Proceedings of the 17th International Conference on Automation & Computing, University of Huddersfield, Huddersfield, UK, 10 September 2011

Volume Deformation Based on Model-Fitting Surface Extraction

Qian Xu, Duke Gledbill, Zhijie Xu
CGIV Research Group, University of Huddersfield
Huddersfield, West Yorkshire, United Kingdom
q.xu@hud.ac.uk, d.gledhill@hud.ac.uk, z.xu@hud.ac.uk

***Abstract* - Over the last decade, visualization techniques for 3-dimensional volumetric models, especially those that can be performed on PC hardware platforms, have attracted intensive attention in the research communities. The rapid evolution on PC computers, specialist hardware, and even gaming consoles have accelerated this trend and seen the volume model-based applications being greatly extended from industrial design, medical simulation, to entertainment usage and beyond. As part of the effort, the interactive manipulation of the appearance of volume models, often referred as volume deformation, has become a research hot-spot due to its potentials in revealing the models' internal structures and material characteristics. This paper reports an innovative volume deformation method based on a self-extracting mechanism for the so-called "control lattice" from the "surface" of a volume model, which can then be applied on the entire model or a specifically segmented part of it based on user requirements. The detail level of the extracted control lattice can be customized based on Active Surface algorithms for ensuring the interactive rate and the final resolution for a particular application.**

Keywords- Volumetric Model; Volume Deformation; Control Lattice; Adaptive Segmentation; Active Surface

I. INTRODUCTION

For showing the internal and implicit information of a 3-dimensional (3D) volumetric model, various PC-based visualization techniques have been developed in the past decade, such as Ray Casting and Splatting. However, for applications such as medical operation planning and design function simulations, volume models need to be "operable" in a rigid manner like splitting or slicing, as well as in a non-rigid form such as elastic deformation.

Volume data sets are often obtained from special industrial cameras and medical scanners, or being generated by mathematical models and algorithms. These models do not come with a mesh/polygonal representation of the compositing voxels, a synonym for volumetric-pixels, nor containing any intrinsic topological information of the models' internal structures and "material" properties - if applicable.

Most of the popular volume visualization techniques performed on PC-grade hardware assemble 3D models through accumulating the voxel contributions to the final "pixel" colours on the 2D virtual image plane. Although first proposed in the 1980s, volume visualization and its applications had really started gathering momentum since the late 1990s attributing to the ever more "powerful" PC capabilities. Volume deformation, as a branch of the trend, is stemmed from the visualization progression and had been focusing on the manipulation of volume models in a pre-defined or free-form manner and their corresponding control mechanisms.

Generally speaking, volume deformation approaches can be classified into two groups: rigid and non-rigid; while the prior can be considered as an extension of the Computational Solid Geometry (CSG) model with its mathematical representations defined by works such as Computational Volume Geometry (CVG) [1]. The latter approach preserves the underlying volumetric assemblies while enabling the description of the physical constraints and relationships among those elements. In the actual volume deformation applications, this approach can really meet the requirements of FFD (Free Form Deformation) which rigid ones cannot easily achieve [2, 3].

Unlike rigid deformation, the non-rigid approaches mostly require a control lattice (analogy to a web of "control" vertices) to be created for each operation before any displacement computations can start. For example, DOGME (Deformation of Geometric Models Editor) method in physically-based simulations, which is a constrained-based method, generally assumes a lattice which is the finest choice and passes the computed displacements to the underlying model or elements in the form of polygonal topologies [4]. For achieving the special visualization objectives, few non-rigid approaches totally rely on the dedicated rendering methods with corresponding calculation and interpolation works. For example, ray-deflectors-based methods, which do not move any voxels, manage to exhibit the deformed results by changing the visualization properties [5]. Based on the various requirements of the precision and usage in actual applications, the non-rigid approaches can be further classified into physically-based and empirical methods [6].

In the applications of physically-based volume deformation, a control lattice can be of a 2D planner form, or a 3D cubic or cylindrical form, or even being parametrically defined, which can represent material properties of the sampled voxels [7, 8]. However, due to the complex process often involved in the construction of a control lattice, and the shape and tessellation of it, the overall quality of the deformation varies.

