Transfer Function Design for Fourier Volume Rendering and Implementation Using GPU

Chang-Chieh Cheng[a, *], Yu-Tai Ching[a, **]
[a] Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan, ROC.

ABSTRACT

Volume rendering is a technique for volume visualization. Given a set of $N \times N \times N$ volume data, the traditional volume rendering methods generally need $O(N^3)$ rendering time. The FVR (Fourier Volume Rendering), that takes advantage of the Fourier slice theorem, takes $O(N^2 \log N)$ rendering time once the Fourier Transform of the volume data is available. Thus the FVR is favor to designing a real-time rendering algorithm with a preprocessing step. But the FVR has a disadvantage that resampling in the frequency domain causes artifacts in the spatial domain. Another problem is that the method for designing a transfer function is not obvious. In this paper, we report that by using the spatial domain zero-padding and tri-linear filtering can reduce the artifacts to an acceptable rendered image quality in spatial domain. To design the transfer function, we present a method that the user can define a transfer function by using a Bézier curve first. Based on the linear combination property of the Fourier transform and Bézier curve equation, the volume rendered result can be obtained by adding the weighted frequency domain signals. That mean, once a transfer function is given, we don't have to recompute the Fourier transform of the volume data after the transfer function applied. This technique makes real-time adjustment of transfer function possible.

Keywords: Fourier volume rendering, transfer function design, classification, Bézier curve, Graphics Process Unit.

1. INTRODUCTION

Volume rendering is a technique to display a 2D projection of a 3D volume data in any view direction. This technique is commonly used in scientific visualization and medical volume data visualization. Volume rendering can be divided into two types, the surface volume rendering and the direct volume rendering. The surface volume rendering technique renders the isosurface specified by a given threshold. Lorensen [1] proposed the Marching cubes method for surface volume rendering. He used 15 geometric primitive types to create the isosurfaces in the voxel. There is ambiguity problem while determining the isosurface. Lin [3] proposed a method to determine the isosurface based on the saddle values of a trilinear function to prevent the ambiguity problem. The direct volume rendering method is a technique that imitates the X-ray passing through an object. The 2D projections are obtained by integrating the voxels along the line of view direction[5]. Regardless the isosurface volume rendering or the direct volume rendering, we need to process all of the voxels in the volume, i.e., they take $O(n^3)$ time.

The FVR is an implementation of the direct volume rendering that was proposed by Dunne [6]. FVR is designed based on the Fourier transform. Suppose that the Fourier transform of the volume data is available. The inverse Fourier transform of a slice in the frequency domain, that passes through the origin and is perpendicular to view direction, is the projection along view direction of the volume. Based on the Fourier slice theorem, since we only need to compute the inverse Fourier transform of a slice, we can volume render the volume in $O(n^2 \log n)$ time. Unfortunately, there is a resampling procedure involved in the FVR that causes artifacts. Malzbender [7] designed several filters to reduce the artifacts. Another work on FVR is due to Levoy [9] who proposed three shading models of FVR including directional lighting and depth cueing.

*chengchc@cs.nctu.edu.tw; phone +886-3-5712121 # 59268
**ytc@cs.nctu.edu.tw; phone +886-3-5131547

Medical Imaging 2008: Visualization, Image-guided Procedures, and Modeling, edited by Michael I. Miga, Kevin Robert Cleary, Proc. of SPIE Vol. 6918, 691806, (2008)
1605-7422/08/$18 · doi: 10.1117/12.768949

A transfer function is to define a classification of voxels and converts the value of each voxel into color intensity or opacity. Transfer function is an important technique for volume rendering in order to enhance the region of interest. It is not difficult to design a transfer function for the traditional volume algorithms. But it is not obvious to do the same thing in the frequency domain, i.e., for the FVR. The major difficulty is that if we multiply each voxel of the volume by a transfer function, we actually need to do convolution in the frequency domain. That means a heavy computation is required and the time required is much more than $O(n^2 \log n)$, the time required for a 2D inverse Fourier transform. Nagy [13] proposed a method to design a transfer function for FVR. He used the Fourier series of a Step function as the transfer function. He then employed Levoy's shading model to implement his method.

