Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

Real-time adjustment of transfer function for Fourier volume rendering

Chang-Chieh Cheng
Yu-Tai Ching

Real-time adjustment of transfer function for Fourier volume rendering

Chang-Chieh Cheng
Yu-Tai Ching
National Chiao Tung University
Department of Computer Science
EC 444, 1001 University Road
Hsin-Chu 30010, Taiwan
E-mail: ytc@cs.nctu.edu.tw

Abstract. *Fourier volume rendering (FVR) is a volume rendering method based on the Fourier slice theorem. With an $n \times n \times n$ volume data, the FVR algorithm requires $O(n^2 \log n)$ time to generate a result. Because it requires time less than $O(n^3)$ does, FVR is preferred for designing a real-time rendering algorithm with a preprocessing step. We improve upon our previous work. We demonstrate that a B-spline is significantly more useful when designing a transfer function. To design an appropriate transfer function with a spline function, additional control points are required. However, the memory space required for the proposed method increases in linear proportion to the number of control points. We show that the set of control points can be clustered into groups, ensuring the memory required is linearly proportional to the number of groups. The proposed technique supports real-time rendering after adjusting the transfer function for FVR. © 2011 SPIE and IS&T.* [DOI: 10.1117/1.3653264]

1 Introduction

Visualization is a technique that generates images, animations, or figures, enabling users to understand and analyze a data set. Volume rendering is among the visualization techniques commonly used for scientific data or medical volume images. Volume visualization techniques can be divided into two types, surface rendering and direct volume rendering. Surface rendering requires geometric primitives to construct the isosurfaces that represent the set of points of a selected constant value. These isosurfaces are then rendered using a shading algorithm. Lorensen and Cline[1] proposed the marching cube method. In this approach, 15 primitive types of polygons were defined to create isosurfaces from a unit cube.

Unlike surface rendering, direct volume rendering produces rendered images by imitating x rays passing through an object. The 2-D projections are obtained by integrating the voxel values on the lines along the view direction. The integral is an optical model proposed by Max.[2] Among the previously proposed direct volume rendering methods, texture-based volume rendering is considered the most efficient.[3] In this method, the volume data are considered a 3-D texture. A set of proxy polygons parallel to the view plane are constructed to sample voxels from the 3-D texture. During the sampling stage, the transfer function converts each voxel to a color and opacity value to enhance the region of interest. Finally, the proxy polygons are drawn using the α blending method on the frame buffer. To improve the rendered result, the number of proxy polygons should be increased, improving the sampling rate. This also increases the rendering time.

Another direct volume rendering method is Fourier volume rendering (FVR).[4–6] FVR is based on the Fourier slice theorem. Assuming that the Fourier transform of volume data is available, the inverse Fourier transform of a slice from the frequency domain, which passes through the origin and is perpendicular to the view direction, is the projection of the volume along the view direction. This projection is the exact result of direct volume rendering. Using FVR, because only the inverse Fourier transform of a slice must be calculated, a volume of n^3 voxels can be rendered in $O(n^2 \log n)$ time, providing the Fourier transform of the volume is available. Literature regarding the FVR method is scarce. Malzbender[6] designed several filters to reduce the artifacts caused by the resampling in the frequency domain. Levoy[5] presented three shading models for FVR, including depth cueing, directional lighting, and specular reflections. Combined with Levoy's shading model, Entezari et al.[7] used the spherical harmonic function to approximate cubic illumination shading for FVR.

To enhance the region of interest, designing an appropriate transfer function is vital. Numerous practical and effective transfer functions have been proposed. For example, Engel et al.[8] proposed the preintegrated 2-D transfer function to improve the rendering quality. Kindlmann et al.[9] used the surface curvature to determine the contour thickness for non-photorealistic volume rendering. Lum and Ma[10] proposed the lighting transfer function to enhance the boundary surfaces of the region of interest. Caban and Rheingans[11] designed a transfer function that considered the texture of a feature. Additionally, Correa and Ma developed a number of multidimensional transfer functions for various purposes.[12–14] To conveniently modify a transfer function, Wu and Qu[15] devised three operations for combining transfer functions. Zhou and Takatsuka[16] utilized the contour tree and residue flow model to automatically generate a harmonic color transfer function.

Paper 10200PRRR received Nov. 19, 2010; revised manuscript received Sep. 6, 2011; accepted for publication Sep. 29, 2011; published online Oct. 26, 2011.

1017-9909/2011/20(4)/043004/11/$25.00 © 2011 SPIE and IS&T

For any volume rendering algorithm that processes volume data in a spatial domain, the transfer function can be applied to the volume and calculate the rendered result within the same time bound. However, for FVR with frequency domain data, rerendering the volume after applying a transfer function within the same $O(n^2 \log n)$ time bound is challenging. A naive approach would be to employ one of the following two methods: apply the transfer function to the volume data and recompute the Fourier transform, or implement the convolution operation in the frequency domain. The time required for both approaches exceeds $O(n^2 \log n)$. To improve the re-rendering time after the application of a transfer function, Nagy et al.[17] proposed designing a binary classification transfer function for FVR. They used the Fourier series of a step function to eliminate unwanted voxels. Cheng and Ching[18] improved on Nagy's method by employing the linear combination property of the Fourier transform and the Bézier curve equation to design a continuous transfer function.

This study further improves the method developed by Cheng and Ching,[18] demonstrating that the curve equation defined using the B-spline is more useful for controlling the shape of the spline curve. In this approach, the memory required is dependent on the number of control points. Using additional control points enables the flexible design transfer function, but restricts the use of graphic processing unit (GPU) due to the memory limitation. To reduce the required memory space, this study clusters the control points into groups. The control points in the same group either share identical shading factors or their shading factors can be obtained through polynomial interpolation. Therefore, the required memory space is linearly proportional to the number of groups.

This paper is organized as follows. In Sec. 2, we briefly describe the preliminaries of FVR. Section 3 details the process of designing a B-spline curve as the transfer function. The method to reduce the required memory by clustering control points is presented in Sec. 4. The results are shown in Sec. 5, and Sec. 6 contains a summary and discussion.

2 Preliminaries

In this section, we briefly describe a number of preliminaries of FVR.

2.1 *Fourier Slice Theorem*

Figure 1 shows the Fourier slice theorem[19] in 2-D space. Given function $f(x, y)$ in a spatial domain [Fig. 1(a)], let $p_\theta(x')$ be the projection of $f(x, y)$ in direction θ, where

$$x' = x \cos\theta + y \sin\theta, \tag{1}$$

and

$$y' = -x \sin\theta + y \cos\theta. \tag{2}$$

Let $F(u, v)$ be the Fourier transform of $f(x, y)$ [Fig. 1 (b)], and $P_\theta(u')$ be the 1-D Fourier transform of $p_\theta(x')$. $P_\theta(u')$ is a line segment with orientation θ in $F(u, v)$ passing through the origin. Using the Fourier slice theorem, the projection of $f(x, y)$ along orientation θ can be obtained by taking the inverse Fourier transform of $P_\theta(u')$.

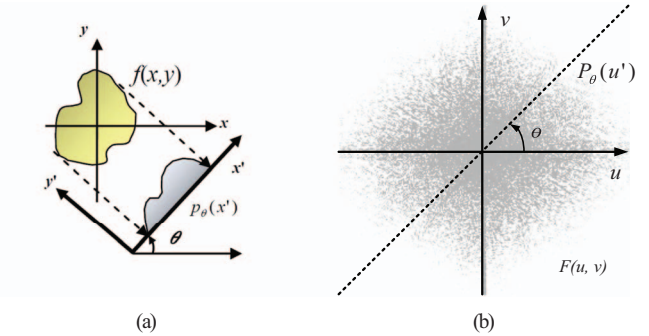

Fig. 1 2-D Fourier slice theorem (a) in the spatial domain; $p_\theta(x')$ is the projection of $f(x, y)$ along $\theta + \pi/2$ (b) in the frequency domain; the line segment, $P_\theta(u')$, is the 1-D Fourier transform of $p_\theta(x')$.

The Fourier slice theorem holds in 3-D space (Fig. 2). Let $F(u, v, w)$ be the Fourier transform of 3-D volume $\boldsymbol{f(x, y, z)}$. Given the view direction $\mathbf{v}$, the projection $P_\mathbf{v}(x, y)$ is the projection along $\mathbf{v}$. Using the Fourier slice theorem, $P_\mathbf{v}(x, y)$ is obtained by taking the inverse Fourier transform of $P_\mathbf{v}(u', v')$, where $P_\mathbf{v}(u', v')$ is the frequency signals in a 2-D plane passing through the origin in $F(u, v, w)$.

