Proceedings of the 16th International Conference on Automation & Computing, University of Birmingham, Birmingham, UK, 11 September 2010

Improved Iso-surface Extraction for Hybrid Rendering Application

Qian Xu, Marcus Holder, Jing Wang, Zhijie Xu
CGIV Research Group, University of Huddersfield
Huddersfield, West Yorkshire, United Kingdom
q.xu@hud.ac.uk, z.xu@hud.ac.uk

***Abstract* - Over the last 2 decades, constructing expert scene graphs to integrate volume rendering with other modeling techniques has attracted increasing attentions. This paper represents design of a flexible conversion between volume and wireframe models. It starts with classifying the volume data sets for maintaining the accuracy of this conversion. Two computer graphics algorithms, which work as indispensable components of conversion, will be explained subsequently. Based on this design, low interactive rate, complicated data processing and lots of artifacts will be abated. It is anticipated that this conversion will enhance the advantages of implementing volume visualization on consumer-grade platform.**

Keywords-scene graph; volume model; wireframe model; interaction; hybrid rendering

I. INTRODUCTION

Stemming from traditional computer graphics researches, volume rendering and visualization have been growing into an important research field since the 1980s. Compared with conventional 3D modelling and visualization techniques, Volume Rendering (VR) enables a visual representation of data sets such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scans. VR allows for direct access to the internal structure of a 3D object instead of only showing the surface features.

In existing three dimensional modeling programs, most CAD tools such as Auto CAD and ProE are based on wireframe (or surface) models, which provide the visual representations of 3D objects using lines and vertices. The applications of surface modeling techniques can extend from rapid prototyping in virtual manufacturing and medical imaging, to highly complex astronomical and atmospheric simulations. With the ever increasing capacity of modern graphics hardware and the maturing methods for accelerating volume-based operations, real-time human interactions with complex combinations of different models on consumer grade PC hardware has becoming a research hot-spot in the last decade.

As the demand increases for high-resolution results in visualization applications like medical imaging and design verifications, there are inevitable needs for obtaining more original information and improving the speed of online visualization and manipulation process in complex 3D scenes. This trend has resulted in a rapid growth in the size of data sets to be processed by rendering systems in a more tightly controlled time frame.

In the existing VR-based applications, the lack of robust innovations in volume rendering and visualization makes users suffer from low interactive rate, complicated data processing and visual artifacts. In order to solve these problems, researchers construct new combinations to cover the disadvantages of volume rendering and visualization. For example, Natalya and Jeremy roughly integrated Direct Volume Rendering (DVR) with Indirect Volume Rendering (IDVR) to achieve the special medical simulation [1]. The IDVR-based parts were pre-calculated and stored. For the opaque surface of IDVR, they chose volume raycasting (a type of DVR) to represent the context and made IDVR parts overlap the same part (DVR-based) directly. The use of storage processes and the rough overlap are the two disadvantages of their design. These problems will decrease the interactive rate and bring artifacts.

The project started from an investigation of volume and surface modeling mechanisms on PC platforms: this work has already been carried out [2]. In this paper, the main investigations are the construction of volume segmentation, implementing a type of conversion of voxels into vertices.

The paper is organized in the following order: Section II provides a brief review of the research into various applications which were based on voxels, vertices or hybrid rendering modes. Section III presents the application of a 2D clustering algorithm in a 3D scene. The implementation of 'Marching Cubes' (MC), which is a type of computer graphics algorithm, is introduced in Section IV. Section V focuses on the results of various functional experiments carried out. Section VI concludes the research with evaluation of experiments and plans for the future work.

II. LITERATURE REVIEW

A. *Various Manipulations of Voxels and Vertices*

In every polygon-based application, each object consists of several primitives for representing freeform figures and surfaces. In order to accomplish these complicated constructions of primitives, the Constructive Solid Geometry (CSG) method was invented to implement these constructions by performing basic Boolean operations [3, 4]. The development of CSG models has been investigated by many researchers and most developed methods provide the direct display of CSG objects.

