COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2014; **25**:155–169
Published online 19 August 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1543

RESEARCH ARTICLE

Interactive deformation and cutting simulation directly using patient-specific volumetric images

Shuai Li[1]*, Qinping Zhao[1], Shengfa Wang[2], Aimin Hao[1] and Hong Qin[3]

[1] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China
[2] Dalian University of Technology, Dalian, Liaoning, China
[3] Stony Brook University, Stony Brook, NY, USA

ABSTRACT

This paper systematically advocates an interactive volumetric image manipulation framework, which can enable the rapid deployment and instant utility of patient-specific medical images in virtual surgery simulation while requiring little user involvement. We seamlessly integrate multiple technical elements to synchronously accommodate physics-plausible simulation and high-fidelity anatomical structures visualization. Given a volumetric image, in a user-transparent way, we build a proxy to represent the geometrical structure and encode its physical state without the need of explicit 3-D reconstruction. On the basis of the dynamic update of the proxy, we simulate large-scale deformation, arbitrary cutting, and accompanying collision response driven by a non-linear finite element method. By resorting to the upsampling of the sparse displacement field resulted from non-linear finite element simulation, the cut/deformed volumetric image can evolve naturally and serves as a time-varying 3-D texture to expedite direct volume rendering. Moreover, our entire framework is built upon CUDA (Beihang University, Beijing, China) and thus can achieve interactive performance even on a commodity laptop. The implementation details, timing statistics, and physical behavior measurements have shown its practicality, efficiency, and robustness. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
volumetric image manipulation; patient-specific medical simulation; physics-based modeling; CUDA
Supporting information may be found in the online version of this article.

***Correspondence**
Shuai Li, State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, 37 Xueyuan Road, Haidian District, Beijing 100191, China.
E-mail: ls@vrlab.buaa.edu.cn

1. INTRODUCTION AND MOTIVATION

In conjunction with the rapid advancement of various imaging modalities such as magnetic resonance imaging, computed tomography, and positron emission tomography, volume graphics-related [1] techniques are expanding rapidly and give rise to widespread applications in diagnosis, biology, and medicine. In particular, high-fidelity and efficient deformation/cutting simulation of soft tissues plays a vital role in virtual-reality-centric digital medicine [2]. However, although a great amount of volumetric medical datasets have been routinely collected and utilized in clinical applications, there are still tremendous difficulties throughout different stages of physics-based simulation, which prevent clinical volumetric images from being efficiently and extensively used [3]. For example, most of the geometrical mesh-based methods, ranging from simple spring-mass structure [4] to complex finite element representations [5], have little concern over the wide variety of patient-specific datasets, which typically adopt commonly used geometric models in practice and do not afford instant data utility and replacement [6].

Meanwhile, the strong demand for medical simulation has been evolving rapidly from being solely model-oriented to a higher level of being patient specific and data driven [7], which synchronously requires real-time exploration of the internal anatomical structures [8]. Many studies [9] have put great emphasis to narrow the gap between flexible manipulation of patient-specific volumetric image and its physically plausible simulation; however, there remain some technical challenges, which are documented as follows.

First, explicit geometry-modeling-based methods usually require 3-D reconstruction and post-processing (e.g., volume smoothing and denoising) [10,11]. Therefore, the

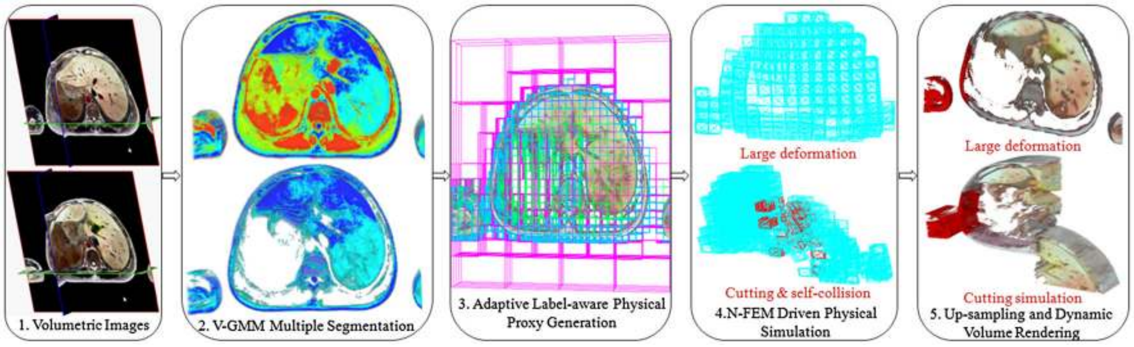

1. Volumetric Images
2. V-GMM Multiple Segmentation
3. Adaptive Label-aware Physical Proxy Generation
Large deformation
Cutting & self-collision
4.N-FEM Driven Physical Simulation
Large deformation
Cutting simulation
5. Up-sampling and Dynamic Volume Rendering

Figure 1. Our framework for volumetric image deformation and cutting simulation driven by non-linear finite element method (NFEM). First, we automatically conduct multi-label segmentation and select the target tissues with little user interaction. Second, we build a label-aware proxy in a user-transparent way to encode the physical properties. Then, we conduct physical simulation. At last, the volumetric image is driven to evolve by resorting to super-resolution of the sparse displacement field, which also serves as a dynamic 3D texture for the interactive volume rendering.

patient-specific modeling process is very time-consuming and requires tedious manual inputs. Besides, most of the reconstructed models are object specific, which precludes the realistic presence because of the omission of surrounding tissues [3,12].

Second, even with dynamically generated geometric meshes of the cutting surfaces, it is hard to real-time synthesize and map a physiologically plausible texture onto the cut surfaces, let alone the challenging task of representing the continuously variable anatomical structures. This unavoidably downgrades the role of patient-specific volumetric medical images in surgical rehearsal [13] and limits the simulation experience of surgeons/clinicians [14].

