Improvements of GPU Implementation of Nonlinear Soft Tissue Deformation with CHAI 3D

Yibo Shi, Hui Xiang and Dehai Yu

Abstract The contradiction between the speed and the accuracy of soft tissue deformation simulation has challenged developers for decades, especially for nonlinear deformation. In this paper, we propose improvements of the GPU implementation of nonlinear soft tissue deformation with CHAI 3D by decreasing the calculation scale and reducing data transfer between device and host. Soft tissue is divided into deformation area and non-deformation area, and points and tetrahedrons in the deformation area are calculated during the deformation. In addition, we redesign the collision detection of CHAI 3D and implement it with CUDA in parallel. By using VBO, OpenGL displays the soft tissue based on data resided in graphics memory directly. As a preparatory work of FEM, we also present a convenient method to divide the soft tissue into tetrahedrons by TetGen.

Keywords：GPU. Nonlinear FEM. TLED. Surgical simulation. CHAI 3D. TetGen

1 Introduction

With the help of surgical simulations, the trainings of surgical practitioners become more efficient and convenient. Studies show that use of computer-based simulations can provide a more effective and systematic training, as well as an objective assessment of technical competence, and it is easy to transfer the skills learned from simulations into operating room [8]. However, these simulations require accurate and fast deformations to imitate realistic scenes, which is a formidable challenge [6]. In the past decades, researchers have dedicated a considerable amount of effort to model soft tissue deformation. But the widely used models are mass-spring model and finite element model [12].

The mass-spring model (MSM) can be fairly large and complex without sacrificing real-time response to user interaction. In addition, this model can handle topological changes at a low computational cost. However, for a successful simulation, the springs connecting a number of points should be designed carefully. And it is not stable in some situations [10]. The most crucial disadvantage is that it is difficult to relate the spring stiffness to material properties, such as Young modulus [4].

Another popular model is the finite element model (FEM). FEM overcomes most of shortcomings of MSM. However, a major drawback of FEM, especially for nonlinear ones, is that it requires a huge solution time to solve these equations in large models [3]. Thus, researchers proposed various acceleration methods. Bro-Nielsen et al. realized a real-time simulation of deformation with condensation technique, which confines the computation to the surface nodes of the mesh [2]. Berkley et al. developed a banded matrix technique for a finite element model in real-time deformation and force feedback [1].

In recent years, the rapid development of GPU provides another solution to deal with the complex computation within surgical simulations, as Sorensen and Mosegaard reviewed in [9]. The early GPU-based simulation focused on MSM, because of its simplicity and low computational cost. And it was demonstrated that MSM is well suited to benefit from GPU-based acceleration. But the GPU-based MSM inherits most of the defects from the traditional MSM. W. Wu and P. A. Heng developed a GPU-based linear FEM for surgical simulation, and the results showed this method could achieve a considerable speedup compared with equivalent CPU implementation [12]. However, their formulations are geometrically and constitutively linear. Miller proposed an efficient Total Lagrange Explicit Dynamic (TLED) finite element algorithm for nonlinear deformation, which could provide real-time simulations for moderately size models [7], Taylor extended Miller's method and designed a GPU-based version, and the results show that a midrange GPU can get a significant solution speedup (16×) compared with equivalent CPU implementations [11].

The rest of this paper is organized as follows: Section 2 gives an overview of our framework and algorithm. Section 3 introduces the discretization of the soft tissue model. The GPU implementation of the soft tissue deformation with CHAI 3D is presented in section 4. Section 5 introduces the improvements of the GPU implementation, followed by the results in section 6. Finally, the conclusions are drawn in section 7.

2 Overview of Framework and Algorithm

The framework of our simulation includes discretization of soft tissue model, model load, pre-computation, display loop and haptic loop. The discretization of a model discretizes the soft tissue model into tetrahedrons. It is independent from deformation calculation and is executed offline. CHAI 3D supports multi-thread simulation. The main thread runs the display loop while the second thread executes the haptic loop. TLED nonlinear finite element proposed by Taylor is used [11]. For handling the volumetric locking, the IANP method proposed by Miller is included [5].

