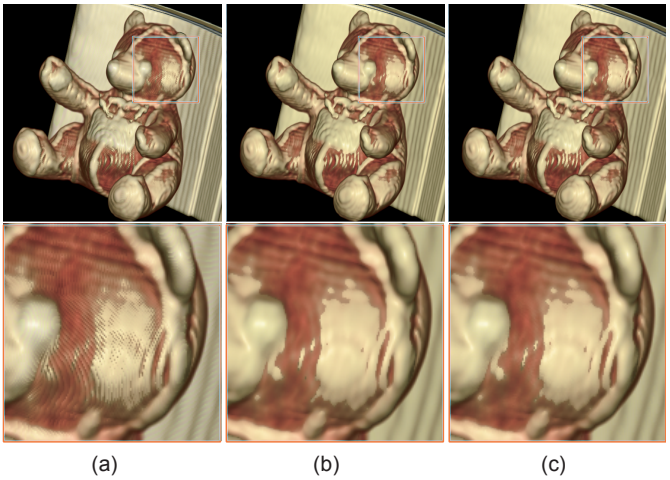
Interestingly, Cubic x4 and the proposed method yielded visually the same rendering results in all the datasets. To compare the reconstruction performance between all true samplings in Cubic x4 and the true and then virtual samplings in our method, we subtracted the images rendered by the two methods (Fig. 8). As shown in Fig. 8, the two methods exhibit almost no mathematical difference in terms of their pixel values, except partly in the top of the rings in Marschner-Lobb data and in the cheek and belly in the teddy bear, which is difficult to perceive in the small subtraction images presented. During the implementation, the true sampling is done using a B-spline curve filter, one of the approximation curves, and the virtual sampling is done using the Catmull-Rom spline, one of the interpolation curves. Our results demonstrate that the intensity calculated from a Catmull-Rom spline curve passing through the B-spline approximated intensity values is likely to be nearly identical to the intensity approximated directly by the B-spline curve at the same sampling position.



(a) (b) (c)

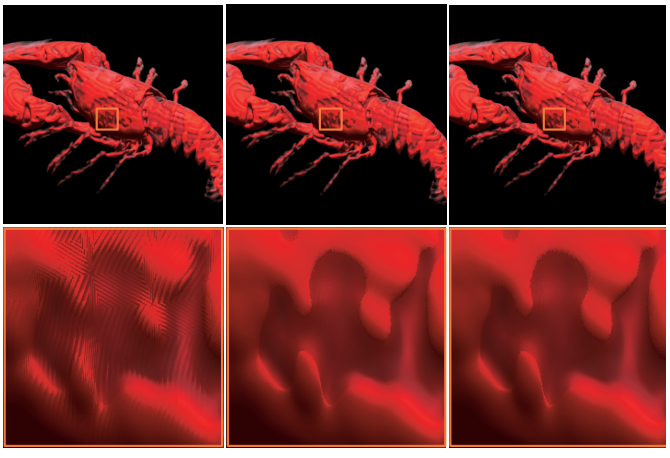
Fig. 5. Marschner-Lobb data (41×41×41) rendered using (a) Cubic x1, (b) Cubic x4, and (c) the proposed method. Lower row shows close-up images of the orange rectangular region in the upper row.

We conducted another experiment to compare the proposed method to the pre-integration approach. As discussed earlier, the pre-integration approach was proposed with the intent to attenuate artifacts when the transfer function has high frequencies. It offers improved quality over the post-classification approach without increasing the sampling rate. The above three methods of Cubic x1, Cubic x4, and the proposed method adopt the post-classification which evaluates the transfer function after intensity sampling. As shown in Fig. 9 with a transfer function with relatively high frequencies, the pre-integration approach effectively removes artifacts of moiré patterns in the back head compared to Cubic x1