Third, although the voxel-based approaches can produce volume-editing results by transforming/sculpting/drilling the voxels in localized volume regions [15], it is not feasible to directly employ such methods for the complex manipulation of patient-specific medical images because of the abundant computation cost resulting from massive voxels. Meanwhile, voxel-based volume editing approaches are usually susceptible to global aliasing artifacts.

Fourth, because the constitutive behavior of organs/tissues is well known to be non-linear, it should be rigorously modeled with non-linear kinematic formulations [16]. And the topological update, collision response, user intervention, and visualization should also be tightly synchronized to accommodate cutting simulation, however, each of which is already very time-consuming. To achieve interactive efficiency, it calls for sophisticated algorithm integration and the unified parallel computation framework.

To tackle the aforementioned challenges, we focus on a unified CUDA-based framework toward directly utilizing the patient-specific volumetric medical images for instant simulation. Figure 1 intuitively illustrates the pipeline of our framework. Specifically, the salient contributions can be summarized as follows:

(1) We systematically articulate a versatile framework for direct volumetric image manipulation while avoiding rebarbative manual geometry processing, which seamlessly integrates the dominating processes of patient-specific images toward its direct utility in surgical simulation.
(2) We generate an adaptive label-aware physical proxy from volumetric images to drive the non-linear deformation and cutting simulation of soft tissues and design a fitting-based sparse displacement field upsampling algorithm for dynamic volume rendering of deformed/cut anatomical structures.
(3) We synchronously integrate cutting simulation, collision detection, and collision response into a non-linear finite element method (NFEM)-based deformable model and design a parallel algorithm for the explicit solving of the physical equations.
(4) We implement the framework within the parallel computing architecture of CUDA to guarantee the interactive efficiency, which can also contribute to other relevant medical applications.

2. RELATED WORK

Relevant to the central theme of this paper, we now briefly review previous work in three categories: volumetric image manipulation, FEM-based deformation, and FEM-based cutting simulation.

Volumetric Image Manipulation. Direct volumetric image manipulation has the advantage of faithfully preserving contents. Most of the earlier studies address this issue as geometry-based image morphing. Driven by the demand of arbitrary exploration of internal structures, some more complex geometric manipulation strategies were successively proposed [8,17]. For example, displacement maps [18,19] were widely used to perform non-interactive manipulation by means of pre-computation. While Masutani *et al*. [20] achieved interactive deformation and plausible manipulation by simply controlling the vertices of textured proxy objects. Takayama *et al*. [21] presented a method to represent volume cutting by repeatedly pasting spatially varying solid texture exemplars. However, the aforementioned methods generally have unacceptable disadvantage in speed and flexibility, which are more suitable for geometry-based volume-editing. Thus, such methods lack both physical realism and efficiency. Recently, Correa *et al*. [22] proposed an efficient volume deformation method by moving the proxy points placed on the cross sections to their desirable positions. Nakao *et al*. [3] extended the proxy points [22] to automatically generated proxy tetrahedral mesh according to the original image curvature, and then, they further refined the proxy mesh by taking adaptive optimization algorithm into account [7] in the pre-processing stage. However, it ignores the topological changes due to cutting. To avoid intractable remeshing, some meshless methods were also proposed to handle deformation [23] and fracturing [24], and Jin *et al*. [25] most recently proposed a meshless algorithm for image-based soft tissue manipulation, but currently, it can only afford 2-D cutting simulation.

FEM-based Deformation. Most of real-time FEM approaches are based on linear formulations. To enable large deformation, Dick *et al*. [5] extended the co-rotated strain formulation to hexahedral elements and achieved linear time complexity by introducing a multi-grid solver. However, its constitutive law remains linear. The total Lagrangian explicit dynamics (TLED) method proposed by Miller *et al*. [26] rigorously formulated the non-linear constitutive law, which was also employed as a deformable model to conduct non-rigid registration in [27]. By integrating the visco-hyperelastic model, TLED was extended to handle anisotropic viscoelastic deformation [28]. Despite the earlier success of TLED in large deformation simulation, it has not been generalized to handle cutting simulation. Most recently, similar to diffusive regularization in image registration, Fortmeier *et al*. [29] proposed to conduct volumetric soft tissue deformation by relaxing ChainMail algorithm on the inverse of the displacement field. However, it can only simulate local deformations caused by needle insertion.

Besides, given an FEM model, the deformation is governed by the underlying equations of motion, which is solved by employing either an explicit method with small time steps [27,28] or an implicit integration scheme [5]. To improve, Chao *et al*. [32] proposed a combination of a geometric material model with a fully variational geometric integrator, while Fierz *et al*. [33] showed how to identify ill-shaped elements hindering stable numerical time integration. Moreover, based on element-wise stability considerations, a hybrid methodology combining explicit and implicit linear integration approaches was also presented [30].

FEM-based Cutting Simulation. FEM-based cutting simulation remains challenging in dynamic topological update and realistic cutting surface visualization [34]. Finite element subdivision-based method [35] is usually used to handle topological update, but it can create severely ill-conditioned simulation elements. Wicke *et al*. [31] improved it by using arbitrarily convex finite elements, but it is even more involved in numerical integration. In sharp contrast, Molino *et al*. [36] proposed a virtual node algorithm to avoid element splitting by duplicating the cut elements and redistributing material components. However, because each fragment is required to contain at least one FEM node, arbitrary cutting is strictly limited. Most recently, adaptive regular hexahedron approximation is used to simulate the cut in linear elastic deformable bodies [5], and *Jerabkova et al*. [34] achieved interactive FEM-based cutting simulation by removing the finest level voxels from multi-resolution volumetric images. However, both of them ignored the synchronized collision handling and realistic visualization of cutting surfaces.