The FVR method can be summarized in Eq. (3). Given the volume data $f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^3$, we define $\Pi_\mathbf{v}$ as an operator that performs FVR from the viewing direction $\mathbf{v}$. The FVR of $f(x)$ from $\mathbf{v}$ can be presented as:

$$\begin{aligned} I &= \Pi_\mathbf{v}[f(\mathbf{x})] \\ &= \mathrm{FT}_2^{-1}[\mathrm{FT}_3[f(\mathbf{x})]\delta_\mathbf{v}]. \end{aligned} \tag{3}$$

In the first part of Eq. (3), I is the projection of $f(\mathbf{x})$ from the viewing direction $\mathbf{v}$; in the second part, $\delta_\mathbf{v}$ restricts the spectrum of the 3-D Fourier transform, FT_3, to a plane passing through the origin and perpendicular to $\mathbf{v}$. We then take the 2-D inverse Fourier transform, FT_2^{-1}, to obtain the projection I.

2.2 *Linear Combination Property of the Fourier Transform and the Shading Model*

Let $f(x)=a \cdot A(x) + b \cdot B(x)$ where $A(x)$ and $B(x)$ are two integrable functions. The Fourier transform, FT, possesses the linear combination property, as shown in Eq. (4),

$$\begin{aligned} \mathrm{FT}[f(x)] &= \mathrm{FT}[a \cdot A(x) + b \cdot B(x)] \\ &= a \cdot \mathrm{FT}[A(x)] + b \cdot \mathrm{FT}[B(x)], \end{aligned} \tag{4}$$

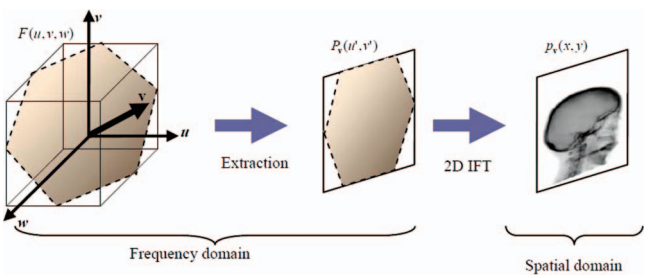

Fig. 2 The FVR algorithm. In the frequency domain, the 2-D frequency function $P_\mathbf{v}(u', v')$ passing through the origin of $F(u, v, w)$ is extracted. The projection $p_\mathbf{v}(x, y)$ is obtained by taking the 2-D inverse Fourier transform of $P_\mathbf{v}(u', v')$.

where a and $b \in \mathbf{R}$ are two constants. This property can be used to efficiently implement the shading model and transfer function of FVR.

The shading model of FVR proposed by Levoy[5] employs the linear combination property advantage. Let g be a shading model applied to the volume data $f(\mathbf{x})$. From Eq. (3), the FVR of the resulting volume can be presented as

$$I = \Pi_{\mathbf{v}}\{g[f(\mathbf{x})]\} = \mathrm{FT}_2^{-1}(\mathrm{FT}_3\{g[f(\mathbf{x})]\}\delta_{\mathbf{v}}). \tag{5}$$

If $g[f(\mathbf{x})]$ can be decomposed into a linear combination of m terms, then

$$g[f(\mathbf{x})] = h_0 \cdot g_0[f(\mathbf{x})] + h_1 \cdot g_1[f(\mathbf{x})] + \cdots + h_{m-1} \cdot g_{m-1}[f(\mathbf{x})], \tag{6}$$

where h_i is the shading factor. Substituting Eq. (6) into $g[f(\mathbf{x})]$ in Eq. (5), we obtain

$$\begin{aligned} I &= \Pi_{\mathbf{v}}\{h_0 \cdot g_0[f(\mathbf{x})] + h_1 \cdot g_1[f(\mathbf{x})] + \cdots + h_{m-1} \cdot g_{m-1}[f(\mathbf{x})]\} \\ &= FT_2^{-1}[(h_0 \mathrm{FT}_3\{g_0[f(\mathbf{x})]\} + h_1 \cdot \mathrm{FT}_3\{g_1[f(\mathbf{x})]\} + \cdots + h_{m-1} \cdot \mathrm{FT}_3\{g_{m-1}[f(\mathbf{x})]\})\delta_{\mathbf{v}}] \\ &= h_0 \cdot \Pi_{\mathbf{v}}\{g_0[f(\mathbf{x})]\} + h_1 \cdot \Pi_{\mathbf{v}}\{g_1[f(\mathbf{x})]\} + \cdots + h_{m-1} \cdot \Pi_{\mathbf{v}}\{g_{m-1}[f(\mathbf{x})]\} \\ &= \sum_{i=0}^{m-1} h_i \cdot \Pi_{\mathbf{v}}(g_i[f(\mathbf{x})]). \end{aligned} \tag{7}$$

In Eq. (7), the rendered result is the summation of m weighted FVR results. Therefore, recomputing the 3-D Fourier transform is not required if $\Pi_{\mathbf{v}} g_i[f(\mathbf{x})]$, $i = 0, 1, \ldots, m-1$, in Eq. (7) are available.

2.3 *Bézier Curve Transfer Function*

Cheng and Ching applied the shading model and used a Bézier curve as the transfer function.[18] For completeness, we briefly describe the method in this subsection. A Bézier curve defined by m control points is shown as follows:

$$B(u) = \sum_{i=0}^{m-1} w_i b_{i,m-1}(u), \tag{8}$$

where u and $w_i \in \mathbf{R}$, and

$$b_{i,m-1}(u) = \binom{m-1}{i}(1-u)^{m-i-1}u^i. \tag{9}$$

As shown in Fig. 3, a Bézier curve is defined in the **uw** coordinate system, where the **u**-axis corresponds to the voxel value and the **w**-axis corresponds to the shading weight of the voxel value. The Bézier curve consists of m control points, $\mathbf{p}_i = (u_i, w_i)$, $i = 0, 1, \ldots, m-1$, $u_i = u_0 + ih$ and $h = 1/(m-1)$. By adjusting w_i, we edit the transfer function. Combining Eqs. (8), (5), and (7) FVR with a Bézier curve transfer function is written as

$$I = \mathrm{FT}_2^{-1}\left(\mathrm{FT}_3\left\{\sum_{i=0}^{m-1} w_i b_{i,m-1}[f(\mathbf{x})]\right\}\delta_{\mathbf{v}}\right)$$

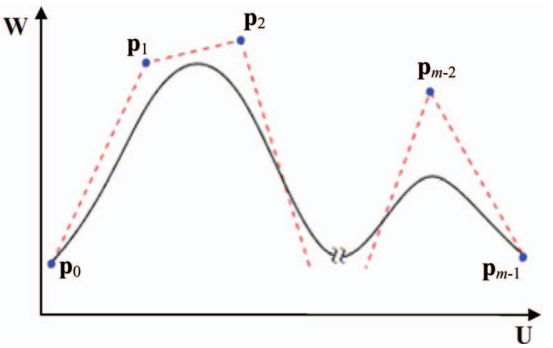

Fig. 3 A spline with m control points can be used to present the transfer function. The shape of a spline curve can be modified by changing the height of the control points using a GUI system.

$$= \mathrm{FT}_2^{-1}\left(\left\{\sum_{i=0}^{m-1} w_i FT_3(b_{i,m-1}[f(\mathbf{x})])\right\}\delta_{\mathbf{v}}\right)$$

$$= \sum_{i=0}^{m-1} w_i \Pi_{\mathbf{v}}(b_{i,m-1}[f(\mathbf{x})]). \tag{10}$$

According to Eq. (10), the FVR of a volume after applying the Bézier curve transfer function of m control points is the summation of m weighted FVR results.

Viola et al.[20] presented an implementation of FVR using GPU with the advantage of parallel computing to accelerate FVR computation.

3 B-spline Transfer Function

One disadvantage of the Bézier curve transfer function is its inability to enable good localized control of the curve shape. Modifying a control point may alter the shape of a significant portion of the Bézier curve. Compared to the Bézier curve, a B-spline[21] is more useful in controlling the curve shape.