In this paper, we present a method for designing the transfer function for FVR. The user can provide a transfer function specified by a Bézier curve and the rendered result can be obtained in $O(n^2 \log n)$ time. The organization of this paper is as follows. In Section 2, we briefly describe the theories required for FVR. The proposed method for designing transfer function is presented in Section 3. Then the implementation and results are shown in Section 4. Section 5 contains summary, discussion, and the future work.

2. METHODS

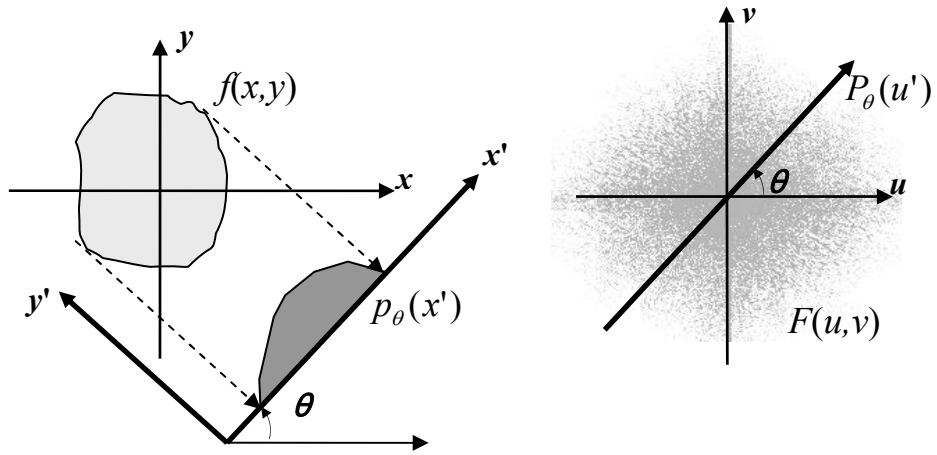
2.1 The Fourier Slice Theorem

We briefly state the Fourier Slice Theorem [21] in this section. Figure 1 shows that the 2D case of the Fourier slice theorem. Given a function *f*(*x, y*) in the spatial domain (Fig. 1a), let $p_\theta(x')$ be a projection of *f*(*x, y*) in θ direction where:

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

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

Let *F*(*u, v*) be the Fourier Transform (Fig. 1b) of *f*(*x, y*). The Fourier Transform of $p_\theta(x')$ is the 1D function $P_\theta(u')$ which is a line segment with orientation θ in *F*(*u, v*) passing through the origin (Fig. 1b). Given $P_\theta(u')$ we can generate the projection of *f*(*x, y*) along orientation θ by taking the inverse Fourier transform of $P_\theta(u')$.



(a) (b)

Fig. 1. The 2D Fourier slice theorem. (a) is the spatial domain and (b) is the frequency domain.

In 3D space, the Fourier Slice theorem is similar. Let **v** be the view direction. Suppose that we have a 2D Fourier transform $P_v(u', v')$ that is in a 2D plane passing through the origin is the frequency domain of the volume data, based on the Fourier slice theorem, we can generate the projections of a volume data along direction **v**.

We conclude the FVR procedures in the following: Given a 3D function $f(x, y, z)$ to represent a volume data, we perform the following procedures.

1. Compute the frequency function $F(u, v, w)$ of f by 3D Fourier transform.
2. Given a viewing direction **v**, let $P_v(u', v')$ be a 2D Fourier transform passing through the origin of $F(u, v, w)$ and in a plane normal to **v**.
3. Obtained the spatial function $p_v(x, y)$ of $P_v(u', v')$ by employing the 2D inverse Fourier transform. The $p_v(x, y)$ is the projection of $f(x, y, z)$ along **v** .

Figure 2 describes the algorithm of FVR.