Brief Summary. Although most of the current algorithms are competitive, they are rather lonesome and still require a trade-off between physical (or visual) plausibility and speed. Therefore, a unified and efficient framework that affords instant image utilization, physics-based large-scale deformation, arbitrary cutting, synchronized collision handling, realistic cut surface texturing is urgently needed. This paper aims to elaborate a novel scheme toward this goal.

3. FRAMEWORK OVERVIEW

As shown in Figure 1, we first focus on the framework overview as follows.

Multi-label Tissue Segmentation. We design the volumetric Gaussian mixture model (V-GMM) algorithm to automatically conduct multi-label segmentation and choose the target tissues with little user interaction.

Physical Proxy Generation. In a user-transparent way, we successfully build a label-aware tetrahedral mesh according to the multi-label segmentation, which will serve as a proxy to encode the geometry structure and physical properties based on NFEM.

Physical Simulation. In each simulation cycle, on the basis of CUDA, we compute the internal and external forces according to user interaction and self-collision, update the proxy structure and its accompanying properties such as displacement, velocity, position, etc.

Dynamic Volume Rendering. Based on CUDA, the cut/deformed volume is driven to evolve by resorting to super-resolution of the displacement field resulted from proxy structure and serves as a dynamic 3-D texture for the interactive volume rendering.

4. PHYSICAL PROXY GENERATION

4.1. Multi-label Volumetric Image Segmentation

To analyze the physical structure of an organ and its surrounding tissues, we should first perform multi-label seg-

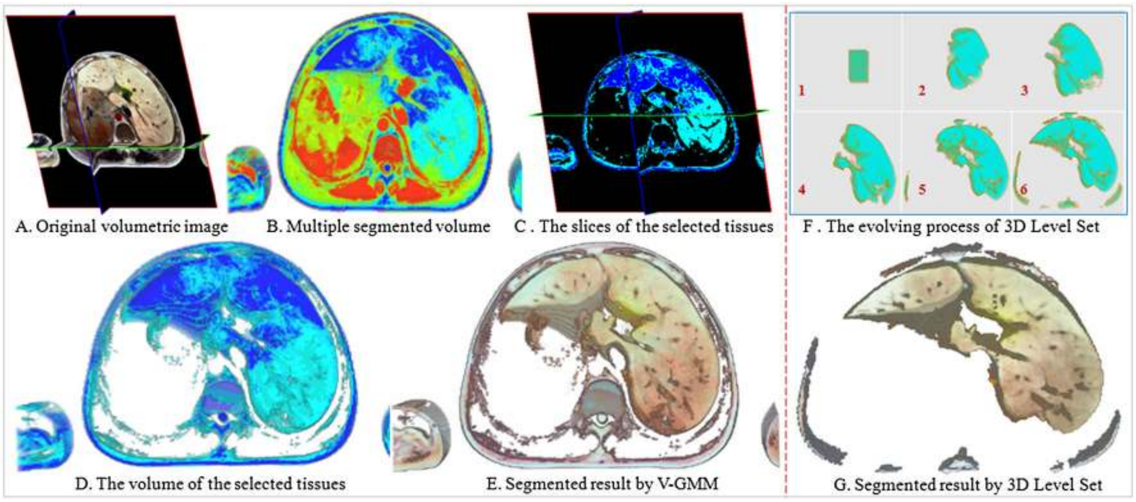

A. Original volumetric image
B. Multiple segmented volume
C. The slices of the selected tissues
F. The evolving process of 3D Level Set
1
2
3
4
5
6
D. The volume of the selected tissues
E. Segmented result by V-GMM
G. Segmented result by 3D Level Set

Figure 2. V-GMM-based multi-label segmentation of volumetric image. (A) is the input original volumetric image; (B) and (C) are respectively the volume view and slice view of multiple segmented image; (D) illustrates the selected tissues according to the multi-label segmentation result; (E) is the final result corresponding to (D); (G) shows the segmentation result of 3-D Level set for comparison, while (F) illustrates the evolving process of 3-D Level Set.

mentation on the original volumetric image, which is represented by a stack of 2-D image slices in 3-D space, and where the variation of tissue types gives rise to varying color. Because some disconnected regions might have the same labels due to their intensity similarity, in principal, accurate multiple-label segmentation results are hard to achieve in both automatic and semi-automatic algorithms with little user intervention. However, to achieve the goal of instant utility of patient-specific 3-D images, it is necessary to develop an automatic image segmentation method in our integrated framework. To the best of our knowledge, Gaussian mixture model (GMM) is a relatively better multi-clustering method, which has achieved great success in 2-D image segmentation.

Yet, GMM is still lacking high accuracy and efficiency if it is naively adopted to conduct volumetric image segmentation slice by slice, wherein specific segmented regions (organ and surrounding tissues) are also hard to be selected. Therefore, given patient-specific volumetric images, we extend GMM to V-GMM for automatic multi-label segmentation. Figure 2 demonstrates the automatic V-GMM-based multi-label segmentation result of volumetric image. On the basis of the multi-label segmentation results, we can further pick out the target organs and its surrounding tissues (Figure 2(F)) by manually indicating the color-coded segmentation labels. Therefore, the so-called target organs are some clusters manually selected from the multiple segmentation results by specifying the colors of the desired tissue types. Upon that basis, it further requires to manually assign the physical property type for each cluster, which are the only required manual inputs in our framework. It should be noted that V-GMM can be replaced by any automatic image segmentation method, because our framework's technical foci are on the generality and overall efficiency of the entire pipeline.

Even so, Figure 2(G) also provides the segmentation result of 3-D level set method for comparison. Meanwhile, Figure 3 lists more experimental results and their comparison, wherein the first column shows the original volumetric images, the second column shows the multi-label segmentation results by V-GMM, the third column shows the segmentation results of the target tissues that are manually picked out, and the 3-D level set-based segmentation results are shown in the fourth column for the comparison purpose. The results indicate that our V-GMM method is more accurate than the 3-D level set method for images with low color contrast, and 3-D level set method is hard to be employed for multi-label segmentation. As a preprocessing step of our framework, V-GMM can usually obtain the segmentation results for the volumetric image with the approximate size of $512^2 \times 85$ in tens of seconds.