A k-degree B-spline is defined as follows:

$$R(u) = \sum_{i=0}^{m-1} w_i r_{i,k}(u), \tag{11}$$

where

$$r_{i,k}(u) = \frac{s_{i,k}(u)}{\sum_{j=0}^{m-1} s_{j,k}(u)} \tag{12}$$

and $s_{i,k}(u)$ is the B-spline basis function. Let a nondecreasing sequence $\mathbf{T} = \{t_i | t_i \le t_{i+1}, i = 1, \ldots, l-1\}$ be the knot vector. $s_{i,k}(u)$ is recursively defined as

$$s_{i,k}(u) = \frac{u - t_i}{t_{i+k} - t_i} s_{i,k-1}(u) + \frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}} s_{i+1,k-1}(u), \tag{13}$$

and

$$s_{i,0} = \begin{cases} 1 & \text{if } t_i \le u < t_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Given the volume data $f(\mathbf{x})$, taking Eq. (11) as the transfer function, and applying FVR to the resulting volume, we have

$$I = \Pi_{\mathbf{v}}\{R[f(\mathbf{x})]\}. \tag{15}$$

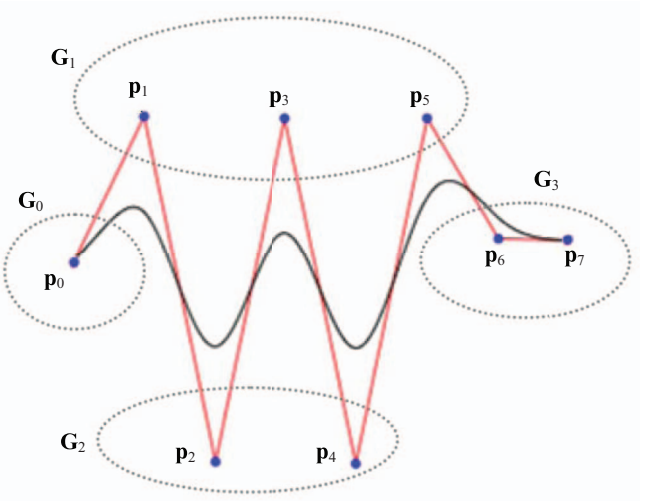

Fig. 4 Seven control points are clustered into four groups. In this case, only four copies of volume data are required.

Substituting Eq. (11) into Eq. (15), and according to Eq. (7), we have

$$
\begin{aligned}
I &= \mathrm{FT}_2^{-1}\left(\mathrm{FT}_3\left\{\sum_{i=0}^{m-1} w_i r_{i,k}[f(\mathbf{x})]\right\}\delta_{\mathbf{v}}\right) \\
&= \mathrm{FT}_2^{-1}\left[\left(\sum_{i=0}^{m-1} w_i \mathrm{FT}_3\{r_{i,k}[f(\mathbf{x})]\}\right)\delta_{\mathbf{v}}\right] \\
&= \sum_{i=0}^{m-1} w_i \Pi_{\mathbf{v}}\{r_{i,k}[f(\mathbf{x})]\}. \qquad (16)
\end{aligned}
$$

The FVR of a volume after applying the B-spline transfer function is the summation of the m-weighted FVR results.

4 Increasing the Control Points

Using additional control points enables the transfer function to be easily shaped to a desired form. Unfortunately, the memory space requirements are linearly proportional to the

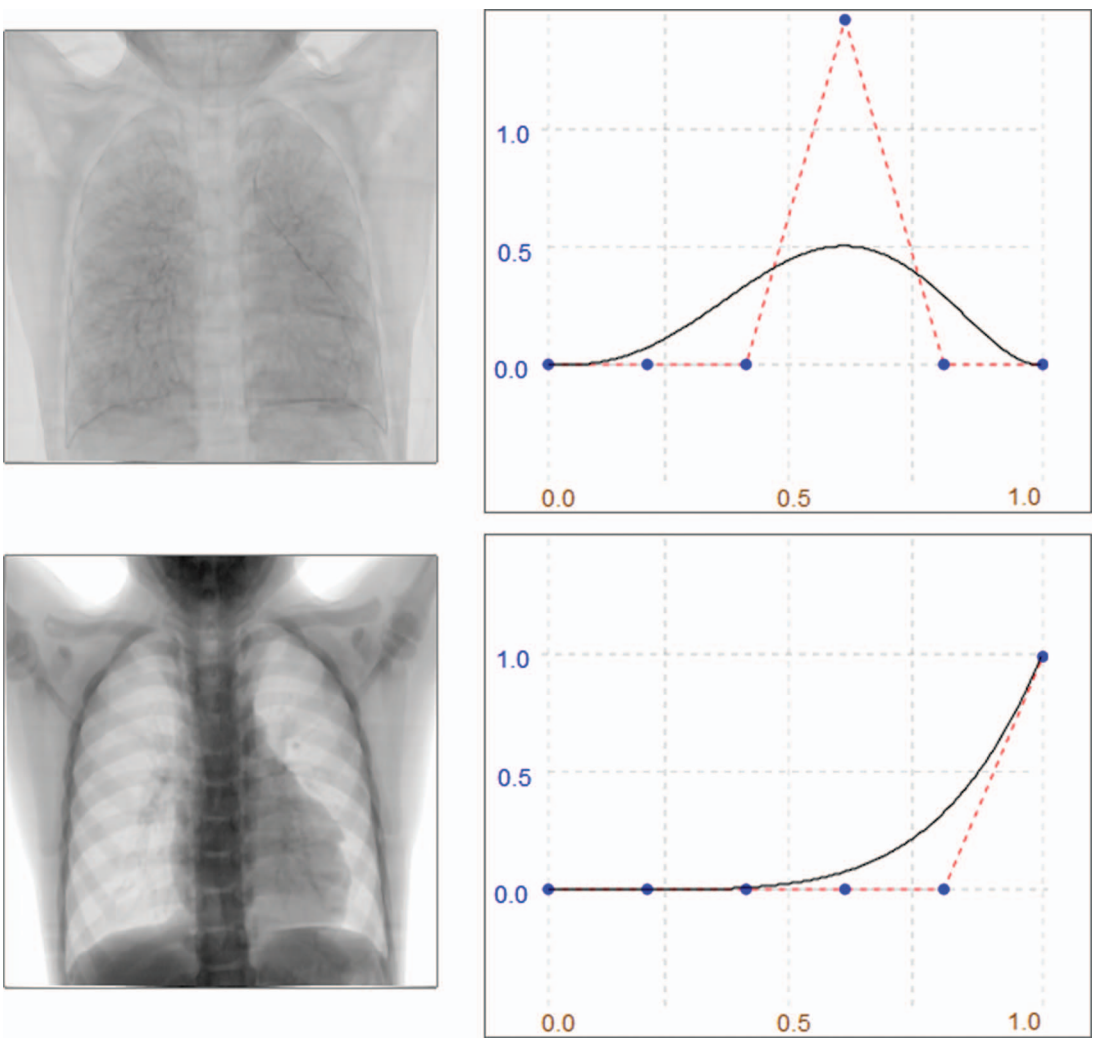

Fig. 5 The rendered results that were obtained using the Bézier curve transfer function. The data set was the CT-scan human chest of 256^3 voxels. The left image in the first row depicted lung structure. The image in the right shows the transfer function. We tried to enhance the gray scale between 0.4 and 0.65. But the gray scale between 0.25 and 0.8 is also enhanced so that the lung capillaries were blurred. The left image in the second row shows the bone structures through enhancing the gray scale between 0.9 and 1.0.

number of control points. Consequently, the volume data may not fit into the GPU memory. To overcome this problem, we propose to increase the number of control points, but then cluster the control points into groups. Control points in the same group will either share identical shading factors or their shading factors can be obtained through interpolation. The memory required then depends on the number of groups, which are manageable in size.