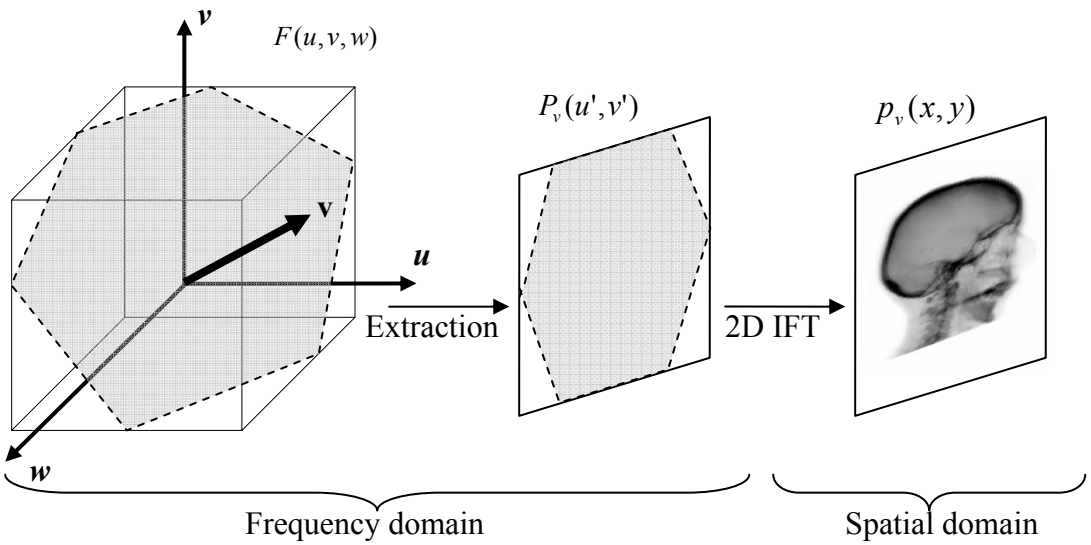
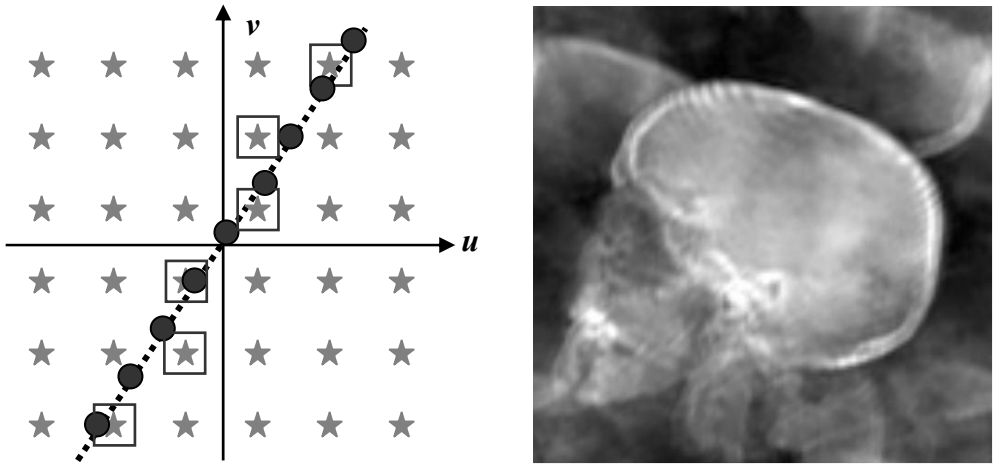


Fig. 2. The algorithm of FVR.

2.2 Resampling Problems

The direct implementation of the FVR causes artifacts. Note that $F(u, v, w)$ is a set of discrete points in 3D space. If we extract a plane passing through the origin in $F(u, v, w)$, the sampled points may not be exactly in the plane (Fig. 3a for a 2D case). The resampling problem occurs in the sampling stage and that produces artifacts (Fig. 3b).



(a) (b)

Fig. 3 The resampling problem. The sampled points may not be exactly in a plane. That causes artifacts as shown in the above figure on the right.

To solve the resampling problem, Malzbender [7] and Lichtenbelt [22] designed different filters to eliminate the artifacts of FVR. The windowed-sinc filter is the ideal reconstruction filter for FVR, but the size of the convolution kernel must be larger than 5^3 voxels. That causes a heavy computational cost. There are two computational efficient filters, the tri-linear filter and tri-cubic filter, are commonly used for sampling in 3D space. They are designed respectively based on the linear interpolation [4] and the cubic interpolation [23]. Although the quality of results obtained by the use of the tri-cubic filter is better than the tri-linear filter, but the computational cost is much higher when the tri-cubic filter is used. We found that by using the spatial domain zero-padding and tri-linear filtering could reduce the artifacts to an acceptable level in spatial domain. The zero-padding in spatial domain is to expand the volume size and surround the original signal by 0. The effect of the zero-padding is to increase the sampling rate in frequency domain so that artifacts can be further reduced. In our method, we combined the zero-padding and tri-linear interpolation to gain the efficiency of computation and quality of the produced image.