4.2. Label-Aware Proxy Generation

On the basis of the multi-label segmentation results (Figure 4(A) and (B)), we continue to generate a proxy mesh to encode the physical topology and properties in a user-transparent way. As shown in Figure 4(C), we first hierarchically divide the working space by way of an adaptive octree. For any sub-cuboids at the current level, whether to perform further space partitioning is determined by the tissue types contained in the sub-space. And the material type (encoded with different colors in Figure 4(B)) of the sub-cuboid can be determined according to the voxel labels within it. We then generate a uniform tetrahedral proxy mesh by decomposing each leaf cuboid into five tetrahedra. To avoid edge intersection between neighboring tetrahedra that share the same cuboid face, we define two different partitioning schemes (shown in Figure 4(D)). By taking the 3-D adjacency relations of the leaf cuboid into

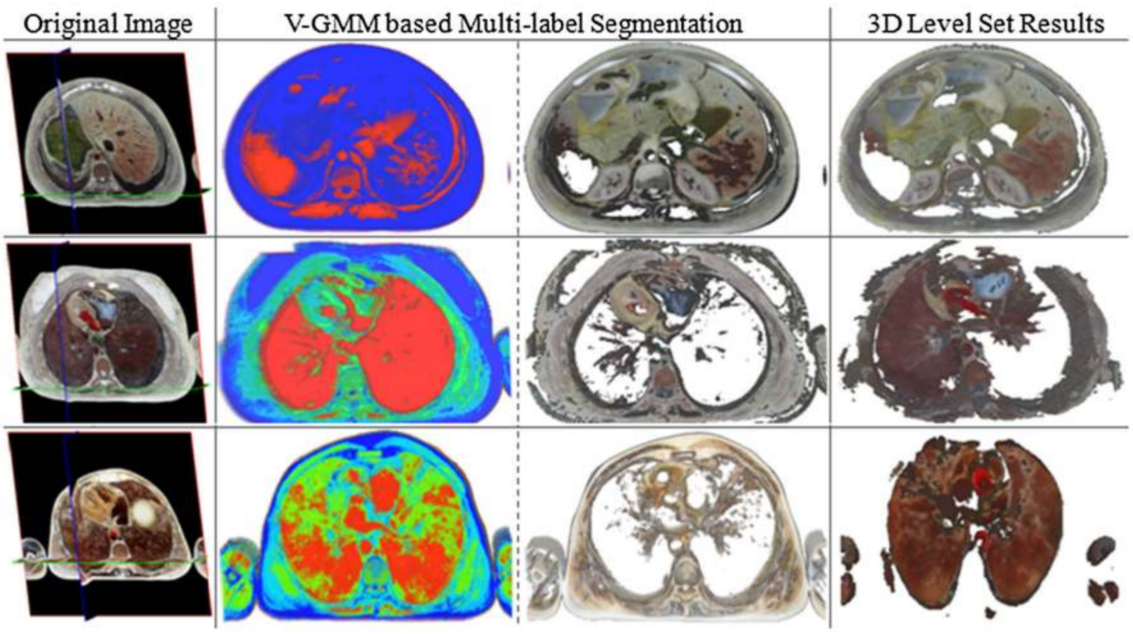


Figure 3. Multi-label segmentation results and their comparison with 3-D Level Set. V-GMM method is more accurate than the 3-D level set method for images with low color contrast, and 3-D Level set method is hard to be employed for multi-label segmentation.

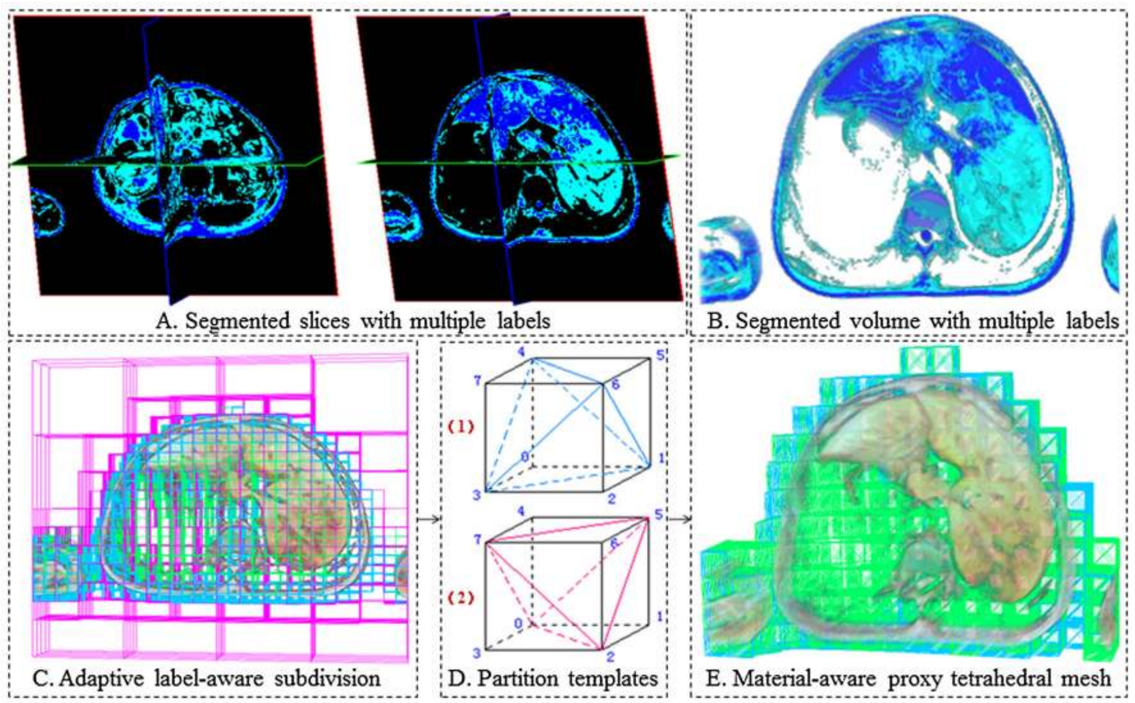

A. Segmented slices with multiple labels
B. Segmented volume with multiple labels
C. Adaptive label-aware subdivision
D. Partition templates
E. Material-aware proxy tetrahedral mesh
(1)
(2)
0
1
2
3
4
5
6
7