The CSG method can also be extended to process volumetric units as CSG primitives in the application of volume visualization and rendering. These models are named volumetric CSG (or VCSG) models, and are potentially useful for managing the combinations of multiple volumetric objects, or maintaining the accuracy of the products of vertices-based operations [5,6]. In addition, voxelization and volume rendering techniques are also used for CSG or VCSG models [7]. These techniques utilize the beam-oriented voxelization algorithm, the volume sampling approach, the point sampling algorithm, the octree-based rendering algorithm, or the distance volume algorithm [8,9]. These algorithms generally reconstruct a volume for the entire CSG or VCSG model in several forms, which will be rendered by a volume rendering algorithm. Since these volume reconstruction algorithms for CSG or VCSG models suffer from low interactive rates, the VCSG technique has not yet provided effective support to interactive volumetric modeling. In my research, a similar style of VCSG models will be created with solving these problems.

B. *Iso-surface Extraction*

VR renders the 3D objects without an intermediate boundary representation. In the related researches, Westerman and Ertl implemented a method for texture-based rendering of volume data sets defined on a uniform regular grid [10]. They also proposed a generalized method for rendering volumes defined over tetrahedral grids [11].

While these techniques are limited to rendering applications, an alternate class of methods extracts an intermediate iso-surface (vertices-based) that can be used for further manipulations and processing, such as collision detection, shadow casting, and animation [1]. The most commonly used algorithms for iso-surface extraction are derivatives of the Marching Cubes algorithm [12, 13]. Most of these algorithms construct the iso-surface following the '15 unique cube configurations' [3].

Iso-surface extraction is a compute-intensive method [1, 14]. For the increasing raw data size, it is hard to maintain the interactive rate by implementing iso-surface extraction on CPU platform. As a result, GPU-based iso-surface extraction has been a research hotspot for the last several years. The aim of these investigations is to generate highly efficient visualization and rendering at high frame rates. In the former generation of GPUs, the lack of programmable primitives processing and the inability to output geometric quantities for later operations was a direct cause of low frame rates and the poor quality of visualizations. As a result, previous GPUs lacked the ability to generate a polygonal surface directly on the GPU and use this for subsequent computation such as collision detection or optimizing volumetric surrounding organs. Most of the extraction work was redundantly performed regardless of whether the iso-value was dynamically changing or not, resulting in wasted computation.

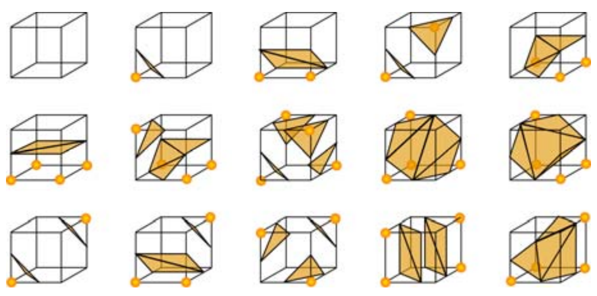
Figure 1. 15 unique cube configurations [3].

In order to improve the current problems, a partial conversion between voxels and vertices is proposed to increase the low interactive rate, minimizing the workload of data processing in visualization or rendering loops in the application. The aims of this design are:

- Applying clustering method for analyzing the framework of volume data sets;
- Carrying out the partial iso-surface extraction and converting the chosen clustered voxels into vertices utilizing MC algorithm;
- Managing the groups of vertices for future voxelization.

III. CLUSTERING-BASED VOLUME SEGMENTATION

Abstracting structures from measured volume data, i.e. to separate the volumes into its component objects (or regions), is one of the most essential goals in VR [15, 16]. When dealing with medical data, it can visually divide and selectively extract the objects of interest in the volume data sets. Explicit object membership information will be captured by implementing volume segmentation. Besides separating volumes, the processes of segmentation assign one or several segmentation masks to corresponding objects. By utilizing these masks, each voxel can be determined and rendered according to related optical properties.

The volume segmentation methods applied in this project so far are all based on extending 2D segmentation/clustering techniques into the 3D domain. The clustering approaches are applied because of their efficiency and robustness [17, 18]. In addition, the Transfer Function (TF) is used to convert the meaningless information into visible output for rendering the outcome of segmentation. In this paper, all the results of volume segmentation have been rendered by TF.