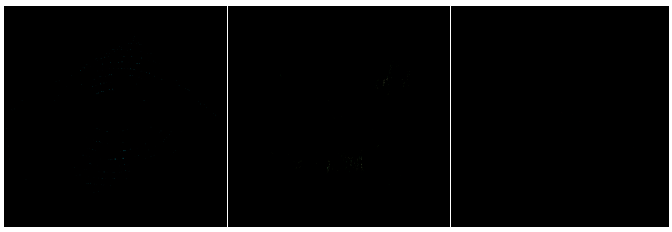
(a) (b) (c)

Fig. 6. Teddy bear (128×128×62) rendered using (a) Cubic x1, (b) Cubic x4, and (c) the proposed method. Lower row shows close-up images of the orange rectangular region in the upper row.



(a) (b) (c)

Fig. 7. Lobster (301×324×56) rendered using (a) Cubic x1, (b) Cubic x4, and (c) the proposed method. Lower row shows close-up images of the orange rectangular region in the upper row.



(a) (b) (c)

Fig. 8. Subtraction image between the rendered images by Cubic x4 and the proposed method. (a) Marschner-Lobb data, (b) Teddy bear, and (c) Lobster.

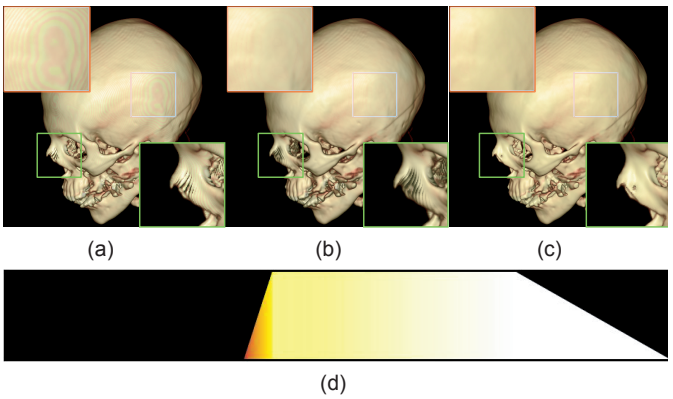
(see the orange rectangular region in Fig. 9(b)). However, it experiences the information loss in the nose due to its insufficient samplings (see the green rectangular region in Fig. 9(b)). By contrast, the proposed method reduces moiré patterns in the back head effectively as well as preserves the shape of the nose. Fig. 10 presents another comparison using a transfer function with a smooth transition. As shown in Fig. 10, Cubic x1 and the pre-integration approach exhibit similar quality renderings with a similar amount of artifact, whereas our method offers superior rendering quality with markedly attenuated artifact in the back head. This result demonstrates that when the transfer function is relatively smooth, an effective reduction of artifacts can be achieved by increasing the sampling rate, rather than by using the pre-integrated transfer function.

Our method applies a third-order filtering (i.e., cubic B-spline filtering) for the true sampling at a unit-voxel size interval, derives a Catmull-Rom spline cubic function interpolating the truly sampled intensities, and then evaluates the virtual samplings from the derived cubic function. The true and virtual samplings can be carried out using lower-order filter functions. We assessed how the order of the true sampling filter and/or the virtual sampling function affects the final rendering quality.