2.3 Shading Models

The shading model of the FVR was proposed by Levoy [9]. He used the linear combination property of the Fourier transform to pick out the coefficients of the shading function from the 3D Fourier transform. He defined an operator Π_v to present the FVR of the viewing direction **v**.

$$I = \Pi_v\left(f(x)\right) = FT_2^{-1}\left\{FT_3\left\{f(x)\right\}\delta_v\right\} \tag{3}$$

where I is the projection intensity, $f(x)$ is a volume data, FT_2^{-1} is the inverse 2D Fourier transform, FT_3 is the 3D Fourier transform, and δ_v restricts the spectrum to a plane passing through the origin and perpendicular to viewing direction **v**. The Fourier transform has the linear combination property. It is can be written as follows:

$$FT\left\{af(x) + bg(x)\right\} = aFT\left\{f(x)\right\} + bFT\left\{g(x)\right\} \tag{4}$$

where a, b are constants.

We multiply a shading function $g_t(x)$ to Equation (3), where t is a shading parameter. Equation (3) can be rewritten as follows.

$$I = \Pi_v\left(f(x)g_t(x)\right) = FT_2^{-1}\left\{FT_3\left\{f(x)g_t(x)\right\}\delta_v\right\}. \tag{5}$$

In Equation (5), the 3D Fourier transform must be executed again since t has been changed. However, if we can decompose the $g_t(x)$ into a linear combination as the following:

$$g_t(x) = g_1(x)h_1(t) + g_2(x)h_2(t) + \ldots + g_n(x)h_n(t) \tag{6}$$

After substituting Equation (6) into $g_t(x)$ in Equation (5), we obtained the equation,

$$\begin{aligned} I &= \Pi_v\left(f(x)\left(g_1(x)h_1(t) + g_2(x)h_2(t) + \ldots + g_n(x)h_n(t)\right)\right) \\ &= FT_2^{-1}\left\{\left(h_1(t)FT_3\left\{f(x)g_1(x)\right\} + h_2(t)FT_3\left\{f(x)g_2(x)\right\} + \ldots + h_n(t)FT_3\left\{f(x)g_n(x)\right\}\right)\delta_v\right\} \\ &= h_1(t)\Pi_v\left(f(x)g_1(x)\right) + h_2(t)\Pi_v\left(f(x)g_2(x)\right) + \ldots + h_n(t)\Pi_v\left(f(x)g_n(x)\right) \\ &= \sum_{i=1}^{n} h_i(t)\Pi_v\left(f(x)g_i(x)\right). \end{aligned} \tag{7}$$

Equation (7) is the shading model of FVR.

3. CLASSIFICATION AND TRANSFER FUNCTION DESIGN

There was only Nagy [13] who proposed a classification method for FVR in recent years. He used the Fourier series of the step function to design a transfer function. By applying the shading model, the threshold can be arbitrarily adjusted in rendering stage. But that was a binary classification method. We prefer to have the transfer function to be similar to a user custom function. As shown in Figure 4, the *t* axis is the shading weight, and the *s* axis is the value of the voxel. The user can draw a curve to describe the transfer function that the weight as a function of the voxel value.

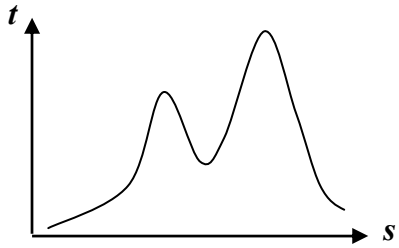


Fig. 4. The shape of transfer function.

We designed a transfer function by the use of the Bézier curve. The Bézier curve equation is shown as follows.

$$B(s) = \sum_{i=0}^{n-1} \binom{n-1}{i} t_i (1-s)^{n-i-1} s^i, \tag{8}$$

where t_i are the location of control points, n is the number of the control points, and $s \in [0,1]$.