A. *Applying Clustering Methods into 3D Applications*

The clustering methods intend to sort the separated elements according to predefined spectrums. In the case of volume segmentation, voxels own similar signatures to pixels which are the basic element of the image. As a result, the volume segmentation can benefit from 2D-based methods such as the K-Means (KM) clustering approach without changes on the foundational mathematic model. The main difference from pixel-based operations is determining the extra dimension in the 3D feature space. Image processing methods usually refer to

2D space coordinates and the colour value of pixels. Consequently, the feature space generated is a 5D space (x, y, r, g, b), in which (x, y) demotes the space coordinates and (r, g, b) the colour of the pixel. These five elements represent a single point x_i in the feature space. In the case of volume segmentation, the feature will become a 6D space defined as (x, y, z, r, g, b), where (x, y, z) demotes the space coordinates and (r, g, b) the colour of the voxels. The pixel will be replaced by the voxel as the discrete element. The identical clustering methods can then follow suit.

As shown in Fig. 2, the pipeline of clustering method-based volume segmentation includes three processes: Data Input, Data Clustering and Judgment. The introduction of clustering mainly consists of the related explanations of these three processes. However, the clustering methods only take charge of clustering the volume data sets. As a result of this, the outcome of segmentation is colourless. In order to exhibit the effects of the clustering method, the dedicated TF was designed to variegate the results.

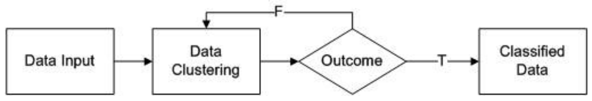

Figure 2. The framework of data clustering process in KM clustering method.

B. KM Clustering-based Volume Segmentation

a) Data Input: Besides the volume data set (x_1, x_2, x_3,…x_n), the predefined parameters contains "K" which represents the number of clustered data sets (K<n) and the fixed value V_k (used in the Judgment Process).

b) Data Clustering: This process is an iteration which contains three sub-processes: Clustering, Calculation and Comparison (as shown in Fig. 3). In the data sampling process, the values of all volume data will be obtained and form a threshold. The volume data set will be firstly clustered by dividing the whole threshold into K equal parts. The second process is to calculate the respective mean value for each part, which can be expressed as:

$$m_i = \frac{1}{|s_i|} \sum_{x_j \in s_j} x_j , \left(j \in [1, n], i \in [1, k] \right) \tag{1}$$

where m_i denotes the mean value and S_i represents the clustered data set. According to the calculated mean values (m_i), all elements will be clustered again, i.e. the constructions of classification will be rebuilt. In the comparison process, the new clustered data sets strictly contain the elements which own the closest value to the associated mean values (m_i). This clustering work can be expressed as:

$$s_i = \left\{ x_j : \|x_j - m_i\| \le \|x_j - m_l\|, \left(l \ne i, \forall l \in [1, k] \right) \right\} \tag{2}$$

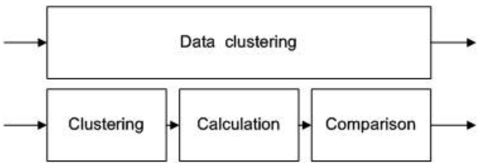

Figure 3. The framework of data clustering process in KM clustering method.

c) Judgement: The Judgement process acts as an indicator for managing the iteration in each clustering method. It is implemented to determine the iteration by comparing V_k with the real-time second derivative of unfixed mean values (v_m) [19]. For the condition ($V_k < v_m$), the iteration will be carried on. On the contrary, the latest clustered data sets is the outcome of the KM clustering-based segmentation.

IV. IMPLEMENTATION OF MARCHING CUBES

A. Marching Cubes

Marching Cubes traverses all cubes in one grid (one voxel or several voxels) and checks whether the corresponding eight vertices are equal in sign [20, 21]. If not, the iso-surface intersects the voxel (or voxels) and the algorithm generates a patch which consists of four triangles. The triangle configuration is indexed in a lookup table, which this time contains $2^8 = 256$ entries (as shown in Fig.1). It is difficult to make a definite description of some cases within the 15 configurations without labelling the eight vertices [3]. As a result, checking the sign inside the cube (tri-linear interpolation) is used to avoid scenarios of ambiguous description [22].