Figure 4. Label-aware proxy generation. We first hierarchically divide the working space by an adaptive octree according to the number of tissue types contained in the sub-space. To avoid edge intersection between neighboring tetrahedra, we define two different partitioning types. By taking into account the adjacency relations of the leaf cubes, we can easily determine the partitioning type and decompose each leaf cube into five tetrahedra.

account, we can easily determine the partitioning type of each cuboid. Figure 4(E) demonstrates the generated label-aware tetrahedral mesh over a segmented liver volumetric image, where each tetrahedron will serve as a label-aware finite element for the subsequent physical modeling.

4.3. NFEM-Based Physical Modeling

Given the material parameters of the target tissue such as density ρ, Young's modulus E, and Poisson's ratio ν, we first compute the mass of each tetrahedron according to its volume and non-uniformly distribute each tetrahedron's mass to its four vertices as that in [34]. Meanwhile, we calculate the time step $\triangle t$ that can guarantee the stability of the explicit time integration [16] by using

$$\begin{cases} \triangle t = \frac{L_e}{c} \\ c = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}} \end{cases}, \tag{1}$$

where L_e is the smallest edge length among the tetrahedra, and c is defined as the dilatational wave speed of the material.

Similar to TLED, we measure the deformation at time t with respect to its undeformed configuration at time 0 via deformation gradient tensor ${}_0^t\mathbf{X}$, which describes the stretches and rotations due to non-linear elastic deformation as

$${}_0^t\mathbf{X}_{ij} = \frac{\partial^t x_i}{\partial^0 x_j}. \tag{2}$$

Then, we can respectively obtain the right Cauchy–Green deformation tensor ${}_0^t\mathbf{C} = {}_0^t\mathbf{X}^T{}_0^t\mathbf{X}$ and Green–Lagrange strain tensor ${}_0^t\mathbf{E} = \frac{1}{2}({}_0^t\mathbf{C} - \mathbf{I})$. And the second Piola–Kirchhoff stress ${}_0^t\mathbf{S}$ is employed to measure the stress, which is defined as

$$\frac{{}^0\rho}{{}^t\rho} = {}^tJ = det\left({}_0^t\mathbf{X}\right). \tag{3}$$

where ${}^t\mathbf{T}$ represents the Cauchy stress per unit area in the deformed geometry, ${}_t^0\mathbf{X} = ({}_0^t\mathbf{X})^{-1}$, and the mass density ratio ${}^0\rho/{}^t\rho$ is governed by

$${}_0^t\mathbf{S} = \frac{{}^0\rho}{{}^t\rho}\left({}_t^0\mathbf{X}\right)\left({}^t\mathbf{T}\right)\left({}_t^0\mathbf{X}^T\right), \tag{4}$$

With the help of the proxy, we use the shape functions of iso-parametric tetrahedral finite element to interpolate relevant physical quantities.

$$\begin{cases} h_1 = (1-r)(1-s)(1-t)/8 \\ h_2 = (1+r)(1-s)(1-t)/8 \\ h_3 = (1+s)(1-t)/4 \\ h_4 = (1+t)/2 \end{cases} \tag{5}$$

h_1, h_2, h_3, and h_4 are respectively the shape functions of the four vertices in each tetrahedron. Thus, the stiffness matrix needs not to be dynamically assembled anymore, which can be directly computed at the element level by

$$K(U)U = \sum_e {}^t\mathbf{F}(e), \tag{6}$$

$${}^t\mathbf{F} = \int_{{}^0V}\left({}_0^t\mathbf{B}_L^T\right)\left({}_0^t\hat{\mathbf{S}}\right)d^0V, \tag{7}$$

where ${}_0^t\hat{\mathbf{S}}$ is the vector form of the second Piola–Kirchhoff stress, ${}_0^t\mathbf{B}_L$ is the strain-displacement matrix, whose i-th sub-matrix can be obtained by transforming ${}_0\mathbf{B}_{L0}$ using the deformation gradient ${}_0^t\mathbf{X}$ through ${}_0^t\mathbf{B}_L^{(i)} = {}_0\mathbf{B}_{L0}^{(i)}{}_0^t\mathbf{X}^T$. And ${}_0\mathbf{B}_{L0}^{(i)}$ can be computed by means of the spatial derivatives of the shape functions:

$${}_0\mathbf{B}_{L0}^{(i)} = \begin{bmatrix} \frac{\partial h_i}{\partial^0 x} & 0 & 0 \\ 0 & \frac{\partial h_i}{\partial^0 y} & 0 \\ 0 & 0 & \frac{\partial h_i}{\partial^0 z} \\ \frac{\partial h_i}{\partial^0 y} & \frac{\partial h_i}{\partial^0 x} & 0 \\ 0 & \frac{\partial h_i}{\partial^0 z} & \frac{\partial h_i}{\partial^0 y} \\ \frac{\partial h_i}{\partial^0 z} & 0 & \frac{\partial h_i}{\partial^0 x} \end{bmatrix} (i = 1, \ldots, 4). \tag{8}$$