Applying the transfer function to the volume data can be written as

$$f(x)B(f(x)). \tag{9}$$

Applying the FVR to Equation (9), we have

$$I = \Pi_v (f(x)B(f(x))). \tag{10}$$

By Equation (3) and substituting Equation (8) into Equation (10), we have

$$\begin{aligned} &= FT_2^{-1}\left\{ FT_3\left\{ f(x)\sum_{i=0}^{n-1}\binom{n-1}{i} t_i (1-f(x))^{n-i-1} f(x)^i \right\}\delta_v \right\} \\ &= FT_2^{-1}\left\{ (\sum_{i=0}^{n-1}\binom{n-1}{i} t_i FT_3\left\{ f(x)(1-f(x))^{n-i-1} f(x)^i \right\})\delta_v \right\} \\ &= \sum_{i=0}^{n-1}\binom{n-1}{i} t_i \Pi_v (f(x)(1-f(x))^{n-i-1} f(x)^i). \end{aligned} \tag{11}$$

From Equation (11), the FVR of a volume after Bézier transfer function is actually a summation of n FVR results multiplying the location of the control points.

4. IMPLEMENTATION AND RESULTS

4.1 The Implementation of FVR

Viola [26] proposed a method to implement the FVR algorithm by GPU. His method was applied in our work. The steps of this implementation are listed as follows:

1. Transform the coordinates of the volume data so that the origin is shifted to the center of the volume data.
2. Obtaining the 3D frequency data after 3D Fourier transform.
3. Shifting the 3D frequency data to centralize the low frequencies to the origin. Then we stored it in the video RAM.
4. Creating a proxy polygon to fetch a plane from the 3D frequency data passing through the origin. There are at most six vertices in this polygon.
5. Rendering the proxy polygon on the FBO (Frame Buffer Object)[25]. At this time, the pixel shaders are triggered for sampling and filtering.
6. Reading the 2D frequencies data from the FBO. We then compute the projection image by the use of the inverse 2D Fourier transform. The inverse Fourier transform can also be done using GPU. In this case, we keep the 2D frequency data in the FBO. Then the projection image is obtained by the use of the inverse 2D FFT on GPU [20].
7. Transforming the coordinates of the projection image.

Graphically, we illustrated these steps in Figure 5.

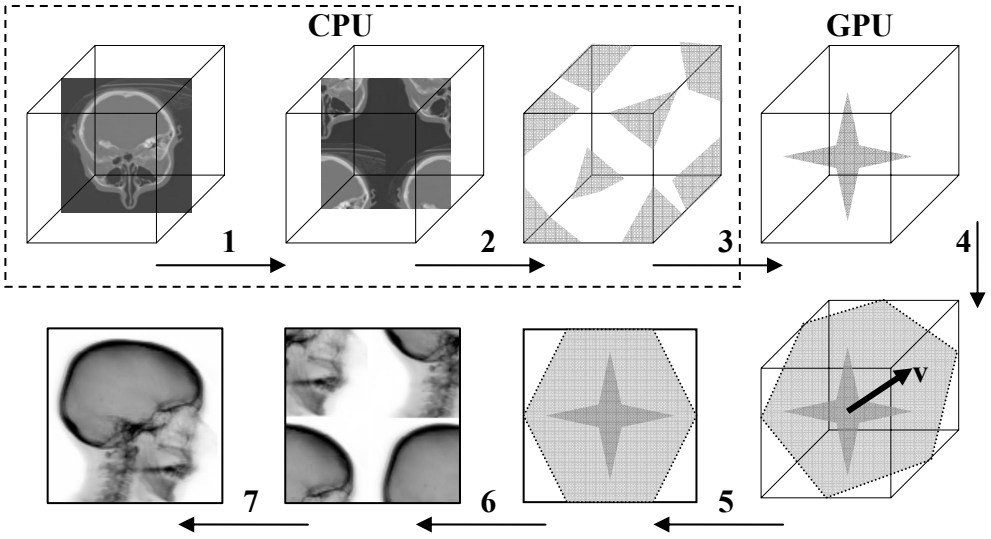


Fig. 5. The block diagrams for FVR on GPU.