In the surveyed volume deformation pilots, a small difference in the lattice definition strategy can lead to substantially varied voxel displacement results [9, 10]. Generally, a volume model is defined without any intermediate boundary representation. Westerman and Ertl

have developed a method for texture-based rendering of volume data based on a uniform regular grid as well as over a tetrahedral grid, which partially resolved the problem [11]. However, these techniques are focused on rendering aspects rather than interactive manipulations and precise simulations.

An alternative approach is to extract an intermediate iso-surface (polygonal mesh) from the volume data set that can be used for further manipulation and processing, such as collision detection, shadow casting, and animation. The most commonly used algorithms for iso-surface extraction are mainly derivatives of the Marching Cubes (MC) algorithm developed by Lorensen and Cline [12], which construct the iso-surface generation mechanism following the renowned "15 unique cube configurations" [13].

In this project, an innovative volume deformation approach is proposed based on an embedded MC process for constructing the so-called "model-fitting" control lattices. This paper will cover the Snake-based volume data orientation and the assistant design for MC-based lattice extraction in Section III and IV. The design and implementation of GPU-based Octree data structure are introduced in Section V with the deformed results.

II. SYSTEM DESIGN

In this research, a volume deformation pipeline based on the latest programmable GPUs has been developed as shown in Fig. 1. This paper mainly covers three sections: Data Segmentation, Mesh Modification and Spatial Relationships Determination.

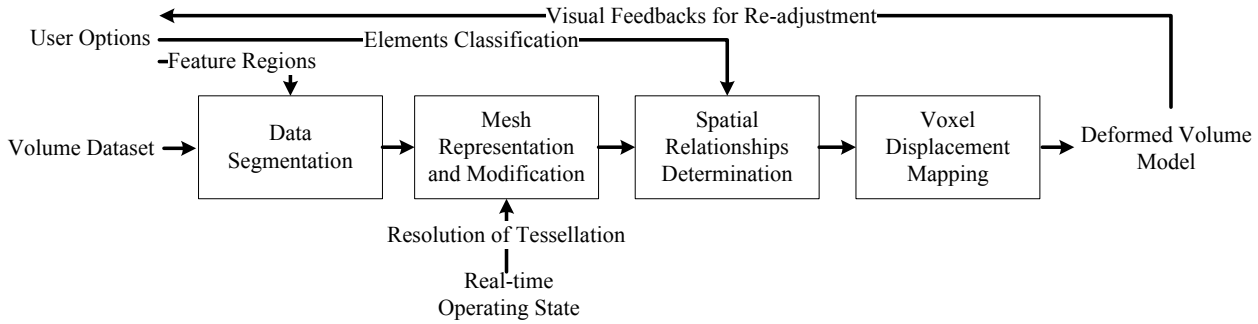


Figure 1. The pipeline of improved construction of lattice for volume deformation.

A. Data Segmentation Process

For reducing the workload of processing the entire data set in each processing cycle, a segmentation process has been designed to detect and track the "interested" objects which are mixed with other useless parts, for example, an organ in a MRI-scanned human body, or a section of the generated "point clouds". This step will reduce the overall memory "footprint" for storing and accessing the volume data set at runtime, and the complexity of the iso-surface extracted for forming the control lattice. Besides, implementing deformations on respective classified data sets can really support a shape-changed volume model containing different grades of transformation in the physically-based deformation applications.

B. Mesh Representation and Modification Process

However, the classified data set still leads to a time-consuming iso-surface extraction on CPUs for MC mechanism. The system developed in this project applied a GPU-accelerated (Graphics Process Unit) iso-surface extraction method for alleviating this problem. Both CPU-based and GPU-based MC processes often make extracted surfaces consist of far too many vertices to be used as a control lattice, which is literally a large spring network. The prototype system contained an adaptive tessellation engine for adjusting the resolution of the lattice via changing the number of vertices on the iso-surface.