Fig. 11 shows renderings of the nasal cavity in Fig. 9 at an image resolution of 512^2. The upper row of Fig. 11 shows the results using Cubic x1, Cubic x4, our method (Cubic x1 + VS x3), and our method with twice more number of virtual samplings (Cubic x1 + VS x7). The lower row shows the results using Linear x1 (linear sampling at a unit-voxel interval), Linear x4 (linear sampling with a sampling rate 4x higher than the previous rate), Linear x1 + VS x3 (linear sampling at a unit-voxel size interval and three virtual samplings using the cubic Catmull-Rom spline curve) and Cubic x1 + LinearVS x3 (cubic sampling at a unit-voxel size interval and three virtual samplings using the linear interpolation function). As shown in the figure, Cubic x1 shows much more amount of artifact than Cubic x4 and Cubic x1 + VS x3 (see the green arrow marks in Fig. 11(a)), and Cubic x4 and Cubic x1 + VS x3 offer visually the same rendering, which is consistent with Fig.s. 5, 6, and 7. All the three methods using the linear true sampling, Linear x1, Linear x4, and Linear x1 + VS x3, demonstrate more artifacts than Cubic x4 and Cubic x1 + VS x3. The last method of Cubic x1 + LinearVS x3 shows better rendering quality than the three methods using the linear true sampling; however, it still exhibits more artifacts than Cubic x4 and Cubic x1 + VS x3. These experiments reveal that the orders of both the true and virtual sampling filters significantly affect the final rendering quality. True and/or virtual samplings with a lower-order filter experiences considerable information loss even with an increased sampling rate, and thus both of the two samplings are recommended to be performed using a higher-order filter function for the reconstruction of a smooth and accurate surface.

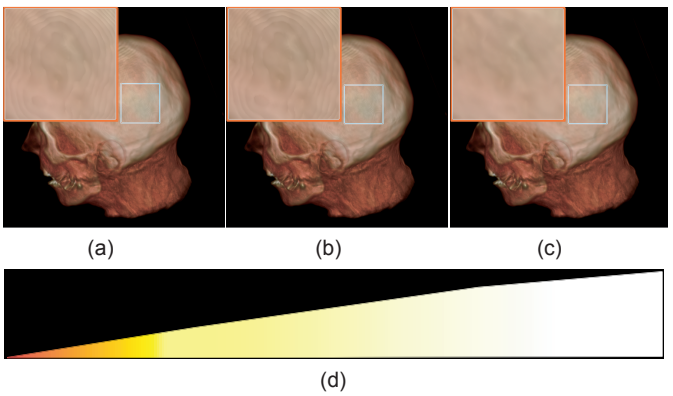
Table 1. Computational performance results for nasal cavity rendering

Sampling mode	Rendering Speed (fps)
Cubic x1	9.1
Cubic x4	2.5
Cubic x1 + VS x3	7.9
Cubic x1 + VS x7	7.1



(a) (b) (c) (d)

Fig. 9. Head (256×256×225) rendered using (a) Cubic x1, (b) pre-integration approach, and (c) the proposed method. The transfer function being applied is presented in (d). Close-up images of the orange and green rectangular regions are presented in the top-left and bottom-right corners in the image, respectively.



(a) (b) (c) (d)

Fig. 10. Head (256×256×225) rendered using (a) Cubic x1, (b) pre-integration approach, and (c) the proposed method. The transfer function being applied is presented in (d). Close-up images of the orange rectangular region are presented in the top-left corner in the image.

To demonstrate the computational simplicity of the arithmetic virtual samplings in our method, we measure the rendering speeds for the nasal cavity renderings by the four methods in the upper row in Fig.11, averaged over multiple tests (Table 1). Cubic x1 shows the fastest rendering. The second fast rendering, Cubic x1 + VS x3, delivers close to a 3.2x rendering speed-up over Cubic x4. When a rendered object has fine details or a complicated shape, as does a nasal cavity, the ray needs to traverse more number of min max blocks, travelling a long distance. In this case, our method benefits from the virtual samplings more than otherwise, providing a greater improvement in the rendering performance. This is also the case when the transfer function has a very smooth transition as in Fig. 10 (3.2x speed-up over Cubic x4). The last method in Table 1, Cubic x1 + VS x7, which can be regarded as doubling the sampling rate of Cubic x1 + VS x3, exhibits a slight degradation (10.1%) in rendering speed from Cubic x1 + VS x3, indicating that increasing the sampling rate by double, quadruple, or more can be achieved very efficiently with our method. The rendering quality of Cubic x1 + VS x3 and Cubic x1 + VS x7 is visually the same (Fig. 11(c) and (d)).