Suppose that all the nodal forces ${}^t\mathbf{F}$ and the external forces ${}^t\mathbf{R}$ have been obtained, we can further iteratively update the displacement for each vertex in the proxy mesh by

$${}^{t+\triangle t}\mathbf{U}_i = A_i\left({}^t\mathbf{R}_i - {}^t\mathbf{F}_i\right) + B_i{}^t\mathbf{U}_i + C_i{}^{t-\triangle t}\mathbf{U}_i. \tag{9}$$

```
Algorithm 1: Physical proxy generation.
input  : segmented image $f_s$, $\rho$, $E$, $\nu$, and the
         maximal octree depth $n_d$.
output: void.
1: Create adaptive octree based on $f_s$,$n_d$;
2: Generate tetrahedral proxy mesh;
3: Compute $L_e$, $\triangle t$, and perform mass distribution;
4: Allocate memory for $A_i$, $B_i$, $C_i$, ${}^t\mathbf{U}_i$, ${}^{t+\triangle t}\mathbf{U}_i$, and
   the shape function derivative lists on GPU;
5: Invoke a CUDA kernel to precompute $A_i$, $B_i$, $C_i$;
6: Invoke a CUDA kernel to precompute the derivatives
   of shape functions;
```

And A_i, B_i, and C_i can be pre-computed in a vertex-wise fashion as

$$\begin{cases} A_i = \frac{1}{\mathbf{D}_{ii}/(2\triangle t)+\mathbf{M}_{ii}/\triangle t^2} \\ B_i = \frac{2\mathbf{M}_{ii}/\triangle t^2}{\mathbf{D}_{ii}/(2\triangle t)+\mathbf{M}_{ii}/\triangle t^2} = \frac{2\mathbf{M}_{ii}}{\triangle t^2}A_i \\ C_i = \frac{\mathbf{D}_{ii}/2\triangle t-\mathbf{M}_{ii}/\triangle t^2}{\mathbf{D}_{ii}/(2\triangle t)+\mathbf{M}_{ii}/\triangle t^2} = \frac{\mathbf{D}_{ii}}{\triangle t}A_i - \frac{B_i}{2} \end{cases} \tag{10}$$

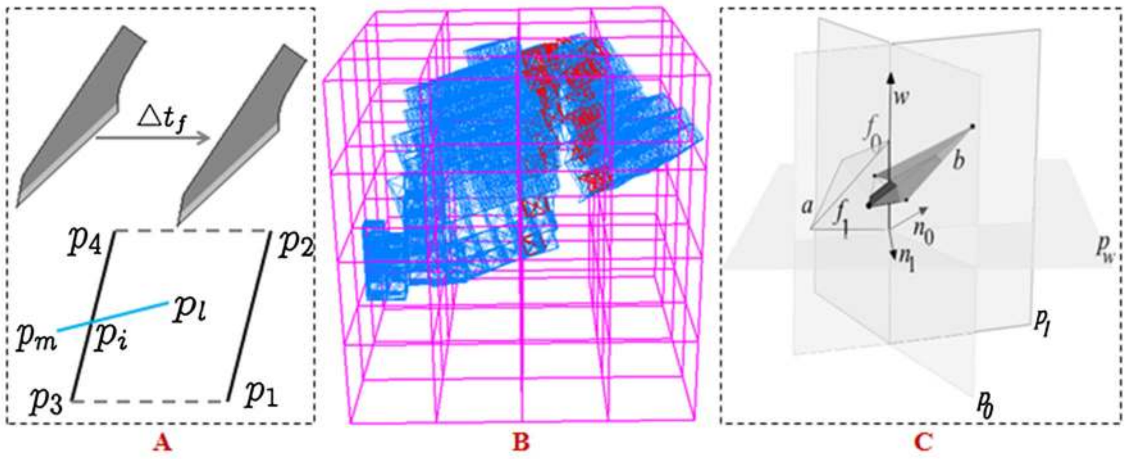


Figure 5. Cutting and collision handling. Because the inter-frame time interval is usually very short, the surface swept by the knife can be approximated as a plane (A). By determining the cutting type of the element, we refine the proxy mesh on CUDA (B). CUDA-based fast tetrahedron overlap testing algorithm (C) is employed to perform collision detection.

To sum up, the pseudo-code of physical proxy generation is documented in Algorithm 1.

5. CUDA-BASED SIMULATION

5.1. Deformation

Because the external force ${}^{t}\mathbf{R}_i$ can be directly obtained, at each time step, we only need to re-compute ${}^{t}\mathbf{F}_i$ according to Equation (7). In fact, ${}^{t}\mathbf{F}_i$ transitively depends on ${}^{t}_{0}\hat{\mathbf{S}}$ and ${}^{t}_{0}\mathbf{X}$. For ${}^{t}_{0}\hat{\mathbf{S}}$, we compute it using a hyperelastic neo-Hookean model given by

$$ {}^{t}_{0}\hat{\mathbf{S}} = \mu\left(\delta_{ij} - {}^{t}_{0}\mathbf{C}^{-1}_{ij}\right) + \lambda\left({}^{t}J\right)\left({}^{t}J - 1\right)\left({}^{t}_{0}\mathbf{C}^{-1}_{ij}\right), \quad (11) $$

where μ and λ are the Lamé constants and δ_{ij} is the Kronecker's delta. As for ${}^{t}_{0}\mathbf{X}$, it can be computed by way of shape-function-based interpolation:

$$ {}^{t}_{0}\mathbf{X} = {}^{t}\mathbf{u}_e^{T}\,\partial\mathbf{H} + \mathbf{I}, \quad (12) $$

where ${}^{t}\mathbf{u}_e$ is a 4×3 matrix consisting of the vertex displacements of each tetrahedron, and $\partial\mathbf{H}$ is a 4×3 matrix consisting of the pre-computed shape function.