C. Spatial Relationships Determination Process

Similar to the DOGME method in traditional vertex-based applications, this project is delivering the displacement on a control point to underlying elements which are "preformed" by many masses in a designed mass-spring model. A dedicated Octree data structure was constructed to manage the mass partition stage and determine the internal relationships between voxels. By implementing this data structure on GPU, the mass accessing and the internal spring-like relationships can be more efficiently implemented.

III. VOLUME MODEL SEGMENTATION

Separating objects from backgrounds or from each other through segmentation, which is an important process in computer vision and image processing, can reduce the complexity of the subject studied with its applications often found in measuring object size, tracking vehicle movement and scientific visualization. The volume data segmentation operation devised in this project is mainly based on the Active Surface that is a 3D extension of Active Contour Theory.

Active Contour (or Snake) is a method that enables delineating 2D outlines from an image. This technique contains a "spline" which follows the energy minimization rule and can be deformed by "forces". The forces are determined by an assembly of intrinsic and extrinsic constraints. The intrinsic constraints are

determined by the material properties of the spline, such as mass distribution parameter and viscosity of neighbouring medium. The extrinsic constraints represent the external forces which links splines (lines or surfaces) to underlying elements (pixels or vertices). The Energy $E(V)$ can be calculated according to "forces". Kass et al. proposed the equation of Energy Minimization in 1988 [14]:

$$E(V) = \int_0^1 (E_{\text{int}ernal}(V) + E_{external}(V))dV \quad (1)$$

where $E_{\text{int}ernal}$ is the internal energy of the bended spline and $E_{external}$ serves as external energy acting on the spline. $E_{external}$ contains E_{image} represents the image force acting on the spline and E_{con} denotes external constraint forces defined by the user.

Active Surface is a 3D variation of the Active Contour technique. In this project, region-based Active Surface algorithm is used to analyze 3D data sets. Its mathematical model can be represented by a parameterized surface δ [15]:

$$\delta : \Omega^2 \to IR^3 \quad S(U,V) = [x(U,V), y(U,V), z(U,V)]^{\delta} \quad (2)$$

Where (U,V) determines the coming changes of surface δ. And the changed surface $S(U,V)$ can be represented via an assembly of moved vertices. The calculation of minimal energy in the sampling region δ is represented via equation:

$$E[\delta] = \varphi E_{smooth} + (1-\varphi)E_{region} \quad (3)$$

where φ is a pre-input parameter that weights the significance of smoothness term. For processing 3D data, Mille proposed the smoothness energy for calculating 3D data sets:

$$E_{smooth}[\delta] = \iint_{\Omega^2} \left\| \frac{\partial S}{\partial U} \right\|^2 + \left\| \frac{\partial S}{\partial V} \right\|^2 dUdV \quad (4)$$

Surface δ separates the object into an internal domain γ_{in} and an external domain γ_{out}. Based on the Chan-Vese model [16], the energy of 3D domain can be calculated in

$$E_{region}[\delta] = \iint_{\gamma_{in}} \mu_{in} dUdV - \iint_{\gamma_{out}} \mu_{out} dUdV \quad (5)$$

where μ_{in} and μ_{out} are intensity descriptors inside and outside the surface respectively.

At the beginning of the Active Surface segmentation process, there is a step for defining the "3D splines". Fig. 2 A_0 and B_0 illustrate the definition of rectangle-based splines in any two cross sections in the volume model. Based on the collection of 2D splines (Fig. 2 A_1-A_5 and B_1-B_5), the region-based Active Surface algorithm can be implemented for direct and continuous analysis of 3D data sets. Its results are point-based and processed in the following MC stage.