We now present a brief introduction about the calculation steps, and further details may be obtained from [5].

In the pre-processing stage:

1. Load the model and prepare constants, such as mass matrix, initial element volumes and nodal volumes, the spatial derivatives of shape function;
2. Allocate the memory of CPU and GPU, transfer the data to GPU and prepare other auxiliary work.

While in the loop calculation stage, when the virtual device touches the soft tissue, the collision detection gets the triangle containing the collision point and the external force. Then the simulation executes the following loop:

1. Determine the deformation area;
2. For each tetrahedron in deformation area, calculate and save the product of the pressure and the volume of the tetrahedron;
3. For each point in deformation area, sum the pressure of tetrahedrons containing the point;
4. For each tetrahedron in deformation area, calculate the internal force of the points located in the tetrahedron;
5. For each point in deformation area, calculate the displacement according to internal and external forces, previous and current displacements and so forth；
6. Sum all the internal forces of points and save the sum as the internal force of the model；
7. If the internal force of the model is zero, the deformation is over; otherwise update the interval, and turn to step 2.

3 Prepare the Tissue Model

The first step of FEM procedure is to discretize tissue model into elements, such as tetrahedrons or hexahedrons. Compared with hexahedron, tetrahedron is superior in various aspects, for example, tetrahedron has a better approximation to surface of a model, and is easy generated automatically by tools or libraries.

In our work, tetrahedron is adopted and a library named TetGen is used. TetGen is a quality tetrahedral mesh generator and three dimensional Delaunay triangularization created by Hang Si. It provides the ability to compute an exact 3D Delaunay tetrahedralization. The original model created by 3D MAX is a surface model containing spatial and topological information of surface vertices. Because the 3ds file format is not supported by TetGen, we extract the spatial and topological information of the model form 3ds file and save them as an off file format. Then, the TetGen reads the off file, discretizes the surface model into tetrahedrons, and produces a tetgenio structure which is a collection of arrays of points, facets, tetrahedrons, and so forth. In order to represent a soft tissue, we need not only spatial and topological information, but also material properties, such as Lame constant and Poisson rate, and constraints, such as, fix constraints and boundary constraints. Thus, we organize all this information as nodes in a XML file.

4 GPU Implementation with CHAI 3D

To implement the TLED nonlinear FEM deformation with CHAI 3D, we need to implement a volumetric model, a surface model and a solver. The volumetric model contains all the points and tetrahedrons which are used for computing internal forces and displacements of points. The surface model is used for display and collision detection. The main work of the solver is to calculate internal forces and displacements of points by preparing the needed parameters and invoking appropriate kernel functions which are executed on GPU in parallel.

As described in calculation steps, each step containing "each point" or "each tetrahedron" can be organized as a kernel function. The whole process of deformation calculation consists of two parts: calculate internal forces of points and calculate the displacements of points. In order to compute the internal forces with IANP method, three kernels are used. The first kernel performs over tetrahedrons and computes a product of the pressure and volume of tetrahedron. The second kernel performs over points for computing the nodal pressure by collecting the products calculated in tetrahedrons containing the point. The third kernel performs over tetrahedrons, constructs modified deformation gradient and computes the internal forces of points. For using previous and current displacements and internal and external forces to compute the next displacements, we invoke the forth kernel which performs over points and computes the displacements according to the equation of equilibrium.

5 Improvements of Deformation Calculation

For improving the computational efficiency of deformation calculation, decreasing the calculation scale and reducing data transfer between GPU and CPU are employed in our study. When the virtual device touches the soft tissue, the deformation area is not the whole tissue but part of tissue neighboring with the collision point, which means the deformation area need be determined before calculating the deformation. In addition, the data transfer between GPU and CPU is very costly, which influences the computational efficiency of GPU. By using VBO and redesigning the collision detection of CHAI 3D, we eliminate most of data transfer between GPU and CPU.