Given n control points in q clusters, the i'th cluster contains m_i control points with the weight w_i, and the formula of the spline curve is provided as follows:

$$R(u) = \sum_{i=0}^{q-1} \sum_{j=0}^{m_i-1} w_i \alpha_j r_{\hat{i}(\mathbf{p}_j),k}(u), \tag{17}$$

where α_j is a constant and $\hat{i}(\mathbf{p}_j)$ is the index of the control point $\mathbf{p}_j$. Applying Eq. (17) to Eq. (7), we have

$$\sum_{i=0}^{q-1} w_i \Pi_{\mathbf{v}} \left\{ \sum_{j=0}^{m_i-1} \alpha_j r_{\hat{i}(\mathbf{p}_j),k}[f(\mathbf{x})] \right\}. \tag{18}$$

The control points in the same cluster share the same shading weight; thus, only one copy of the volume data is stored in the GPU memory. In Fig. 4, the curve consists of eight control points. We can cluster the eight control points into four groups, $\mathbf{G}_0 = \{\mathbf{p}_0\}$, $\mathbf{G}_1 = \{\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_5\}$, $\mathbf{G}_2 = \{\mathbf{p}_2, \mathbf{p}_4\}$ and $\mathbf{G}_3 = \{\mathbf{p}_6, \mathbf{p}_7\}$. In this case, only four copies of the volume data are required.

If the weight of a control point can be obtained by interpolation from the weights of other control points, then the control point can be clustered into a group. Given a sequence of control points $\mathbf{p}_c$, $0 \le i \le c \le j < m$, the weight of the control point $\mathbf{p}_d$, $i < d < j$, can be obtained through polynomial interpolation

$$w_d = \sum_{c=i, c\neq d}^{j} w_c l_c(u_d). \tag{19}$$

In Eq. (19), l_c is an interpolation basis function, such as linear interpolation, spline interpolation, or Lagrange interpolation. For the ease of explanation, we assume only one control point is obtained through interpolation. However, generalizing to

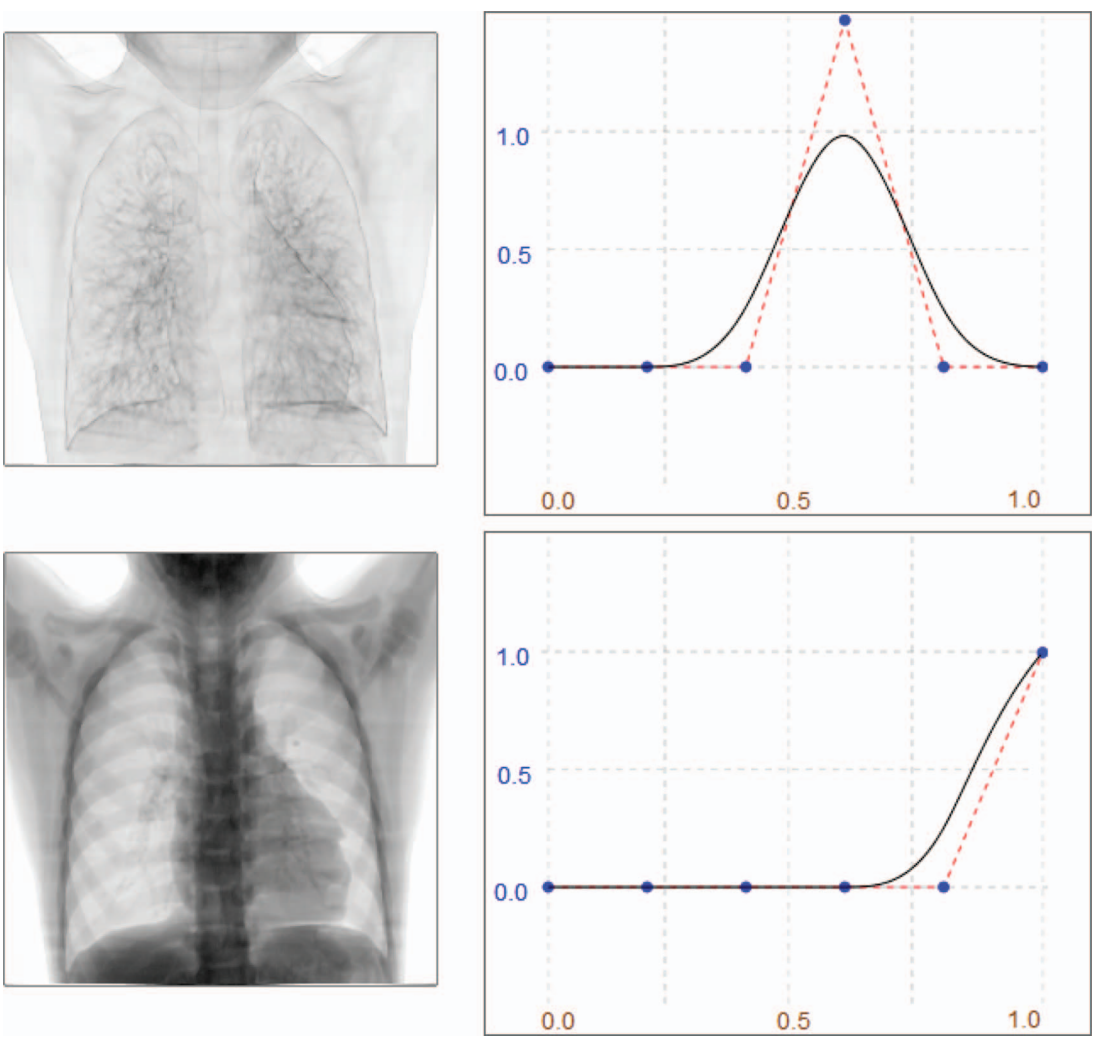

Fig. 6 The same volume data are used as in Fig. 5. The rendered results were obtained using the B-spline transfer function. The first row shows lung structure and second shows bone structure. Using the B-spline transfer function was easier to control the curve shape. The lung capillaries were more clearly shown in the rendered result compared to the result in Fig. 5.

more than one control point is not difficult. Applying the transfer function to a volume, Eq. (16) becomes

$$
\begin{aligned}
I &= w_d \Pi_{\mathbf{v}}\{r_{d,k}[f(\mathbf{x})]\} + \sum_{c=0,c\neq d}^{m-1} w_c \Pi_{\mathbf{v}}\{r_{c,k}[f(\mathbf{x})]\} \\
&= \sum_{c=i,c\neq d}^{j} w_c l_c(u) \Pi_{\mathbf{v}}\{r_{d,k}[f(\mathbf{x})]\} + \sum_{c=0,c\neq d}^{m-1} w_c \Pi_{\mathbf{v}}\{r_{c,k}[f(\mathbf{x})]\} \\
&= \sum_{c=i,c\neq d}^{j} w_i \Pi_{\mathbf{v}}\{l_c(u) r_{d,k}[f(\mathbf{x})] + r_{c,k}[f(\mathbf{x})]\} \\
&\quad + \sum_{0\le c<i} w_c \Pi_{\mathbf{v}}\{r_{c,k}[f(\mathbf{x})]\} + \sum_{j<c<m} w_c \Pi_{\mathbf{v}}\{r_{c,k}[f(\mathbf{x})]\}.
\end{aligned}
\tag{20}
$$

The simplest polynomial interpolation is linear interpolation, given three control points $\mathbf{p}_i = (u_i, w_i)$, $\mathbf{p}_j = (u_j, w_j)$ and $\mathbf{p}_d = (u_d, w_d)$, where $u_i < u_d < u_j$. The weight w_d can be obtained by

$$
w_d = (1-\beta) w_i + \beta w_j, \tag{21}
$$

where $\beta = (u_d - u_i)/(u_j - u_i)$. Applying the transfer function to the volume, Eq. (16) becomes

$$
\begin{aligned}
I &= w_i \Pi_{\mathbf{v}}\{r_{i,k}[f(\mathbf{x})]\} + w_d \Pi_{\mathbf{v}}\{r_{d,k}[f(\mathbf{x})]\} \\
&\quad + w_j \Pi_{\mathbf{v}}\{r_{j,k}[f(\mathbf{x})]\} \\
&= w_i \Pi_{\mathbf{v}}\{r_{i,k}[f(\mathbf{x})]\} + w_j \Pi_{\mathbf{v}}\{r_{j,k}[f(\mathbf{x})]\} \\
&\quad + [(1-\beta) w_i + \beta w_j] \Pi_{\mathbf{v}}\{r_{d,k}[f(\mathbf{x})]\} \\
&= w_i \Pi_{\mathbf{v}}\{r_{i,k}[f(\mathbf{x})] + (1-\beta) r_{d,k}[f(\mathbf{x})]\} \\
&\quad + w_j \Pi_{\mathbf{v}}\{r_{j,k}[f(\mathbf{x})] + \beta r_{d,k}[f(\mathbf{x})]\}.
\end{aligned}
\tag{22}
$$

Equation (22) shows that recomputing the Fourier transform of the volume data is not required if the weights of the control points are modified.