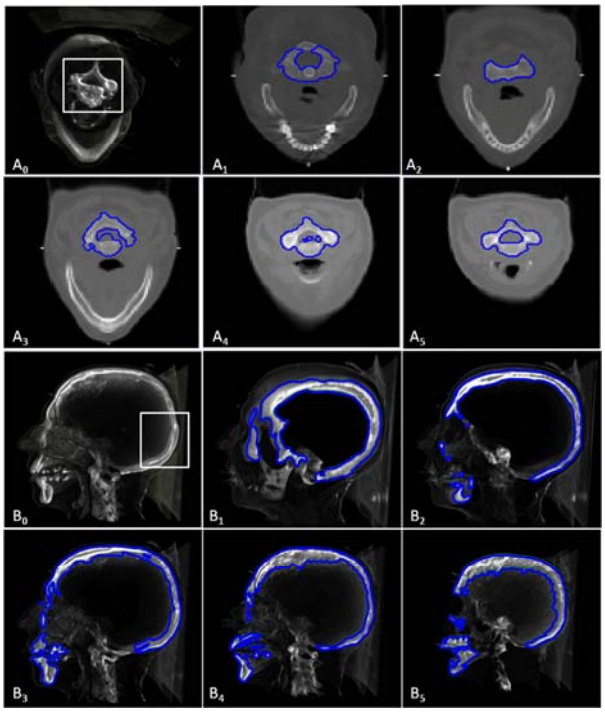


Figure 2. Image (A_0) and (B_0) respectively represent the ways of determining interesting-domains in axial and sagittal cross sections. Image (A_1-A_5, B_1-B_5) are snapshots of the detected regions.

IV. CONTROL LATTICE GENERATION

A. *Marching Cubes and Iso-surface Extraction*

In this project, MC is used to create the so-called "model-fitting" control lattice based on the iso-surface extracted from the volume data set. It travels through the volume model contained in an imaginary cube consisted of numerous "cells" as shown in Fig. 3 (a) and (b). The MC algorithm tests each voxel and produces vertices within the corresponding cells. The density values at the 8 vertices of a cell are evaluated based on the "signs" determined by their relative positions to the iso-surface. The voxels that lie on the boundary between the model and the "empty space" (voxels with null values) are then processed for generating polygons according to the standard "15 unique cube configurations" as shown in Fig. 3 (c).

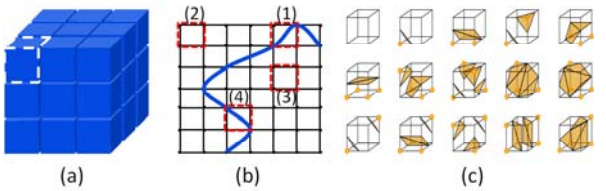


Figure 3. (a) represents the "imaginary cube" and (b) shows different conditions that happened in the sampling progress. (C) illustrates the standard "15 unique cube configurations."

Based on these fifteen basic configurations, there will be a total of 241 derived forms. It is difficult to make a definite description of these 256 cases without labelling the eight vertices. As a result, the tri-linear interpolation (checking the sign for inside of the cube) is used in this

project to avoid ambiguity. As stated by Engel [Engel et al, 2004], one of the advantages of the Marching Cubes algorithm is its locality - the voxels are processed one-by-one based on local information only. The computational processes for MC can be readily parallelized and "mapped" on GPUs for harnessing its data parallelism. Fig. 4 shows the output iso-surfaces from the segmented data sets in the devised program in the form of a closed polygonal mesh.

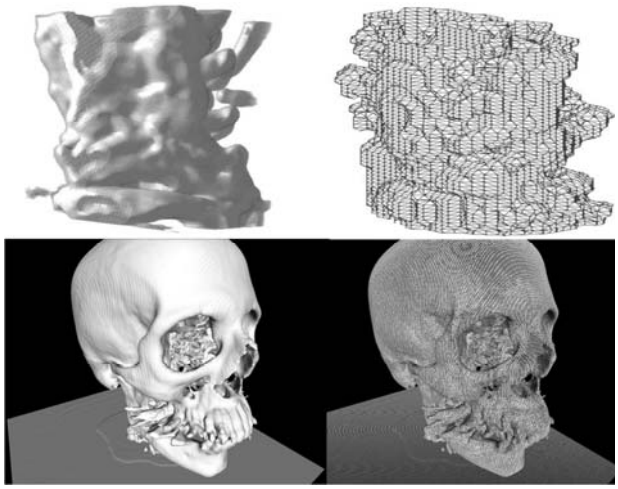


Figure 4. These isosurfaces are respectively extracted from the segmented throat data and skull data in a MRI-scanned human head.