5.2 Quantitative Error Analysis

Since our discussion thus far focuses on subjective evaluation of the rendering quality, it is of interest to investigate an objective metric measuring the rendering error. Towards this end, we compute the root mean square error (rMSE) and the peak signal-to-noise ratio (PSNR) of the rendered images of fish bone in Fig. 12.

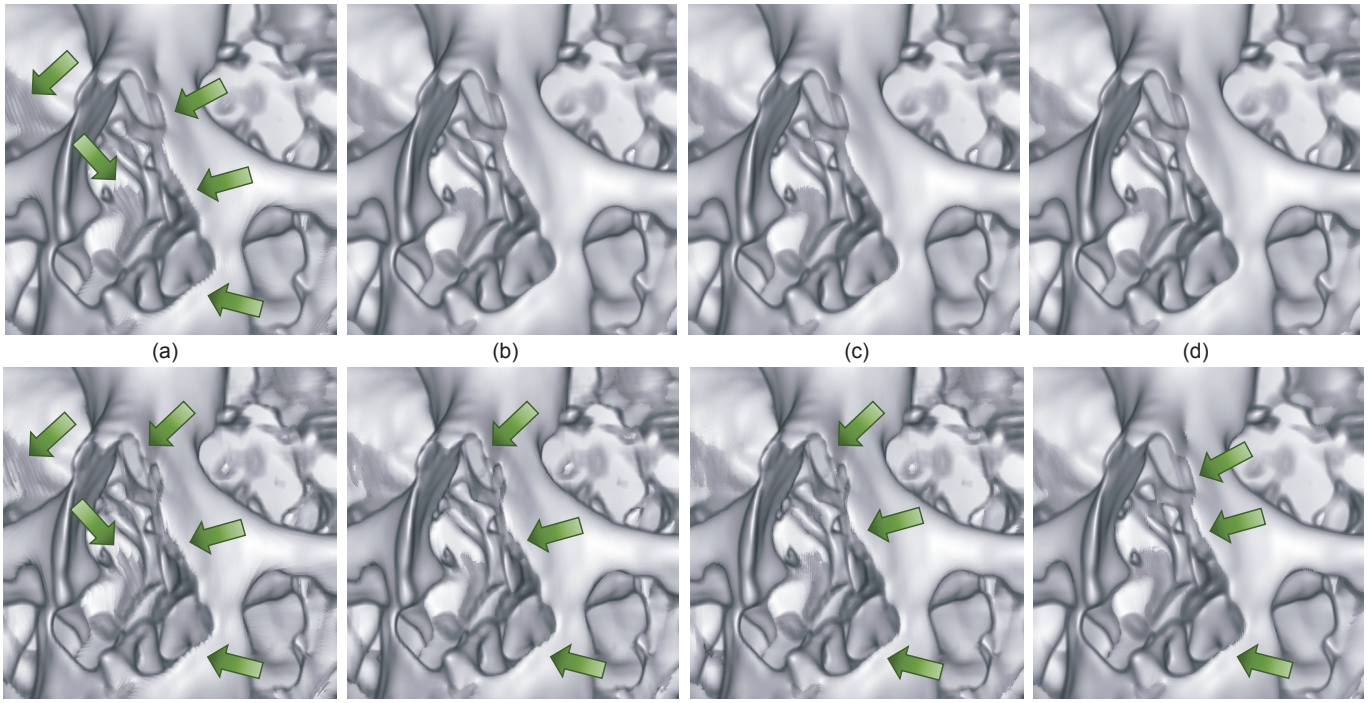


Fig. 11. Nasal cavity from the skull in Fig. 9 rendered using (a) Cubic x1, (b) Cubic x4, (c) the proposed method, Cubic x1 + VS x3, (d) Cubic x1 + VS x7, (e) Linear x1, (f) Linear x4, (g) Linear x1 + VS x3, and (h) Cubic x1 + LinearVS x3.