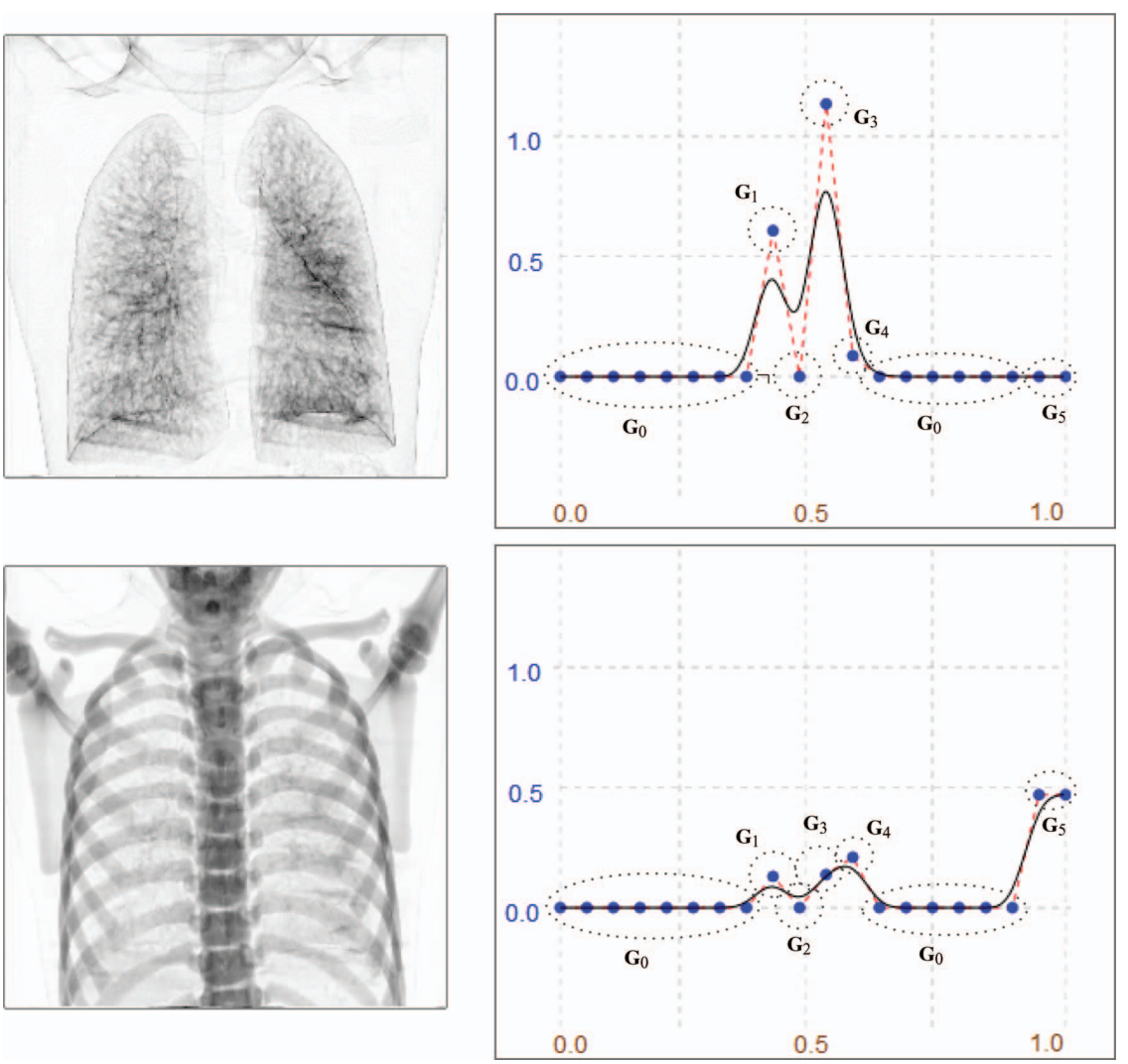

Fig. 7 The same volume data are used as in Fig. 5. The rendered results were obtained using the B-spline transfer function defined by 20 control points. The control points were clustered into six groups, $\mathbf{G}_{0-5}$. The first row shows lung structure, the gray scale between 0.4 and 0.65 was enhanced. The second row shows bone structure, the gray scale between 0.7 and 1.0 was enhanced. Compared to the results that are shown in Fig. 6, the lung capillaries and backbone structure were further enhanced.

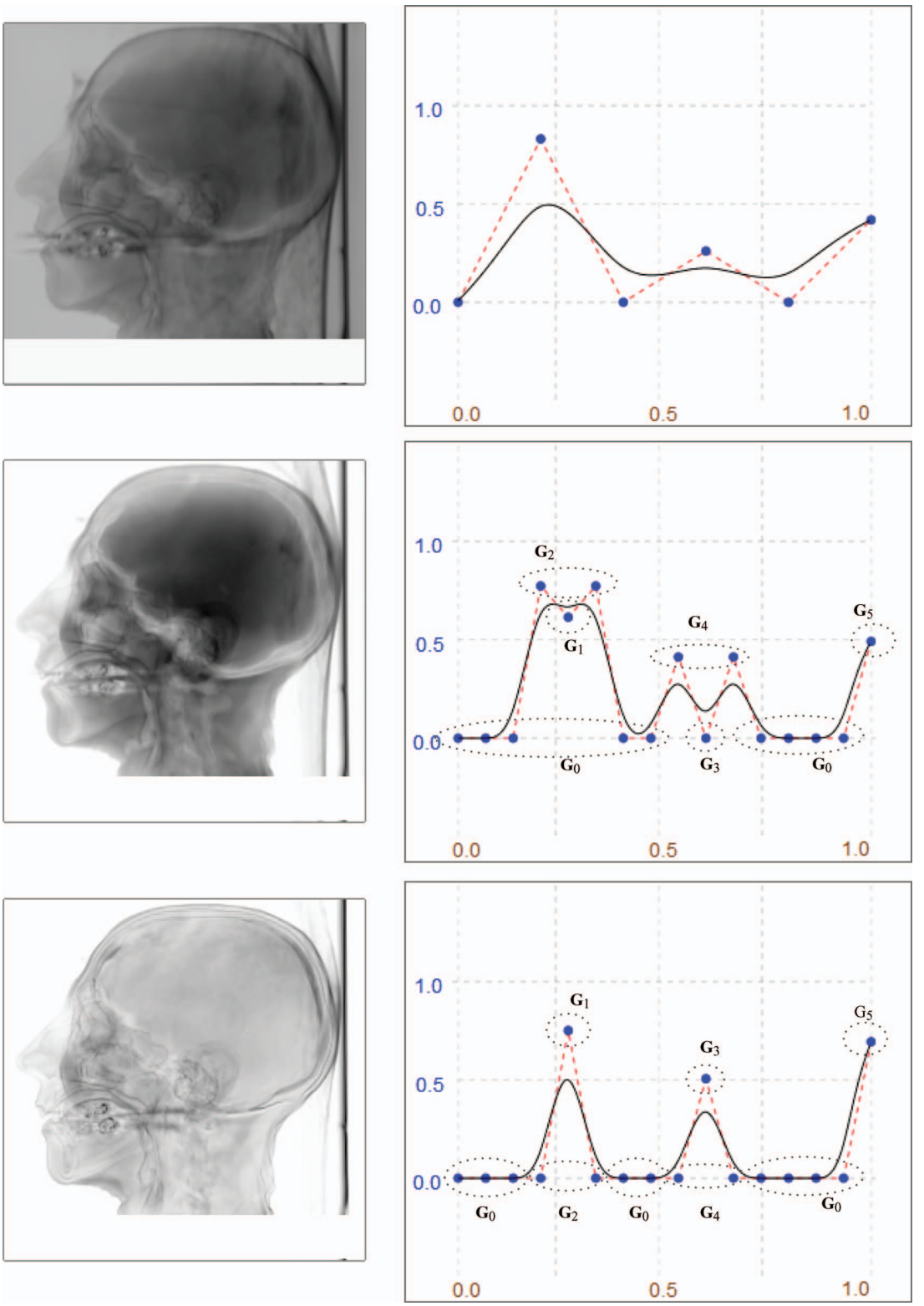

Fig. 8 Chapel Hill CT Head with 256^3 voxels. The results were obtained using the B-spline transfer function. The first row shows the rendered results that were obtained using a B-spline transfer function of six control points. The second and third rows show the rendering results obtained using a B-spline transfer function of 16 control points. These 16 control points were clustered into six groups, $\mathbf{G}_{0-5}$. In the second row, the soft tissue structures were rendered through enhancing $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_4$. On the third row, the boundaries between each structures were rendered through enhancing $\mathbf{G}_1$, $\mathbf{G}_3$, and $\mathbf{G}_5$.