B. GPU-based Adaptive Control Lattice

The surface extracted from the MC process contains a huge number of polygons, e.g. there are 2 million vertices in the iso-surface-based skull model in Fig. 4. This complicated mesh can be slow to interact with when used for calculating deformation parameters such as vertex and voxel displacement offsets and rotational angles. Through controlling the tessellation of the extracted iso-surface, an adaptive control lattice can be formed.

There are existing GPU models supporting hardware-driven tessellation operations that can be adopted for this purpose, e.g. GPU-based Catmull-Clark subdivision is available in mesh-based deformation for adapting the density of vertices in the deforming regions [17, 18]. It is used to implement the real-time distribution of vertices during the deformation, i.e. carry out LOD-based control for managing the number of vertices. However, in this research, the choice of data structure, derived operations and the consideration of its' familiar artefacts called T-junction determine that Catmull-Clark subdivision needs to be replaced by a triangle-based subdivision scheme – Loop Subdivision [19]. Its reverse scheme is chosen to achieve the More-to-Less operation for reducing the vertex count and implementing vertex management in an "on-the-fly" style.

The (reverse-) Catmull-Clark subdivision and (reverse-) Loop subdivision schemes are both based on the management of object vertices and their neighbouring points. Accordingly, implementing reverse Loop subdivision on GPU can be technically divided into three main steps. Firstly, the object mesh is separated into a series of sub-meshes according to the results of triangular partition (represented by the blue triangles in Fig. 5 Step 1). This process does supply outstanding vertices for the second step to find their surrounding points which share the same lines with the objective vertices. After this finding process, the second step carries on labelling all participant vertices for GPU-based vertex storage in the third step. For all vertices stored in texture memory in the form of 2D arrays with the default index numbers (λ in Fig. 5 Step3), the single-threading-management of vertices in the CPU-based Loop subdivision can be implemented more efficiently via the GPU-based multithreading-operations that support synchronized vertex-release processes and the real-time combination of processed arrays. For conveniently understanding the implementation of GPU-based Loop subdivision in this project, the schematic meshes are all ideal and the valence of each vertex is 6. The sketch maps of each reverse subdivision process and simplified results are all respectively illustrated in Fig. 6 and 7. The polygon account is hugely reduced from millions to thousands, e.g. from 2.4 million vertices to 12 thousands in the skull model.

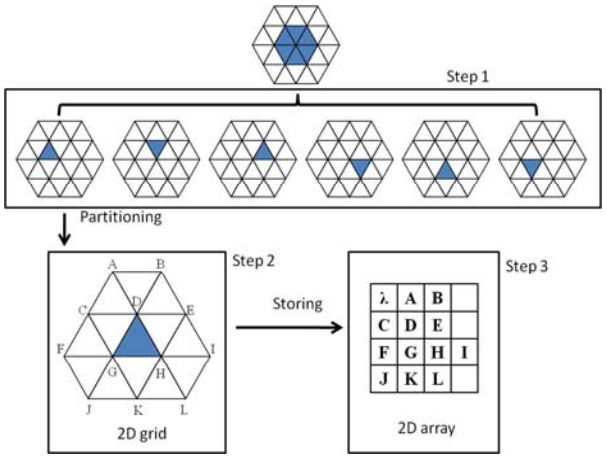


Figure 5. Illustrations for GPU-based reverse Loop subdivison.

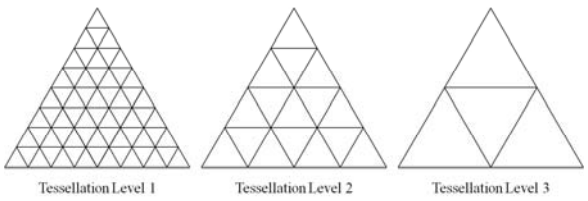
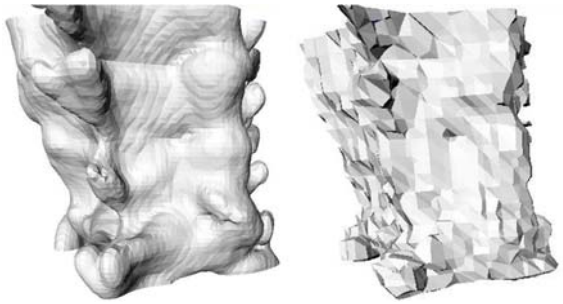


Figure 6. Continous construcntion of a triangle primitive with decreasing tessellation levels.