4.2 The Implementation of the Bézier Curve Transfer Function

We used the Bézier curve of six control points to implement the transfer function for FVR. The formula of this curve is as the follows:

$$B(s) = t_0(1-s)^5 + 5t_1(1-s)^4 s \;\; + 10t_2(1-s)^3 s^2 + 10t_3(1-s)^2 s^3 + 5t_4(1-s)\;\; s^4 + t_5 s^5 \qquad (12)$$

Substituting Equation (12) into Equation (10), we obtained Equation (13) that is our shading function in our implementation.

$$
\begin{aligned}
I = {} & t_0 \Pi_v (f(x)(1-f(x))^5) + \\
& t_1 \Pi_v (5 f(x)(1-f(x))^4 f(x)\ ) + \\
& t_2 \Pi_v (10 f(x)(1-f(x))^3 f(x)^2) + \\
& t_3 \Pi_v (10 f(x)(1-f(x))^2 f(x)^3) + \\
& t_4 \Pi_v (5 f(x)(1-f(x))\ f(x)^4) + \\
& t_5 \Pi_v (f(x) f(x)^5)
\end{aligned}
\tag{13}
$$

4.3 Results

We developed a GUI (Graphical User Interface) application software system for the FVR and the proposed transfer function design. The shape of the Bézier curve can be modified by the input peripheral, such as a mouse, and the rendered result after application of the transfer function can be shown on the screen immediately. Some results are shown in the following figures. The left side of each figure is the rendered result by applying the transfer function shown on the right side. The performance is 8 fps by using the ATI X850 GPU in the case of the tri-linear filter was employed. The size of volume data is 128^3.

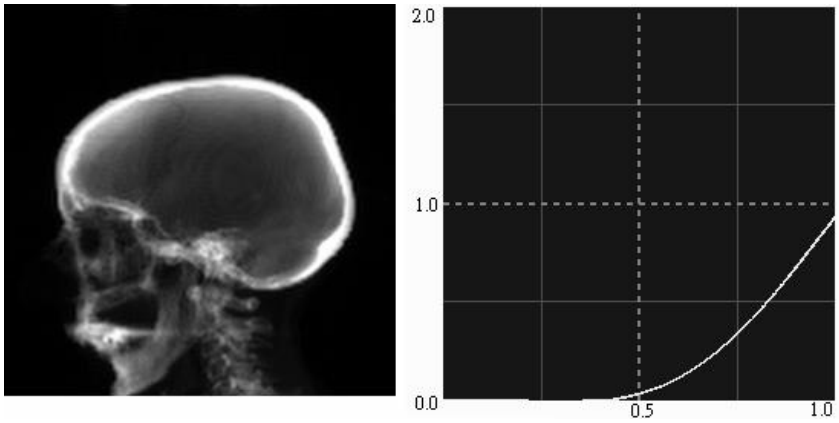


Fig. 6. Bones (CT Head).

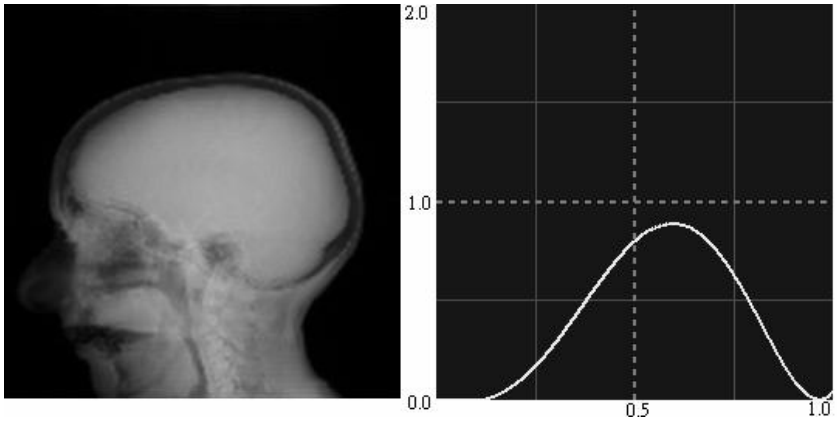


Fig. 7. Soft tissue (CT Head).