5 Results

We developed a graphical user interface (GUI) software system with the proposed transfer function for FVR. Using the developed software system, users can easily modify the transfer function and change the view direction. Our implementation environment was as follows:

1. CPU: Pentium 4, 2.4 GHz
2. Memory: 2 GB
3. GPU: nVidia GeForce 8800 GT
4. Video memory: 768 MB
5. Viewport size: 512 × 512 pixel

To evaluate the performance, we used two sets of volume data of different size, 128^3 and 256^3, with the B-spline transfer functions defined by one, four, and six control points. We measured the rendering time and preprocessing time, which included computing a 3-D FFT. The performance of different cases is shown in Table 1. Because modern GPUs support parallel computing, increasing the volume size and number of control points slightly affect the frame rate. However, the preprocessing time is linearly proportional to the volume size and the number of control points.

Table 1 Computing time of two cases using the B-spline as a transfer function.

Data size (voxel)	128^3			256^3		
No. control points	6	4	1	6	4	1
Memory required (MB)	96	64	16	768	512	128
Preprocessing (s)	18.1	12.1	3	167.4	109	8.7
Frame rate (fps)	64.1	65.8	66.46	59.2	62.41	66.26

This study presents the rendered results using several data sets. The first data set was CT-scan human chest volume

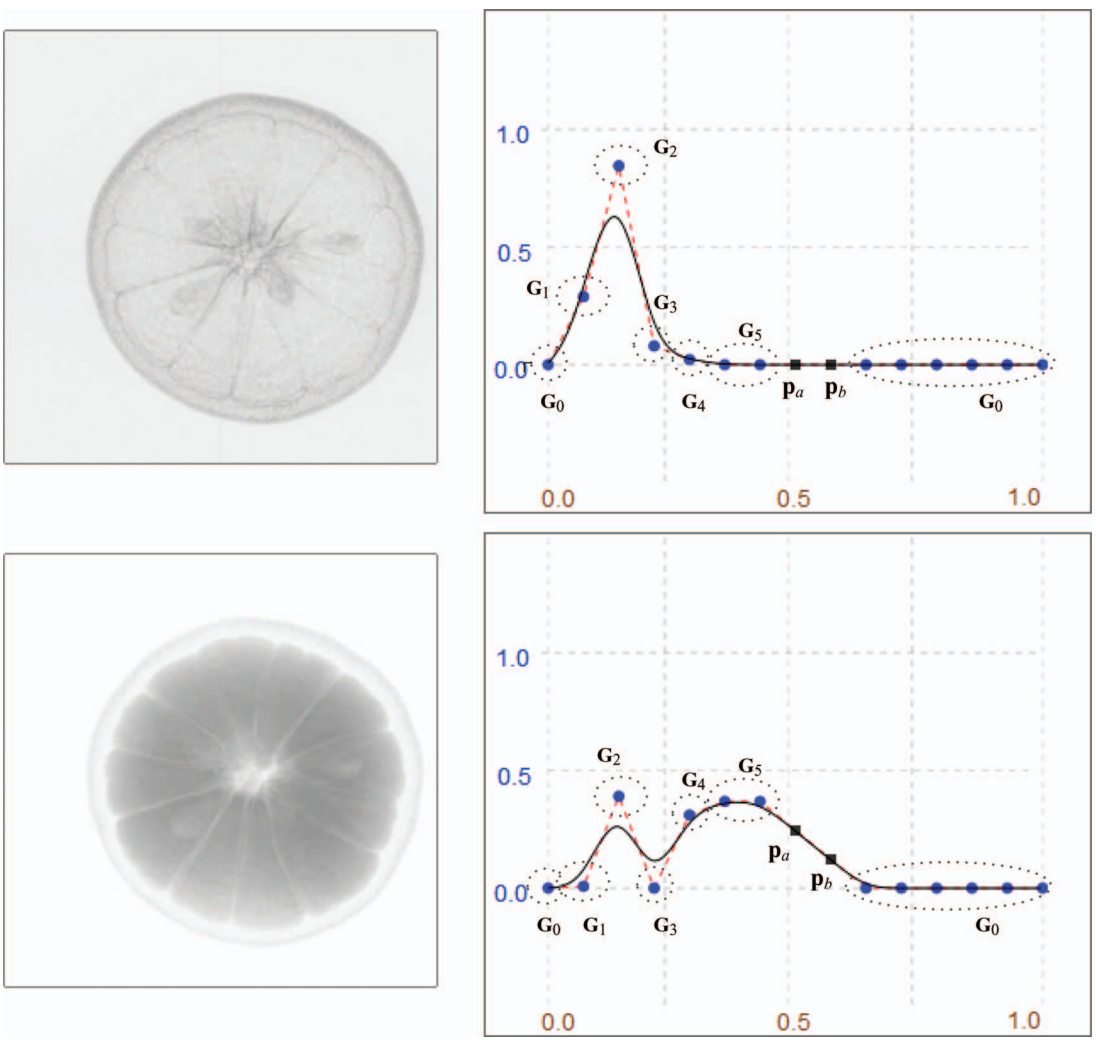

Fig. 9 The Orange with 256^3 voxels. The results were obtained using a B-spline transfer function of 15 control points. We clustered 13 control points that are expressed by blue circles into six groups, $\mathbf{G}_{0-5}$. Two points, $\mathbf{p}_a$ and $\mathbf{p}_b$ (expressed by black squares), were obtained through interpolation. Their vertical positions were linearly interpolated from $\mathbf{G}_0$ and $\mathbf{G}_5$. In the top row, we mainly enhanced the weight of $\mathbf{G}_2$ to render the peel. The bottom row shows the results of the orange pulp. The gray scale of the pulp is between 0.3 to 0.5, the weight of $\mathbf{G}_4$ and $\mathbf{G}_5$ was enhanced to render the orange pulp.

data. The volume size was 256^3 voxels. The rendered results obtained using the Bézier curve and the B-spline transfer functions defined by six control points are shown in Figs. 5 and 6, where the images on the left show the rendered result obtained by applying the transfer functions that are shown on the right. The transfer functions in the first and second rows of both figures were designed to depict lung and bone structure, respectively. As shown in Fig. 6, using B-spline as the transfer function can significantly enhance lung capillaries. The result in Fig. 7 show that using a transfer function with greater control points can achieve better rendering. This study used a B-spline transfer function with 20 control points clustered into six groups. In a comparison of the images shown in Figs. 6 and 7, the lung capillaries were significantly enhanced.

We also tested the three sets of volume data (see Ref. 22) commonly used in most volume rendering literature. In Fig. 8, the input is the Chapel Hill CT Head with 256^3 voxels. The rendered image in the first row is obtained by applying the B-spline transfer function of six control points. The images in the second and third rows are obtained by applying the B-spline transfer function with 18 control points clustered into six groups, $\mathbf{G}_{0-5}$. The second row shows the soft tissue structures, and the third row shows the boundaries of the soft tissue and bone structures. To render the soft tissue structures (gray scales between 0.125 and 0.4), we enhanced the weights of $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_4$. To render the boundaries of each structure, we enhanced the weights of $\mathbf{G}_1$, $\mathbf{G}_3$, and $\mathbf{G}_5$.