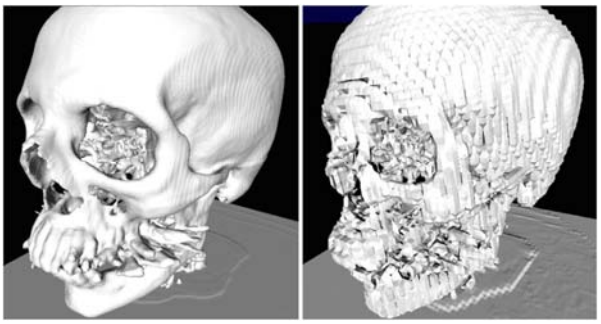


Figure 7. The illustrations of adaptive MC process in different tessellation levels.

V. VOLUME DEFORMATION

After finishing the model-fitting lattice construction, this research uses octree data structure to separate the volume data set for implementing voxel-based masses and parameterized elastic characteristics in the designed mass-spring model. The implementation work mainly consists of three processes. First one is "encoding" the volume data set for GPU-based storing process. Similar to other applications of GPU-based 3D octree data structure, the whole volume model is equally divided into an assembly of elements which generally contain $2^N*2^N*2^N$ voxel(s) (N is the depth of octree), and the partitioned results are stored in an 8-bit RGBA 3D texture in the form of N^3-tree. The coordinates of each subset or element are stored in RGB channels and the Alpha channels record the results of identifying a pointer to the content of a leaf or a child node. 0, 1 and 0.5 respectively indicate pointing to an end of the current branch (voxels with null values), an available child node and an indexable content.

After storing process, the second process is tracking the object node's related leaf contents or nodes on different levels (3D representations are shown in Fig. 8). In order to accomplish this goal, Tree-Lookup function, which is a common module in various applications of GPU-based Octree data structure, determines the resulting contents or nodes based on the choice of Alpha value of each element on the current level. The results are a unit cube or a cubic region with mathematical expression $[O_0, O_1]^3$.

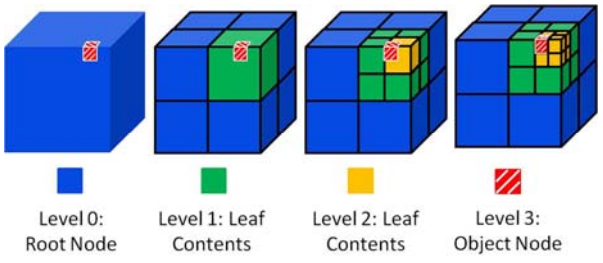


Figure 8. Octree-based method for tracking the objective area or element (represented by the yellow or red square).

The 3rd process uses these detected contents or nodes to represent different grades of transformation in the physically-based deformation of segmented parts. In this project, the mechanism is proportionally assigned different scales of original displacement (contains distance and direction) to corresponding contents or nodes. The related outcomes are an assembly of spatial coordinates. In Fig. 9, the different grades of deformation are carried out via decreasing the displacement along the track (dashed) of the moved control point (solid point). Meanwhile, a simple indicator is added to survey the points value to testify that the fixed number of points ensure no manmade or interpolated artefacts.

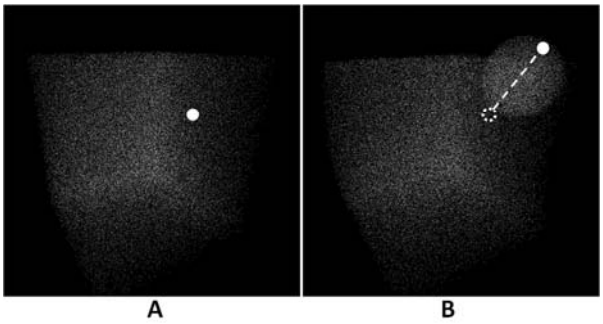


Figure 9. A point-based representation of a volume model (A); the control point (solid point in (A)) obtains a displacement and the deformed result is shown in (B).