"One of the advantages of Marching Cubes is that it is local: cubes are processed one-by-one based on local information only (function values at cube corners)" [23]. Therefore, the processes in marching cubes can be processed parallelized. Implementing marching cubes on the GPU can provide a faster iso-surface extraction than CPU-based implementation. The results are shown in Fig.4

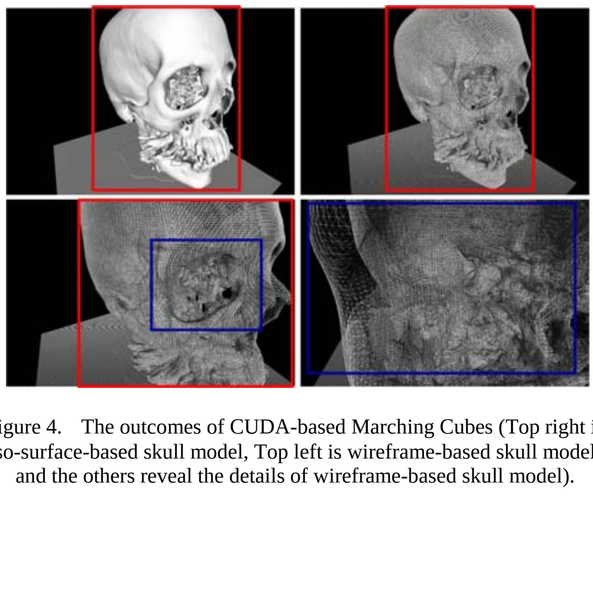

Figure 4. The outcomes of CUDA-based Marching Cubes (Top right is iso-surface-based skull model, Top left is wireframe-based skull model, and the others reveal the details of wireframe-based skull model).

B. *GPU-based Marching Cubes*

In marching cubes, the domain in IR^3 over which F is defined is tessellated into a grid at an arbitrary sampling density [24]. For each edge e = (x_0, x_1) in the tessellation, F (x_0) and F (x_1) will be evaluated according to the intermediate value theorem, e.g. if the signs of F (x_0) and F (x_1) differ, an iso-surface vertex must intersect e.

The edge lookup tables are typically stored in SIMD-specific memory for efficient and coherent access by each shader program instance [25, 26]. On the CPU, constructing large tables can lead to additional register pressure. This problem would be solved by reducing the number of parallel threads simultaneously running on the GPU. Smaller lookup tables ensure a higher order of parallelization because the GPUs are able to schedule a higher number of threads due to a higher number of available registers [27, 28].

V. PROTOTYPE IMPLEMENTATION

The program devised in this research has been implemented using OpenGL and CUDA APIs in a VC++ programming environment. The host PC is an Intel Core2 2.40GHz CPU, NVIDIA GeForce GTX260 with 2G RAM.

This paper mainly focuses on the implementation of volume segmentation and marching cubes. As shown in Fig. 5, the input of the system is raw volume data sets without any data preprocessing. The data sets will be clustered by implementing 3D KM clustering methods. Then, the clustered data sets are respectively processed in the iso-surface extraction function module and converted into groups of vertices. The generated vertices will be used to form rapid CSG operations with other wireframe models for special simulations, such as clipping, forging and bending processes. This design will avoid calculating and processing the whole data sets for save the processing time, i.e. increasing the interactive rate.

In Fig. 6, various volume data sets were represented in IDVR and converted into vertices-based models. Fig. 8 shows the different outcome of KM clustering-based volume segmentation. By setting the TF, the colorless results of volume segmentation are colorized with different K. In Fig. 7, image (A) is similar to image (B) which demonstrates that the maximum number of clustered regions is fixed with predefined properties in the clustering methods.