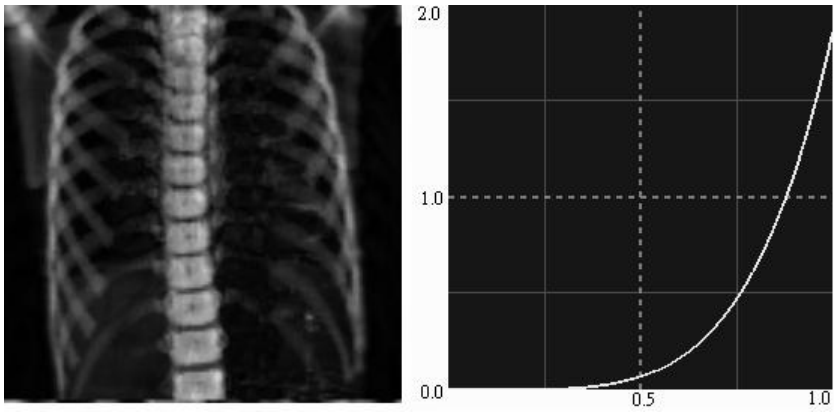


Fig. 8. Bones (CT Chest).

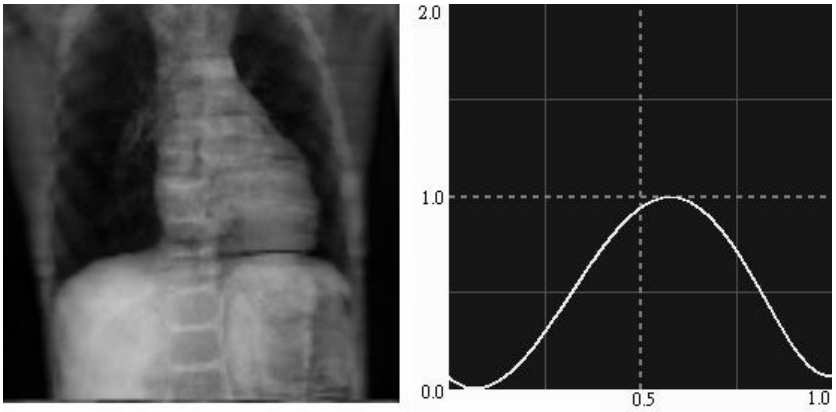


Fig. 9. Soft tissue (CT Chest).

5. CONCLUSIONS AND FUTURE WORKS

We have described the Fourier volume rendering method and designed the transfer function for voxel classification. We used the Bézier curve transfer function and the linear combination property of the Fourier transform to achieve the real-time rendering once the transfer function is modified. In contrast with other transfer function for FVR, the Bézier curve transfer function has much more convenient and easier implementation.

There are problems with FVR. The computing time required for sampling a slice in frequency data is more than the computing required for the inverse 2D Fourier transform. A more efficient implementation for sampling a slice in frequency data is required in order to improve the performance. Another problem is with the memory requirement for FVR. GPU usually has limited memory space. In our approach, zero padding increased volume size. Furthermore, there were 6 control points in the proposed Bézier curve transfer function design method. We need six times more memory space than the original FVR. The Hartley transform can reduce memory consumptions but still not enough. To design a memory efficient FVR algorithm is our future work.

ACKNOWLEDGEMENTS

This work was support under the grant NSC-95-2221-E-009-24, National Science Council.