Then, by summing all the corresponding forces in the tetrahedra that share vertex i, we can finally get the internal force ${}^{t}\mathbf{F}_i$ for the vertex i in the proxy mesh. And the vertex-wise displacement updating is well suitable for CUDA-based parallel computation, which is detailed in Algorithm 2.

Algorithm 2: CUDA-based simulation.

```
input  : user interaction.
output: void.

for each cycle of the simulation loop do
    if cutting mode is switched on then
        1:Invoke a CUDA kernel for collision detection;
        2:Invoke a CUDA kernel to update ${}^{0}V, m, A_i, B_i, C_i$;
    for i=1:iteration number do
        1:Compute ${}^{t}\mathbf{R}_i$;
        2:Invoke a CUDA kernel for ${}^{t}\mathbf{F}_i$;
        3:Invoke a CUDA kernel for ${}^{t}\mathbf{U}_i$;
    end
    1:Invoke a CUDA kernel to construct grid cell indices;
    2:Invoke a CUDA kernel to collect the nearby element indices and assign them to grid cells;
    3:Invoke a CUDA kernel to conduct collision detection and label the affected vertices to be constrained;
    4:Invoke a CUDA kernel to add constraints to ${}^{t}\mathbf{F}_i$.
end
```

5.2. Cutting and Collision Handling

As shown in Figure 5(A), during the cutting procedure, because the inter-frame time interval $\triangle t_f$ is usually very short, the moving direction of the operating tool (e.g., knife) can be considered to be constant, and the surface swept by the knife can be approximated as a plane with a parallelogram $\diamond_{p_1 p_2 p_3 p_4}$, where the edges $p_1 p_2$ and $p_3 p_4$ respectively represent the current and the last positions of the knife.

Given a tetrahedral element, let $p_m p_l$ represent one of its six edges and suppose it intersects with $\diamond_{p_1 p_2 p_3 p_4}$ at p_i, we can derive their geometrical relations as follows:

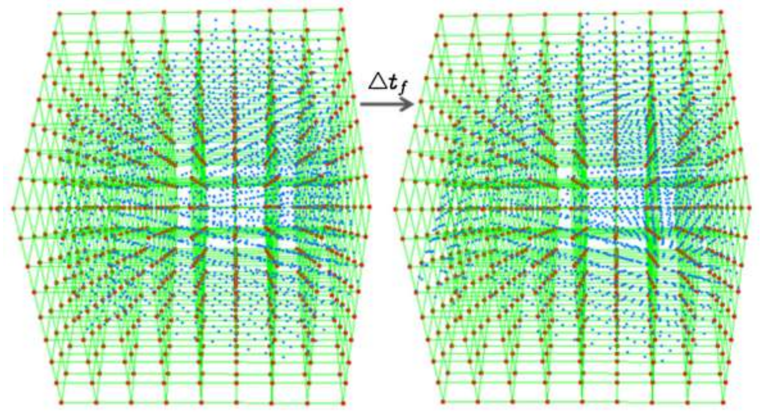


Figure 6. Illustration for the dense scalar field upsampling. Blue points represent the sparse displacement field resulted from the deformed proxy vertexes, while red points represent the sampling positions of the evolving dense displacement field, which can be obtained via fitting-based upsampling.

$$\begin{cases} p_i = p_1 + r(p_3 - p_1) + s(p_2 - p_1) \\ p_i = p_m + t(p_l - p_m) \end{cases}, \tag{13}$$

where r, s, t are the parameter coordinates of p_i. We can further obtain

$$r(p_3 - p_1) + s(p_2 - p_1) + t(p_m - p_l) = p_m - p_1. \tag{14}$$

By noting $p_a = p_3 - p_1, p_b = p_2 - p_1, p_c = p_m - p_l, p_d = p_m - p_1$ and rewriting Equation (14) in matrix form, we can compute the parameter coordinate r, s, t of the intersection point p_i as

$$\begin{bmatrix} r \\ s \\ t \end{bmatrix} = \begin{bmatrix} p_{a_1} & p_{b_1} & p_{c_1} \\ p_{a_2} & p_{b_2} & p_{c_2} \\ p_{a_3} & p_{b_3} & p_{c_3} \end{bmatrix}^{-1} \begin{bmatrix} p_{d_1} \\ p_{d_2} \\ p_{d_3} \end{bmatrix}, \tag{15}$$

where, if and only if $r \in (0,1), s \in (0,1), t \in (0,1)$ are met simultaneously, the edge $p_m p_l$ is confirmed to have been cut by the knife. Thus, we can easily determine the number of cut edges in each tetrahedral element by taking all the six edges into account. We implement the topological changing method for tetrahedra proposed by Forest *et al*. [35] on CUDA to refine the cut tetrahedral mesh, which has 10 distinct refining configurations for each tetrahedron corresponding to different cut cases. Furthermore, according to the refined mesh, we need to update the volume ${}^0V$, the mass distribution, A_i, B_i, C_i, and the shape function derivatives for all the affected tetrahedra. Meanwhile, the integral time step $\triangle t$ should also be updated according to the newly computed smallest edge length L_e. Because the cutting process can be tetrahedron-wisely handled, its CUDA-based parallel computation is integrated with Algorithm 2. For the sake of CUDA implementation, it should be noted that the removed original tetrahedron will only be labeled as *invalid* and all of its related data will still retain their positions in the corresponding lists.