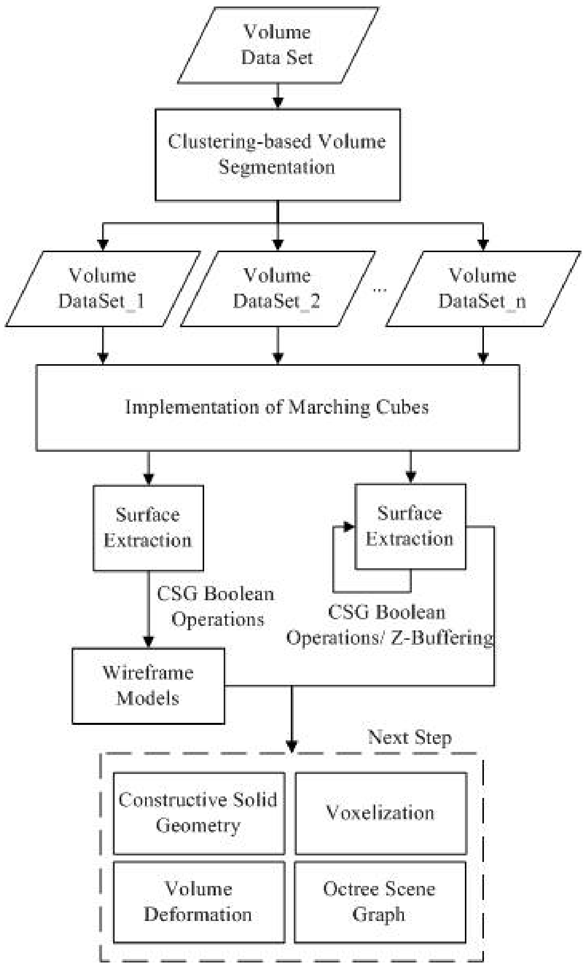

Figure 5. The framework of hybrid rendering system.

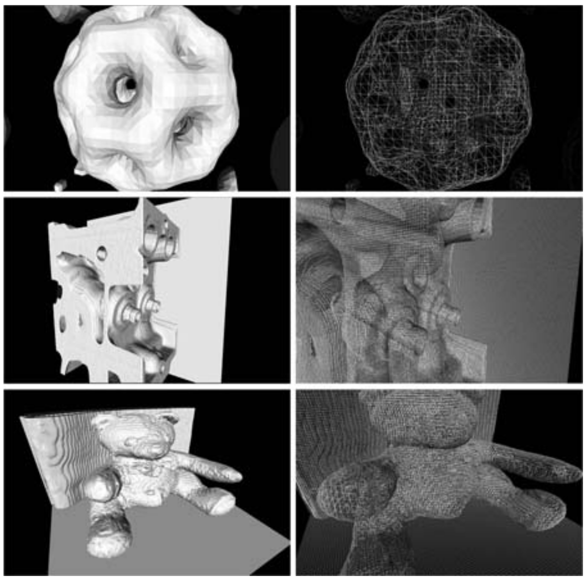

Figure 6. Various results of iso-surface exraction.

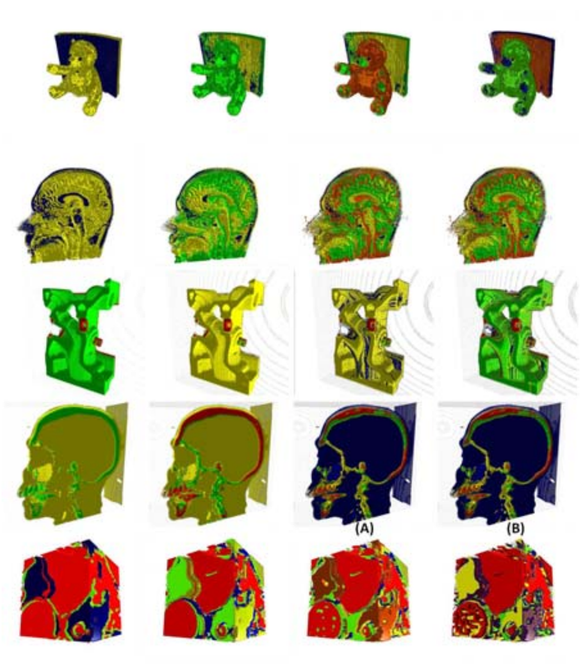

Figure 7. The outcomes of clustering-based segmentation (colored by TF).