REFERENCES

1. W. E. Lorensen, H. E. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm", *Proceedings of ACM SIGGRAPH Computer Graphics '87*, Vol. 21, 163-169, 1987.
2. M. Ikits, J. Kniss, A. Lefohn, C. Hansen, "Volume Rendering Techniques", *GPU Gems*, chapter 39, 667-692, Addison Wesley, New York, USA, 2004.
3. C. C. Lin, Y. T. Ching, "A note on computing the saddle values in isosurface polygonization", *The Visual Computer*, vol. 13, no. 7, 342-344, 1997.
4. S. Hill, "Tri-linear Interpolation", *Graphics Gems IV*, chapter X.1, 521-524, Morgan Kaufmann, San Francisco, 1994.
5. N. Max, "Optical Models for Direct Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1 , Issue 2, 99-108, 1995.
6. S. Dunne, S. Napel, B. Rutt, "Fast Reprojection of Volume Data", *Conference on Visualization in Biomedical Computing '90*, 11-18, Atlanta, Georgia, USA , 1990.
7. T. Malzbender, "Fourier Volume Rendering", *ACM Transactions on Graphics*, Vol. 12, Issue 3, 233–250, 1993.
8. M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, Vol. 8, Issue 3, 29-37, 1988.
9. M. Levoy, "Volume Rendering using the Fourier Projection-Slice Theorem", *Proceedings of Graphics Interface '92*, 61-69, Vancouver, B.C. Canada, 1992.
10. T. Totsuka, M. Levoy, "Frequency Domain Volume Rendering", *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 271-278, Anaheim, California, USA, 1993.
11. K. Engel, M. Kraus, T. Ertl, "High-quality pre-integrated volume rendering using hardware-accelerated pixel shading", *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware*, 9-16, Los Angeles, California, USA, 2001.
12. E. B. Lum, B. Wilson, K. L. Ma, "High-Quality Lighting and Efficient Pre-Integration for Volume Rendering", *Proceedings of the Joint EUROGRAPHICS - IEEE TVCG Symposium on Visualization*, 25-34, Konstanz, Germany, 2004.
13. Z. Nagy, G. Miiller, R. Klein, "Classification for Fourier volume rendering", *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, Vol. 0, 51-58, Seoul, Korea, 2004.
14. J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", *Math. Comput. 19*, 297-301, 1965.
15. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edition, The MIT Press, Cambridge , Massachusetts London, England, 2001
16. T. Jansen, B. von Rymon-Lipinski, N. Hanssen, E. Keeve, "Fourier Volume Rendering on the GPU using a Split-Stream-FFT", *Vision Modeling and Visualization (VMV) 2004, 9th International Fall Workshop*, 395–403, Stanford California, USA, 2004.
17. M. Frigo, S. G. Johnson, "The Design and Implementation of FFTW3", *Proceedings of the IEEE*, Vol. 93, Issue 2, 216–231, 2005.
18. M. J. Harris, G. Coombe, T. Scheuermann, A. Lastra, "Physically-based visual simulation on graphics hardware", *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 109-118, Saarbrucken, Germany, 2002.
19. J. Kr¨uger, R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms", *SIGGRAPH 2003: International Conference on Computer Graphics and Interactive Techniques*, 908-916, San Diego, California, USA, 2003.
20. K. Moreland, E. Angel, "The FFT on a GPU", *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 112–119, San Diego, California, 2003.
21. R. N. Bracewell, *The Fourier Transform and Its Applications*, 3rd edition, McGraw-Hill, New York, 1999.
22. B. B. A. Lichtenbelt, "Fourier Volume Rendering", *HP Labs Technical Reports*, 1995.
23. L. K. Arata, "Tricubic Interpolation", *Graphics Gems V*, chapter III.3, 107-109, Morgan Kaufmann, San Francisco, 1995.
24. R. J. Rost, *OpenGL Shading Language*, 2nd edition, Addison Wesley, New York, 2006.
25. E. Persson, *Framebuffer Objects*, ATI Technologies, Inc.
26. I. Viola, A. Kanitsar, M. E. Gröller, "GPU-based frequency domain volume rendering", *Proceedings of the 20th spring conference on Computer graphics*, 55-64, Budmerice, Slovakia, 2004.

27. T. Theußl, R. F. Tobler, E. Gröller, "The Multi-Dimensional Hartley Transform as a Basis for Volume Rendering", *The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision '2000*, vol. 1, 132–139, Plzen - Bory, Czech Republic, 2000.
28. M. A. Westenberg, J. B. T. M. Roerdink, "Frequency Domain Volume Rendering by the Wavelet X-ray Transform", *IEEE transactions on image processing*, Vol. 9, Issue 7, 1249-1261, 2000.
29. A. Entezari, R. Scoggins, T. Moller, R. Machiraju, "Shading for Fourier Volume Rendering", *Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, 131- 138, Boston, Massachusetts, USA, 2002.
30. M. Artner, T. Möller, I. Viola, M. E. Gröller, "High-Quality Volume Rendering with Resampling in the Frequency Domain", *Proceedings of EuroVis*, 85-92, Leeds, UK, 2005.