The current system is designed and constructed using MATLAB, OpenGL, and CUDA APIs under the Visual Studio/VC++ programming environment. A desktop PC with an Intel Core2 2.40GHz processor and an NVIDIA GeForce GTX260 GPU with 2GB memory have been used for experiments. Because no new voxels are created, the following work is directly following the initial spatial arrangement rule to "filling" original voxels to the new assembly of coordinates extracted from the Octree data structure. Fig. 10 has shown the corresponding result of Fig. 9 in (a) and other derived deformation outputs (reversed and multiple) are shown in (b) and (c).

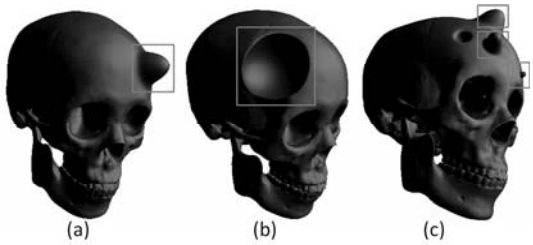


Figure 10. The appearance of various skull deformations.

In Fig. 11, these vertical-sections illustrate the deformed internal/external regions in a human skull model. The effects of physically-based deformation in my project, which need to represent the details of volume deformations with different distributions of displacement in different materials, had been highlighted by the white rectangles.

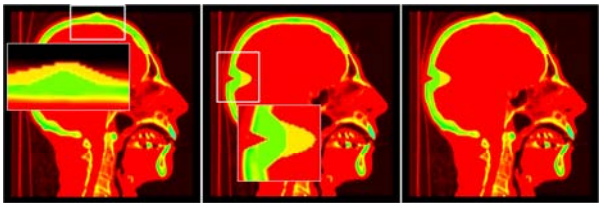


Figure 11. Lower line illustrates the corresponding vertical-sections.

And the frame-rates are totally distributed between 49 and 67 (higher than 30 fps required in actual applications). Therefore, these records can demonstrate the feasibilities of interactive manipulations and multiple deformations in my physically-based volume deformation.

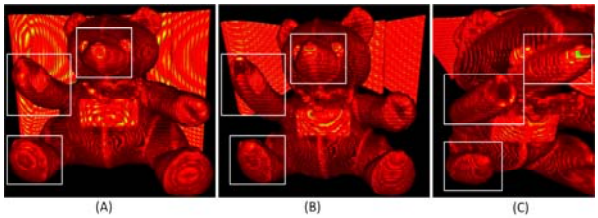


Figure 12. Texture-operation-based volume deformation.

VI. CONCLUSIONS

In Fig. 12, B and C show two deformed teddy-bears which are generated via stretching the initial model (A) in two directions. This deformation method belongs to the texture operation-based volume deformation which is one branch of the non-physically-simulated applications. Its mechanism is moving the carriers to influence the corresponding elements, i.e. modifying textures to transfer voxels. In current volume deformation applications, non-physically-based methods are barely used for simple displaying, colourful rendering or conceptual expiations. In order to achieve imprecise simulations, they must require helps from the complicated segmentation algorithms to carry out certain texture pre-processing for fixing its lack of element-level-operation. Consequently, these designs always suffer from the time-consuming processes, resulting artefacts in texture pre-processing and low interactive rates.

The novel physically-based volume deformation pipeline developed in my project can manage the basic elements (voxels) for achieving complicated simulations and related requirements. It has replaced the conventional methods for constructing control lattices using an improved model-fitting MC-extracted surface mesh. The latest GPU tessellation capacity adopted in this research has further enabled hardware-driven complexity control that optimized the frame of the extracted mesh. Experiments carried out in the research have shown improvements on the accuracy and flexibility of deformation types that can be performed using this design. It is anticipated that applications such as medical operations, simulations, indusial designs, and even computer games can benefit from this innovative volume deformation approach. Future work will focus on further improvements for system integration and real-time performance.