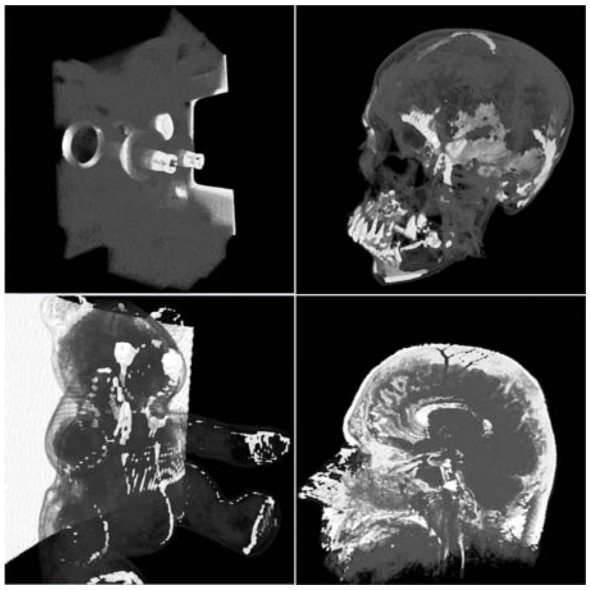

Figure 8. Partial iso-surface extractions.

Fig. 8 shows the outcomes of the implemented system. The opaque and white parts are IDVR-based and represent the regions which have different intensity or density values. The semi-transparent parts belong to DVR and represent the voxels which have the same intensity of density values.

VI. Evaluation and Future Work

By using clustering method to classify the volume data sets and implementing the iso-surface extraction, the polygonization of partial volume data sets and GPU-based implementations for future complicated operations (CSG and voxelization) have been accomplished.

However, the manual input of K in the KM clustering method will have a bad influence on the interactive rate of real-time operations. The inherent problems in clustering methods (over/insufficient segmentation) still cause artifacts in the final output.

These problems will be fixed by modifying the buffering area (e.g. ignoring some regions which contain a fixed number of vertices/voxels). After completing this step; advanced operations on generated vertices will be carried out. For the capability of rapid prototyping, a common vertex-based operation will be constructed between generated vertices and other surface models to implement fast CSG operations between volume and surface models. In future work, an improved volume deformation will also be constructed based on this conversion between voxels and vertices.

Acknowledgment

I would firstly like to thank my first supervisor Dr Zhijie Xu for his great supervision and guidance during this research. I am deeply impressed by his erudition, and knowledge and attitude to science which inspired me to keep on going. I would also like to show my grateful appreciation to Mr. Jing Wang and Mr. Marcus Holder for their great helps to me.

References

[1] N.Tatarchuk, J.Shopf, and C.Decoro, “Advanced Interactive Medical visualization on the GPU”, Journal. Parallel Distribub.Comput, Vol.68, pp.1319-1328, 2008.

[2] Q.Xu, and Z. Xu, “A Hybrid Rendering Framework for Real-time Manipulation of Volume and Surface models ”, Proceedings of the 15th International Conference on Automation & Computing, pp.161-167,2009.

[3] A.Requicha, “A Representation for Rigid Solids: Theory, Methods and Systems”, Computing Surveys 1980, vol.12, pp.437-464.

[4] A.Rappoport, S. Spitz, “Interactive Boolean Operations for Conceptual Design of 3D Solids”, SIGGRAPH ’97, pp.267-278.

[5] S.Fang, and R.Srinivasan, “Volumetric VSG: A Model-based Volume Visualization Approach”, Proceedings of the Sixth International Conference in Central Europe on Computer Graphics and Visualization, 1998, pp.88-95.

[6] N.Shareef, and R.Yagel, “Rapid Previewing via Volume-based Solid modeling”, Proceedings of the Solid Modeling ‘ 95, pp.281-292.

[7] S.Wang, and A.Kaufman, “Volume-sampled 3D Modeling”, IEEE Computer Graphics and Applications, 1994, vol.14, pp.26-32.

[8] D.Breen, “Constructive Cubes: CSG Evaluation for Display using Discrete 3D scalar Data Sets”, Proceedings of the Eurographics, 1991, pp.127-142.

[9] D.Breen, S.Mauch, and RT.Whitaker, “3D Scan Conversion of CSG Models into Distance Volumes”, Proceedings of the 1998 IEEE/ACm Symposium on Volume Visualization, 1998, pp.7-17.