Figure 9 shows the FVR results of an orange with 256^3 voxels. We used a B-spline transfer function with 15 control points, and clustered 13 control points (blue circles) into six groups, $\mathbf{G}_{0-5}$. The weights of the two points $\mathbf{p}_a$ and $\mathbf{p}_b$ were obtained from $\mathbf{G}_0$ and $\mathbf{G}_5$ through linear interpolation. The top row shows the results of the peel structure and the bottom row shows the rendered results of the pulp structure. To render the peel, we substantially enhanced the weight

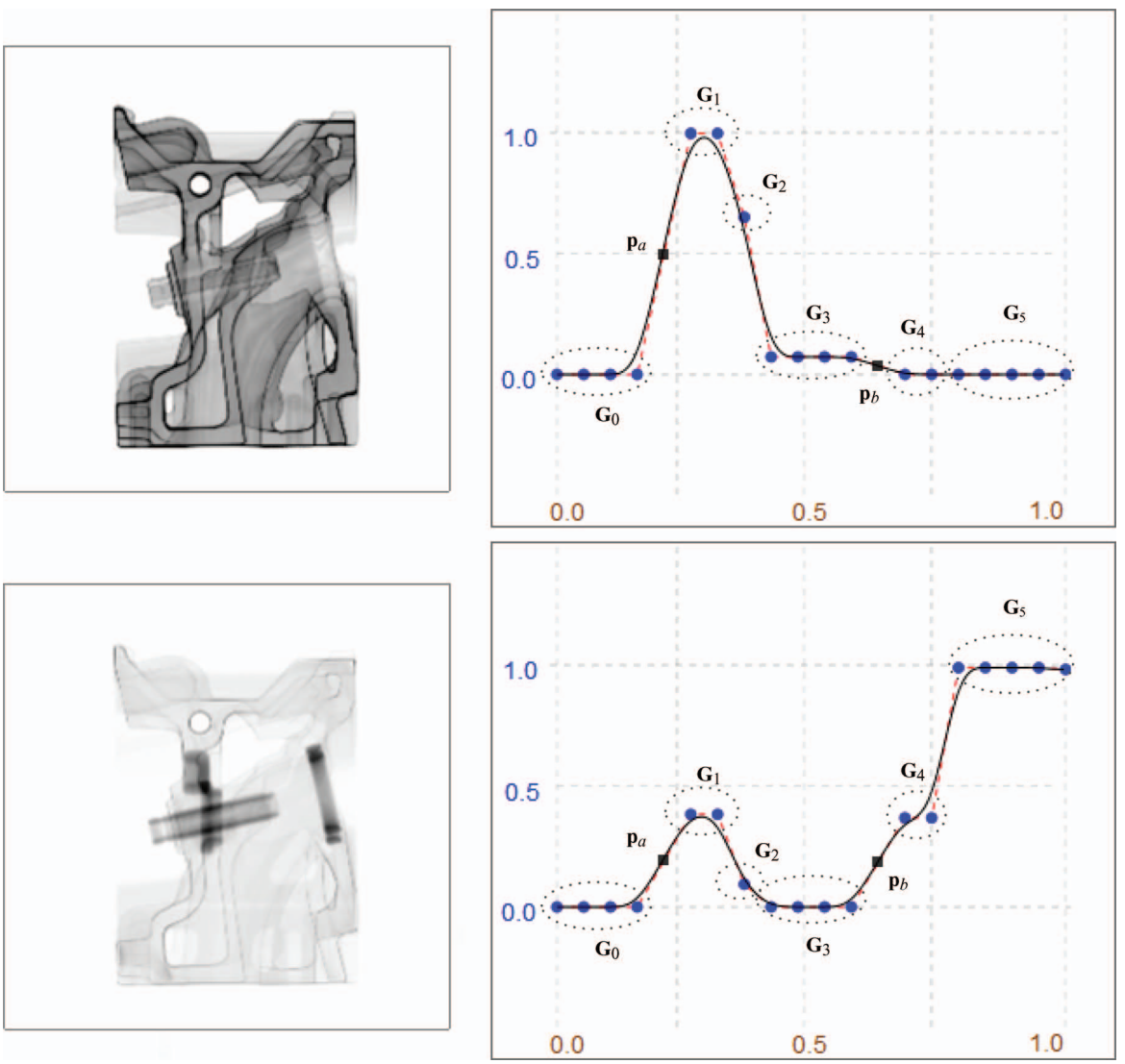

Fig. 10 Engine block with 256^3 voxels. The results were obtained using a B-spline transfer function of 20 control points. We clustered 18 control points that were expressed by blue circles into six groups, $\mathbf{G}_{0-5}$. Two points $\mathbf{p}_a$ and $\mathbf{p}_b$ expressed by black squares were obtained by interpolation. The vertical position of $\mathbf{p}_a$ was linearly interpolated from $\mathbf{G}_0$ and $\mathbf{G}_1$ and the vertical position of $\mathbf{p}_b$ was linearly interpolated from $\mathbf{G}_3$ and $\mathbf{G}_4$. The top and bottom rows show the rendered results of two different portions in the engine.

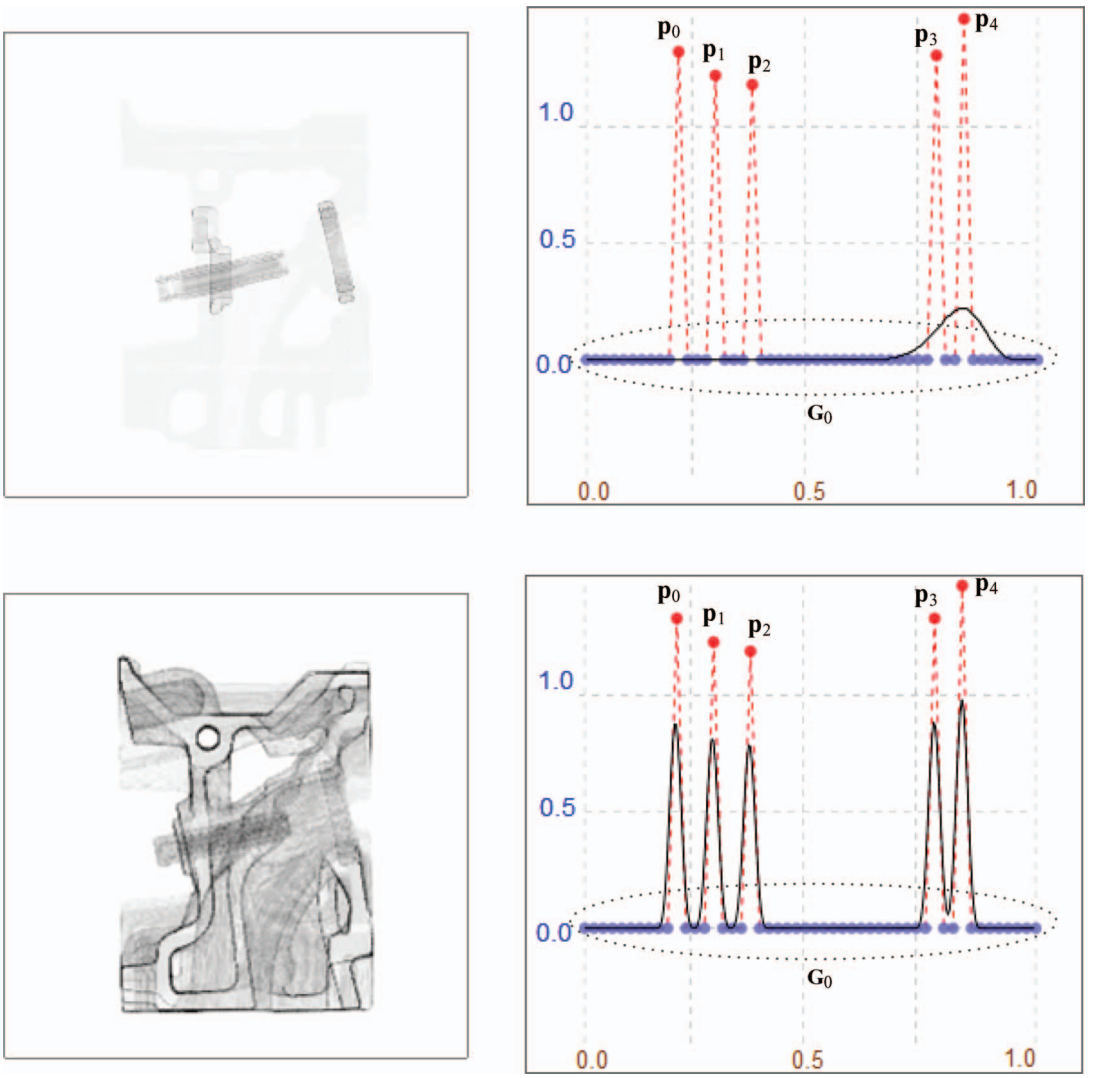

Fig. 11 The same volume data are used as in Fig. 10. The rendered results were obtained using a Bézier curve transfer function (top row) and a B-spline transfer (bottom row). The number of control points was 50. $\mathbf{p}_0$ to $\mathbf{p}_4$ can be moved to adjust the curve. The other 45 control points were clustered in a group $\mathbf{G}_0$. The B-spline transfer function could enhance the regions of interest, whereas the Bézier curve transfer function could not. The main reason is because the B-spline provides a much localized control of the curve. This figure demonstrates that B-spline curve transfer function is much useful than the Bézier curve transfer function.

of $\mathbf{G}_2$. The weights of $\mathbf{G}_1$ and $\mathbf{G}_3$ were slightly enhanced to smooth change the gray scale in the rendered result. To render the orange pulp, we enhanced the weights of $\mathbf{G}_4$ and $\mathbf{G}_5$. Changing the vertical position of $\mathbf{G}_5$ can also change the vertical positions of $\mathbf{p}_a$ and $\mathbf{p}_b$, ensuring the smooth change of the gray scale in the rendered results.