Compared with previous test datasets, fish bone has far more fine details (i.e., high frequency components) so that one can expect that the information loss of the proposed method (mainly due to the Catmull-Rom interpolation) over the Cubic x4 may be pronounced. In this experiment, rMSEs between the sampled intensities by Cubic x4 and the proposed method are 2.32 and 2.49 for the full-body (the upper row in Fig. 12) and the tail part (the lower row in Fig. 12), respectively. Using these results together with the fact that intensity level ranges from 0 to 10,000, PSNRs for the full-body and the tail part are 72.7 dB and 72.1 dB, respectively. Noting that in general it is hard to observe the difference between two images for PSNR greater than 30 dB [23], one can see that the proposed method is comparable to the Cubic x4 in an objective perspective as well. The frequency responses shown in Fig. 13 help to give a better idea of why the performance of the proposed algorithm is comparable to that of Cubic x4. We clearly observe that the difference between two methods for most of passband region (mainlobe) is imperceptible. Although the difference in the stopband (sidelobes) is non-trivial, the impact of difference in stopband on performance is negligible in practice as the attenuation at these frequencies is already below -50 dB.

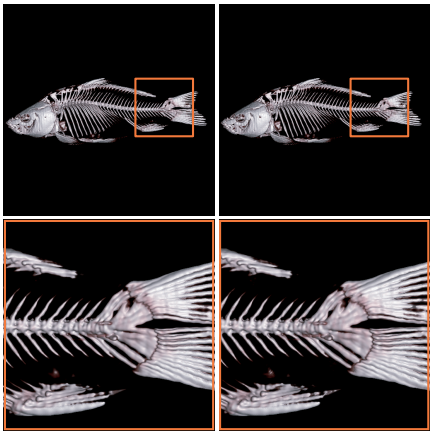


Fig. 12. Carp (256×256×512) rendered using (a) Cubic x4 and (b) the proposed method. Lower row shows close-up images of the orange rectangular region in the upper row.

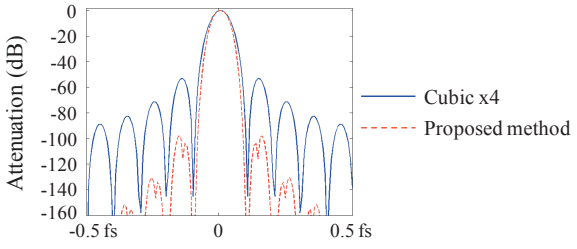


Fig. 13. Frequency responses of Cubic x4 and the proposed method.

5.3 Effect of Refinement of Min Max Blocks

In the proposed method, the min max blocks for ESS are refined to

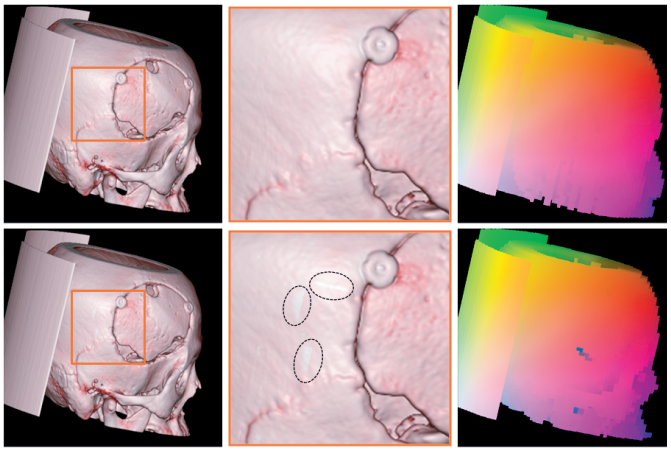


Fig. 14. Skull (512×512×300) rendered using the proposed method with (upper row) and without (lower row) the refinement of min max blocks. The middle column shows close-up images of the orange rectangular region in the first column. The third column shows a block-entry position of each ray.

ensure that all of the virtually sampled intensities in a given block during the ray traversal are in the range of the min and max values of that block. We conducted an experiment to demonstrate the necessity of this refinement.