5.1 Determine the Deformation Area

When the virtual device touches the soft tissue, the collision detection gets the collision triangle which contains the collision point. The points of the collision triangle are regarded as seeds. The flooding method is used to determine the deformation area. After being given the depth, the following steps will determine the deformation area:

1. Save the points of collision triangle as new added points, and set the current depth as one;
2. For each new added point, collect the tetrahedrons containing the point and denote them as candidate tetrahedrons;
3. For each candidate tetrahedron, if it is not saved, select it and save it as new added tetrahedron;
4. For each new added tetrahedron, check the points of the tetrahedron, if the point is not saved, select it and save it as new added point;
5. Update the depth, if the depth is smaller than the given depth, turn to step 2; otherwise, all the points and tetrahedrons in deformation area have been saved.

5.2 Interoperation between OpenGL and CUDA

In our work, the tissue deformation is calculated by GPU, and the results reside in graphics memory. Fortunately, with the help of the Vertex Buffer Object (VBO), we are able to display the soft tissue resided in graphics memory directly. After the related resources being allocated in preprocessing stage, the memory address of soft tissue in graphics memory can be gotten with help of CUDA API, which makes it possible to update the spatial coordinates with the calculation results by invoking a kernel function. Lastly, the shape of soft tissue is displayed with VBO.

5.3 Modify the Collision Detectioin in CHAI 3D

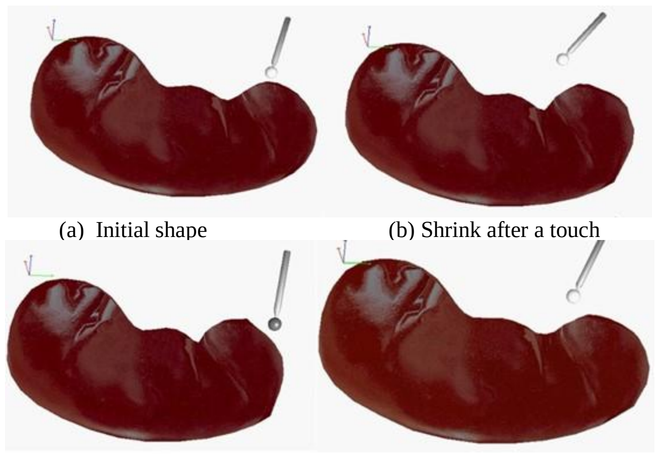
In order to detect the collision using the data in graphics memory, we should redesign the collision detection of CHAI 3D in parallel and implement it with CUDA. In the haptic loop, the previous and current spatial coordinates of the virtual device are recorded, denoted as A and B respectively. CHAI 3D detects whether the line segment AB collides with any triangle of the tissue in the haptic loop too. Because the collision detection between AB and a triangle is independent from the other triangles, we invoke a kernel function to detect the collision. The kernel function performs over each triangle of the tissue and detects collision between AB and the triangle. If the triangle intersects with AB, and the collision point is denoted as P, then the distance between A and P is saved in a array; otherwise, a big value representing the triangle does not intersect with AB is save in the array. In one collision detection, there may be several triangles intersecting with AB. Another kernel function is invoked to find the smallest distance using reduction algorithms in parallel. The triangle with a smallest distance of AP is regarded as the collision triangle.

5.4 Calculate the Reaction Force of Model

In CHAI 3D, when the virtual device applies an external force to an object, the virtual device will receive a reaction force from the object, and the reaction force is the reverse of the external force. That is valid in the rigid collision. In our simulation, when the virtual touches the deformed tissue, the reaction force consists of the internal force of the model and the reaction force of the external force. Thus, the internal force of the model should be calculated in the haptic loop. As described above, the internal forces of points are saved in array. In order to calculate the internal force of the model, a kernel function is invoked to sum up these internal forces using reduction algorithms in parallel, as the way to find out the collision triangle.

6 Experiments

We experiment on a computer with Inter Core2 Duo 2.33GHZ CPU and 8GB memory, and code with VS2008, NVIDIA GeForce GTS 450 platform and CHAI 3D. A kidney model with 8,427 tetrahedrons is used. The Phantom Omni is used to control a drill to touch the kidney and feel the feedback. A simple but still nonlinear material model, that is the Neo-Hookean material model, is employed in our simulation. The material parameters are set as the same with the brain tissue [11]. The depth used to determine the deformation area is set as 5. Fig. 1 shows the shapes of the kidney during the deformation after a touch.