[10] R.Westermann, and T.Ertl, “Efficiently Using Graphics Hardware in Volume Rendering Applications”, SIGGARPH ’98, Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, pp.169-177.

[11] S.Rottger, M.Kraus, and T.Ertl, “Hardware-accelerated Volume and Iso-surface Rendering based on the Cell-projection”, VIS ’00, Proceedings or the Conference on Visualization ’00, IEEE Computer Society Press, pp.109-116.

[12] W.E.Lorensen, and H.E.Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Computer Graphics, Proceeding of SIGGARPH 87, 1987, vol.21, pp.163-169.

[13] P.Shirley, and A.Tuchman, “A Polygonal Approximation to Direct Scalar Volume Rendering”, SIGGARPH Comput.Graph, 1990, Vol.24, pp.63-70.

[14] V.Pascucci, “Iso-surface Compution Made Simple: Hardware Acceleration, Adaptive Refinement and Tetrahedral Stripping”, Proceedings of VisSym, 2004.

[15] T.Ertl, and S.Fery, “Accelerating Raycasting Utilizing Volume Segmentation of Industrial CT Data”, EG UK Theory and Practice of Computer Graphics, 2009, pp. 1-9.

[16] D.S.Ebert, C.J.Morris, P.Rheingans, and T.S.Yoo, “Designing Effective Transfer Functions for Volume Rendering from Photographic Volumes”, IEEE Transactions on Visualization and Computer Graphics, 2002, vol. 8, pp. 183-197.

[17] A.Ahmad, and L.Dey., “A k-mean clustering algorithm for mixed numeric and categorical data”, Data Knowl. Eng., 2007, vol.6, pp. 503-527.

[18] K.L.Wu, and M.S.Yang, “Mean shift-based clustering”. Pattern Recogn., 2007, vol.40, p. 3035-3052.

[19] S.Ben-David, D.Pal, and H.U.Simon, “Stability of K-means Clustering”, Proceedings of te 20th Annual Conference on Learing Theory, 2007, pp.20-34.

[20] K.Engel, H.M., J.M.Kniss, A.E.Lefohn, C.R.Salama, and D.Weiskopf, “Course Notes 28 II: Real-Time Volume Graphics. Special Interest Group on Graphics and Interactive Techniques”, 2004.

[21] M.Botsch, and S.-D.S.A., The lecture notes on “Computer Graphics I" held by Prof. Dr. Leif Kobbelt at RWTH Aachen. 2003.

[22] A.Lopes, and K.Brodlie, “Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing”, IEEE Transactions on Visualization and Computer Graphics, 2003, vol.9, pp.1077-2626.

[23] E.Andres, P.Hehlig, and J.Francon, “Tunnel-free Supercover 3D Polygons and Polyhedra”, Computer Graphics Forum, 1997.

[24] B.Cabral, N. Cam, and J.Foran, “Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware”, Proceeding of the 1994 IEEE Symposium on Volume Visualization, 1994, pp.91-98

[25] S.Gibson, “Using Distance Maps for Accurate Surface Representation in Sampled Volumes”, Proceedings of the 1998 IEEE/ACM Symposium on Volume visualization, pp. 23-30.

[26] D.Banks, and S.Linton, “Counting Cases in Marching Cubes: Toward a Generic Algorithm for Producing Substitopes”, Proceedings of the 14th IEEE Visualization 2003, pp.8

[27] Z.L.Wang, J.C.M.Teo, C.K.Chui, S.H.Ong, C.H.Yan, S.C.Wang, H.K.Wong, and S.H.Teoh, “Computational biomechanical modelling of the lumbar spine using marching-cubes surface smoothened finite element voxel meshing”, Computer Methods and Programs in Biomedicine, 2005, vol.8, pp.25-35.

[28] F.Goetz, T.Junklewitz, and G.Domik, “Real-time Marching Cubes on the Vertex Shader”, Proceedings of Eurographics, 2005.

[29] F.R.A.Hogood, R.J.Hubbold, and D.A.Duce, “Advances in computer graphics II”, 1986, pp.0-186.