Data for the final test from an engine with 256^3 voxels. The results are shown in Figs. 10 and 11. In Fig. 10, we used the B-spline transfer function of 20 control points, where 18 control points expressed by blue circles were clustered into six groups, $\mathbf{G}_{0-5}$. The two control points $\mathbf{p}_a$ and $\mathbf{p}_b$, expressed by black squares, were obtained through interpolation. The vertical position of $\mathbf{p}_a$ is linearly interpolated by $\mathbf{G}_0$ and $\mathbf{G}_1$ and the vertical position of $\mathbf{p}_b$ is linearly interpolated by $\mathbf{G}_3$ and $\mathbf{G}_4$. The top row shows the results of the interior structure. To depict the interior structure of the engine, we enhanced only $\mathbf{G}_1$ and $\mathbf{G}_2$. To render the hardest portion of the engine, $\mathbf{G}_5$ was primary enhanced; the rendered results are shown in the bottom row. In Fig. 11, the top row shows the rendered result of applying the Bézier curve transfer function of 50 control points. The heights of the five control points, $\mathbf{p}_0$ to $\mathbf{p}_4$, could be changed to adjust the curve shape. The other 45 control points were clustered in a group $\mathbf{G}_0$. As shown in Fig. 11, sliding $\mathbf{p}_0$ to $\mathbf{p}_2$ upward cannot raise the curve to the desired position. Moving $\mathbf{p}_3$ and $\mathbf{p}_4$ upward only produces a bump on the curve. By contrast, using the B-spline transfer function, we can generate the desired shape of the transfer function. With numerous control points each controlling a small portion of the curve shape, the B-spline transfer function is significantly more useful compared to the Bézier transfer function.

6 Conclusions and Discussion

This paper shows that the $O(n^2 \log n)$ rendering time can be maintained by applying a transfer function defined by B-spline. We also show that the B-spline is significantly

more useful compared to the Bézier curve when designing a transfer function.

However, one disadvantage of the proposed method is the significant amount of memory required. A user can more easily design a curve shape if more control points are used. However, the memory required increases in linear proportion to the number of control points, and the GPU memory is limited. Currently, commercially available GPUs can process a volume of 256^3 if six control points are used. When additional control points are required, the control points can be clustered into groups if they possess the same weight or their weight can be obtained through interpolation. Each group requires a copy of the volume to ensure the required memory can be maintained at a manageable size.

One limitation of the proposed method is that transfer functions containing a negative value are not allowed. We aim to examine this issue is our future research. Additionally, we also aim to design an efficient sampling method for extracting a slice of the frequency domain volume. We observed that the computing time required for sampling a slice of the frequency domain data is longer than that required to perform the inverse 2-D Fourier transform. A more efficient sampling method is required to improve the overall performance.

References

1. W. E. Lorensen and H. E. Cline, "Marching cubes: a high resolution 3D surface construction algorithm," in *Proc. of ACM SIGGRAPH Computer Graphics*, Vol. 21, pp. 163–169 (1987).
2. N. Max, "Optical models for direct volume rendering," *IEEE Trans. Vis. Comput. Graphics* **1**(2), 99–108 (1995).
3. M. Ikits, J. Kniss, A. Lefohn, and C. Hansen, "Volume rendering techniques," Chapter 39, in *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, pp. 667–692, Addison Wesley, Boston (2004).
4. S. Dunne, S. Napel, and B. Rutt, "Fast reprojection of volume data," in *Proc. of Conf. on Visualization in Biomedical Computing*, Atlanta, pp. 11–18 (1990).
5. M. Levoy, "Volume rendering using the Fourier projection-slice theorem," in *Proc. of Graphics Interface*, Vancouver, pp. 61–69 (1992).
6. T. Malzbender, "Fourier volume rendering," *ACM Trans. Graphics* **12**(3), 233–250 (1993).
7. A. Entezari, R. Scoggins, T. Möller, and R. Machiraju, "Shading for Fourier volume rendering," in *Proc. of 2002 IEEE Symp. on Volume Visualization and Graphics*, pp. 131–138 (2002).
8. K. Engel, M. Kraus, and T. Ertl, "High-quality pre-integrated volume rendering Using Hardware-Accelerated Pixel shading," in *Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware*, Los Angeles, pp. 9–16 (2001).
9. G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Mller, "Curvature-based transfer functions for direct volume rendering: methods and applications," in *Proc. IEEE Visualization*, pp. 513–520 (2003).
10. E. B. Lum and K.-L. Ma, "Lighting transfer functions using gradient aligned sampling," in *Proc. IEEE Visualization*, pp. 289–296 (2004).
11. J. J. Caban and P. Rheingans, "Texture-based transfer functions for direct volume rendering," *IEEE Trans. Vis. Comput. Graphics* **14**(6), 1364–1371 (2008).
12. C. D. Correa and K.-L. Ma, "Size-based transfer functions: a new volume exploration technique," *IEEE Trans. Vis. Comput. Graphics* **14**(6), 1380–1387 (2008).
13. C. D. Correa and K.-L. Ma, "The occlusion spectrum for volume classification and visualization," *IEEE Trans. Vis. Comput. Graphics* **15**(6), 1465–1472 (2009).
14. C. D. Correa and K.-L. Ma, "Visibility histograms and visibility-driven transfer functions," *IEEE Trans. Vis. Comput. Graphics* **17**(2), 192–204 (2011).
15. Y. Wu and H. Qu, "Interactive transfer function design based on editing direct volume rendered images," *IEEE Trans. Vis. Comput. Graphics* **13**(5), 1027–1040 (2007).
16. J. Zhou and M. Takatsuka, "Automatic transfer function generation using contour tree controlled residue flow model and color harmonics," *IEEE Trans. Vis. Comput. Graphics* **15**(6), 1481–1488 (2009).
17. Z. Nagy, G. Müller, and R. Klein, "Classification for Fourier volume rendering," in *Proc. of 12th Pacific Conf. on Computer Graphics and Applications*, Seoul, pp. 51–58 (2004).
18. C.-C. Cheng and Y.-T. Ching, "Transfer function design for Fourier volume rendering and implementation using GPU," *Proc. SPIE* **6918**, 691806 (2008).
19. R. N. Bracewell, *The Fourier Transform and Its Applications*, 3rd ed., McGraw-Hill, New York (1999).
20. I. Viola, A. Kanitsar, and M. E. Gröller, "GPU-based frequency domain volume rendering," in *Proc. of 20th Spring Conference on Computer Graphics*, Budmerice, Slovakia, pp. 55–64 (2004).
21. L. A. Piegl and W. Tiller, *The NURBS Book*, 2nd ed., Springer-Verlag, New York (1997).
22. The Volume Library, http://www9.informatik.uni-erlangen.de/External/vollib/ (2006).

Chang-Chieh Cheng received his MS in computer science and engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2007. He is currently pursuing a PhD there in the Department of Computer Science. His research interests are medical imaging analysis, computer graphics, and computer version.

Yu-Tai Ching received his BS in industrial engineering from Tsing Hua University, Hsinchu, Taiwan, in 1980, and his MS and PhD in computer science from Northwestern University, Evanston, Illinois, in 1983 and 1987, respectively. He is currently a professor in the Department of Computer Science, National Chiao-Tung University, Hsinchu. His research interests are design and analysis of computer algorithms, medical image analysis, and computer graphics.