(a) Initial shape (b) Shrink after a touch

(c) Maximum shrink shape (d) Relaxation after the shrink

Fig. 1 The virtual device controlled by Phantom Omni interacts with the kidney

Table 1 describes the mean solution speed of one deformation calculation. The GPU version 1 means the GPU version without optimization, while the GPU version 2 means the GPU version with optimization. Table 1 shows that the computational efficiency of GPU version with optimization increases 20% compared with the original GPU version.

Table 1 The mean solution speed of a deformation calculation

CPU version (FPS)	***GPU version 1 (FPS)***	***GPU version 2 (FPS)***
25	1500	1800

7 Conclusion

This paper implements the GPU implementation of a nonlinear FEM for an interactive soft tissue deformation with CHAI 3D and proposes improvements by decreasing the calculation scale and reducing data transfer between GPU and CPU. Because the deformation area neighbors with the collision point, we solve the equations of equilibrium on points and tetrahedrons in the deformation area instead of the whole tissue. In order to reduce data transfer, it is necessary to make use of the data in graphics memory to display the soft tissue and detect the collision between virtual device and soft tissue. We also modify the calculation of reaction force in CHAI 3D to take the internal force of the soft tissue into consideration. The results show that after being optimized, the computational efficiency of GPU increases 20% compared with the original calculation. In addition, the haptic device can get a smooth feedback from the deformed tissue.

On the other hand, there are several deficiencies in our work. The elements of the soft tissue in our experiments are the same material type, thus they behavior consistently. Because the explicit finite element is employed, the solution time of one deformation calculation should be small enough to get a converged solution. It is demonstrated that the maximum solution time dependents on the material parameters used and the minimum characteristic element length in the mesh, which limits the scale of the tissue.

References

1. Berkley J, Weghorst S, Gladstone H, et al. Banded matrix approach to finite element modelling for soft tissue simulation [J]. Virtual Reality, 1999, 4(3): 203-212.
2. Bro‐Nielsen M, Cotin S. Real‐time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation[C]. Computer graphics forum. Blackwell Science Ltd, 1996, 15(3): 57-66.
3. Courtecuisse H, Jung H, Allard J, et al. GPU-based real-time soft tissue deformation with cutting and haptic feedback[J]. Progress in biophysics and molecular biology, 2010, 103(2): 159-168.
4. Delingette H. Toward realistic soft-tissue modeling in medical simulation[J]. Proceedings of the IEEE, 1998, 86(3): 512-523.
5. Joldes G R, Wittek A, Miller K. Non‐locking tetrahedral finite element for surgical simulation[J]. Communications in Numerical Methods in Engineering, 2009, 25(7): 827-836.
6. Miller K, Taylor Z, NOWINSKI W L. Towards computing brain deformations for diagnosis, prognosis and neurosurgical simulation[J]. Journal of Mechanics in Medicine and Biology, 2005, 5(01): 105-121.
7. Miller K, Joldes G, Lance D, et al. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation[J]. Communications in numerical methods in engineering, 2007, 23(2): 121-134.
8. Seymour N E, Gallagher A G, Roman S A, et al. Virtual reality training improves operating room performance: results of a randomized, double-blinded study[J]. Annals of surgery, 2002, 236(4): 458.
9. Sørensen T S, Mosegaard J. An introduction to GPU accelerated surgical simulation[M]//Biomedical Simulation. Springer Berlin Heidelberg, 2006: 93-104.
10. Sørensen T S, Greil G F, Hansen O K, et al. Surgical simulation–a new tool to evaluate surgical incisions in congenital heart disease?[J]. Interactive cardiovascular and thoracic surgery, 2006, 5(5): 536-539.
11. Taylor Z A, Cheng M, Ourselin S. High-speed nonlinear finite element analysis for surgical simulation using graphics processing units[J]. Medical Imaging, IEEE Transactions on, 2008, 27(5): 650-663.
12. Wu W, Heng P A. An improved scheme of an interactive finite element model for 3D soft-tissue cutting and deformation[J]. The Visual Computer, 2005, 21(8-10): 707-716.