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 Dr Duke Gledhill for his great help to me.

REFERENCES

[1] M. Chen and J. V. Tucker, “Constructive Volume Geometry,” In Computer Graphics Forum. Vol. 19, No. 4, pp. 281-293, 2000.

[2] N. Chen, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk and J. F. O’Brien, “Interactive Simulation of Surgical Needle Insertion and Steering,” In ACM Transactions on Graphics. Vol. 28, No. 3, pp. 1-10, 2009.

[3] C. Corea, D. Silver and M. Chen, “Feature Aligned Volume Manipulation for Illustration and Visualization,” In IEEE Transactions on Visualization and Computer Graphics. Vol. 12, No. 5, pp. 1069-1076, 2006.

[4] B. Sarvage, S. Hahmann, G. P. Bonneau and G. Elber, “Detail Preserving Deformation of B-spline Surface with Volume Constraint,” In Computer Aided Geometric Design. Vol. 25, No. 8, pp. 678-696, 2008.

[5] Y. Kurzion and R. Yagel, “Interactive Space Deformation with Hardware-assisted Rendering,” IEEE Computer Graphics and Applications. Vol. 14, No. 5, pp. 66-77, 1997.

[6] C. D. Corea, D. Silver and M. Chen, “Constrained Illustrative Volume Deformation,” In Computers & Graphics. Vol. 34, No. 4, pp. 370-377, 2010.

[7] M. Hong, S. Jung, M. Choi and S. W. J. Welch, “Fast Volume Preservation for a Mass-Spring System,” In IEEE Computer Graphics and applications. Vol. 26, No. 5, pp. 83-91, 2006.

[8] G. Yin, Y. Li, J. Zhang and J. Ni, “Soft Tissue Modeling Using Tetrahedron Finite Element Method in Surgery Simulation,” Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering. Nanjing, China, pp. 3705-3708, 2009

[9] R. Westermann and C. Rezk-Salama, “Real-time Volume Deformations,” In Computer Graphics Forum. Vol. 20, No. 3, pp. 443-451, 2001,

[10] S. Walton and M. Jones, “Volume Wires: A framework for Empirical Non-linear deformation of Volumetric Data Sets”. In Journal of WSCG. Vol. 14, No. 3, pp. 81-88, 2006.

[11] R. Westermann and T. Ertl, “Efficiently Using Graphics Hardware in Volume Rendering Applications,” Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. Orlando, USA, pp. 169-177, 1998.

[12] W. Lorensen and H. Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm,” In ACM SIGGRAPH Computer Graphics. Vol. 21, No. 4, pp. 163-169, 1987.

[13] K. Engel, M. Hadwiger, J. M. Kniss, A. E. Lefohn, C. R. Salama and D. Weiskopf, “Course Notes 28 II: Real-Time Volume Graphics,” ACM SIGGRAPH 2004 Course Notes, pp. 1-282, 2004.

[14] M. Kassm, A. Witkin and D. Terzopoulos, “Snakes: Active Contour Models,” In International Journal of Computer Vision. Vol. 1, No. 4, pp. 321-331, 1988.

[15] J. Mille, “Narrow Band Region-based Active Contours and Surfaces for 2D and 3D Segmentation,” In Computer Vision and Image Understanding. Vol. 113, No. 9, pp. 946-965, 2009.

[16] T. Chan and L. Vese, “Active Contours without Edges,” In IEEE Transactions on Image Processing. Vol. 10, No. 2, pp. 266-277, 2001.

[17] E. Catmull and J. Clark, “Recursively Generated B-Spline Surfaces on Arbitrary Topology Meshes,” In Computer Aided Design. Vol. 10, No. 6, pp. 350-355, 1978.

[18] M. Pharr and R. Fernando, “GPU Gems 2: Chapter 7. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping,” In GPU Gems2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. Printed in USA, 2004.

[19] C. Loop, “Smooth Subdivision Surfaces Based on Triangles,” Thesis for the degree of Master of Science, 1987.