Fig. 14 presents the rendering results of a skull at an image resolution of 512^2 using our method with and without min max block refinement. Without the refinement, the rendering yields streak artifacts (see the dotted circles in the lower row). The artifacts arise due to the skipping of the blocks in which the min max values do not overlap with the transfer function range but where the virtually sampled intensities overlap with the range. Those blocks should not be skipped but should instead be traversed to contribute to the transfer function evaluation. Refinement of the min max blocks can avoid such inappropriate block skipping, providing accurate volume rendering. The right column in the figure shows the block-entry positions from which each ray starts to sample. Through the refinement of the min max blocks, many rays have their block-entry positions adjusted closer to the image plane compared to a case without the refinement.

6 Conclusion

We have proposed a fast high-quality volume ray-casting algorithm that features *virtual* samplings. While a ray traverses the volume, the intensity values are uniformly sampled using tricubic texture filtering. Additionally, virtual samplings are carried out, within a ray segment, from a Catmull-Rom spline curve interpolating those uniformly sampled intensities. These virtual samplings are performed at any position within a ray segment by evaluating the polynomial function of the Catmull-Rom spline curve, which are simple arithmetic operations. This algorithm increases the sampling rate effectively, delivering visually identical rendering to that of a conventional ray-casting algorithm using cubic filtering at the same sampling rate, while achieving a rendering speed-up of 2.5x to 3.3x.

As the virtual samplings in the proposed method can introduce intensities that are out of the range of the input volume data, the min max blocks for ESS are refined to assure that the virtually sampled intensities are in the range of the refined min max values. This refinement is conducted considering the extreme case when the Catmull-Rom spline curve exceeds the range of the original min and max values to the greatest extent. This conservative refinement may generate loosely-bound min max values, increasing the blocks to be traversed and consequently incurring some degradation in the rendering performance. However, it is likely an inevitable limitation, considering that the calculation of tightly-bound min max values is not possible prior to completing the ray traversal.

Our algorithm adopts a Catmull-Rom spline as an interpolation cubic function during the virtual sampling. Catmull-Rom splines are frequently used due to their local control property and computational simplicity. Catmull-Rom splines are C^1-continuous cubic function. It is likely that cubic splines with higher continuity, such as natural cubic splines with C^2-continuity, provide better rendering quality. However, the local control property of Catmull-Rom splines is a non-negotiable property, considering the combined use of the proposed algorithm with the acceleration technique of ERT. In addition, based on our experimental results, Catmull-Rom splines provide sufficiently good reconstruction of the intensities and gradients during the virtual sampling process.

In the future, we are planning to access the feasibility of using our algorithm in combination with the pre-filtered reconstruction scheme. Pre-filtered reconstruction consists of the two steps of discrete pre-filtering and continuous post-filtering. In volume rendering applications, the pre-filtering step can be carried out during the pre-processing step. Its output can be then used in a subsequent sampling process along the rays. Combined with this pre-filtering step, the algorithm proposed in this paper can produce renderings of more improved quality while embracing its advantageous rendering speed-up.

Acknowledgments

This work was supported in part by the BK 21 Project in 2010 and by Basic Science Research Program through the NRF funded by the MEST of Korea (No. 2010-0017178). This work was also supported in part by the Industrial Strategic Technology Development Program funded by the MKE of Korea (No. 10035474) and in part by Grant no. 10888 from the Seoul Research and Business Development Program. The ICT at Seoul National University provides research facilities for this study.

References

[1] H. Pfister, *et al.*, "The VolumePro real-time ray-casting system," in *SIGGRAPH '99*, 1999, pp. 251-260.

[2] J. Kruger and R. Westermann, "Acceleration techniques for GPU-based volume rendering," in *Proc. IEEE Visualization*, 2003, pp. 287-292.

[3] A. Knoll, *et al.*, "Volume ray casting with peak finding and differential sampling," *IEEE Trans. Vis. Comput. Graphics,* vol. 15, pp. 1571-1578, 2009.

[4] G. Wolberg, "Interpolation kernels," in *Digital Image Warping*, ed Los Alamitos, CA: Wiley-IEEE Computer Society Press, 1990, pp. 117-161.

[5] N. Max, *et al.*, "Area and volume coherence for efficient visualization of 3D scalar functions," in *Proc. 1990 workshop on Volume visualization*, 1990, pp. 27-33.

[6] K. Engel, *et al.*, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shaing," in *Proc. ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, New York, NY, USA, 2001, pp. 9-16.

[7] H. Kye, *et al.*, "Interactive classificiation for pre-integrated volume rendering of high-precision volume data," *Graphical Models,* vol. 70, pp. 125-132, 2008.

[8] E. Lum, *et al.*, "High-quality lighting and efficient pre-integration for volume rendering," in *Proc. VisSym '04*, 2004, pp. 25-34.

[9] K. Bjorke, "High-quality filtering," in *GPU Gems*, R. Fernando, Ed., ed Upper Saddle River, NJ: Addison-Wesley, 2004, pp. 391-415.

[10] M. Artner, *et al.*, "High-quality volume rendering with resampling in the frequency domain," in *Proc. Joint EUROGRAPHICS-IEEE VGTC Symp. Visualization (EuroVis '05)*, 2005, pp. 85-92.

[11] A. Li, *et al.*, "Methods for efficient, high quality volume resampling in the frequency domain," in *Proc. IEEE Visualization*, 2004, pp. 3-10.

[12] T. Blu, *et al.*, "Generalized interpolation: higher quality at no additional cost," in *Proc. IEEE International Conference on Image Processing*, 1999, pp. 667-671.

[13] T. Malzbender, "Fourier volume rendering," *ACM Trans. Graphics,* vol. 12, pp. 233-250, 1993.

[14] B. Csebfalvi, "An evaluation of prefiltered B-spline reconstruction for quasi-interpolation on the body-centered cubic lattice," *IEEE Trans. Vis. Comput. Graphics,* vol. 16, pp. 499-512, May/Jun 2010.

[15] B. Csebfalvi, "Prefiltered gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid," presented at the 16th IEEE Vis. 2005, 2005.

[16] P. Thevenaz, *et al.*, "Interpolation revisited," *IEEE Trans. Med. Imag.,* vol. 19, pp. 739-758, 2000.

[17] T. Blu, *et al.*, "Linear interpolation revitalized," *IEEE Trans. Image Process.,* vol. 13, pp. 710-719, 2004.

[18] C. Sigg and M. Hadwiger, "Fast third-order texture filtering," in *GPU Gems 2*, M. Pharr, Ed., ed Los Alamitos, CA: IEEE Computer Society: Addison Wesley, 2005, pp. 313-329.

[19] M. Levoy, "Efficient ray tracing of volume data," *ACM Trans. Graph.,* vol. 9, pp. 245-261, 1990.

[20] S. Marschner and R. Lobb, "An evaluation of reconstruction filters for volume rendering," in *Proc. IEEE Visualization*, 1994, pp. 100-107.

[21] K. Engel. 2001, *Data*. Available: http://www.vis.uni-stuttgart.de/~engel/pre-integrated/data.html

[22] Available: http://www.gris.uni-tuebingen.de/edu/areas/scivis/volren/datasets/datasets.html

[23] C. Lee, *et al.*, "Adaptive lossless steganographic scheme with centralized difference expansion," *Pattern Recognition,* vol. 41, pp. 2097-2106, 2008.