As for collision detection, we propose a regular grid assisted method to compute it on CUDA in parallel. As shown in Figure 5(B), we partition the 3D working space into a coarse regular grid in the initial stage. In each cycle of simulation loop, to rapidly exclude a large majority of tetrahedron-pairs which are impossible to collide with each other in advance, we first assign each deformed tetrahedron to some grid cells according to the current positions of its four vertices (Figure 5(B)). Meanwhile, we make a CUDA-based implementation for the fast tetrahedron overlap testing algorithm proposed by Ganovelli et al. [37]. As shown in Figure 5(C), its central idea is: let e be an edge shared by faces f_0, f_1, p_0, p_1 be half-planes extending those faces, and W be the resulting wedge containing the tetrahedron, then if the other tetrahedra intersect with W there can not be a separating plane containing e, for details please refer to [35]. Next, taking the tetrahedra allocated in the same grid cells as a group, we can parallelly conduct collision detection in an element-wise fashion. The collided tetrahedra are shown in red in Figure 5(B).

If a tetrahedron is detected to collide with others, we add a penalty force to its four vertices along the opposite direction of their own motion as

$${}^t\mathbf{U}_i = {}^t\mathbf{U}_i + \left({}^{t-\Delta t}\mathbf{U}_i - {}^t\mathbf{U}_i\right)/2. \tag{16}$$

And the pseudocode for CUDA-based cutting and collision handling is documented in Algorithm 2.

5.3. Dynamic Volume Rendering

So far, in each simulation cycle, we can get the vertex displacement of the proxy and thus their deformed positions (the blue points illustrated in Figure 6). However, the displacement field is too sparse to accurately represent the deformed volumetric images. Therefore, to obtain the evolving dense scalar field represented by the red points in Figure 6, we need to make use of upsampling. Obviously,

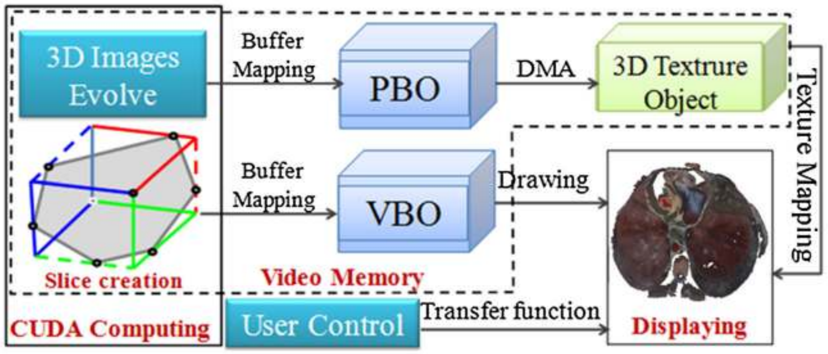


Figure 7. CUDA data flow of rendering pipeline. Pixel buffer object (PBO) serves as a bridge for image data transfer from CUDA buffer to texture cache while involving no CPU cycle, and the generated object-order primitives by decomposing the volume into slice stacks can be directly mapped to vertex buffer object (VBO) from CUDA buffer.

it is not accurate to directly upsample the sparse intensity field dictated by the proxy vertices. We shall adopt a scheme based on moving least squares (MLS) fitting to upsample the displacement field at each voxel position p in each cycle, and then use the upsampled displacement to fetch the corresponding voxel color from original volumetric images.

Similar to collision handling, the regular grid can also be used to assist the CUDA-based fitting computation, which can greatly improve the search efficiency of neighboring proxy vertices. We first compute the cell coordinate according to p, and then collect the vertex set S_v from the $3 \times 3 \times 3$ neighboring cells according to the element indices assigned to the cells. Thus, the adjacent dense field locations will partially use the overlapped proxy vertices when conducting MLS fitting. As mentioned in Section 5.2, when cutting occurs, some original tetrahedra will be labeled as *invalid* because they may have been replaced by some smaller, more-refined tetrahedra, and they should be removed. Therefore, the vertex set S_v must exclude those vertices belonging to such tetrahedra.

Given the second-order monomial basis $\mathbf{b}$, we locally fit a polynomial function around each cell p by minimizing the following energy function:

$$\sum_{v \in N(p)} \left(\mathbf{b}^T \cdot \mathbf{c} - {}^t\mathbf{U}(v)\right)^2 \omega_p(v). \qquad (17)$$

$N(p)$ represents the vertex set in the supporting domain of the spatial Gaussian kernel function ω_p, whose supporting domain is set to be $3L_e$. Then, the evolved voxel color can be further indexed from the original volumetric images on the basis of its deformed position through

$$\begin{cases} I(p) = f(p - \mathbf{f}_d(p)), (p - \mathbf{f}_d(p)) \in \Omega \\ I(p) = 0, \text{otherwise} \end{cases} \qquad (18)$$

The aforementioned process can also be efficiently implemented on CUDA; we document the algorithm together with CUDA-based time-varying volume visualization in Algorithm 3.

Algorithm 3: CUDA-based rendering.

```
input  : slice number $n_s$, 3-D texture cache $tex$,
         transfer function $T_f$.
output: visualization.

for each cycle of the simulation loop do
    if the first cycle then
        1:Initiate a $PBO$;
        2:Initiate a $VBO$ according to $n_s$;
        3:Allocate memory for primitive vertex array
          $A_r$;
    1:Update $T_f$ with user control;
    2:Invoke a CUDA kernel to upsample ${}^t\mathbf{U}$;
    3:Map $PBO$ to deformed image $I$;
    4:Invoke a CUDA kernel to compute $I(p)$;
    5:Transfer data from $PBO$ to $tex$;
    6:Map $VBO$ to $A_r$;
    7:Invoke a CUDA kernel to generate $A_r$;
    8:Invoke the rendering function;
end
```

One of our goals is to synchronously visualize the anatomical structure both on the cutting surface and inside the organs/tissues during cutting simulation. Although all the runtime computation is executed on CUDA, in each cycle, it conventionally needs to transfer the evolved 3-D images to host memory first and then copy them to graphics processing unit (GPU) texture cache through graphics application programming interface (API), where the greatest bottleneck is the bandwidth and texture cache coherency. As shown in Figure 7, our strategy is to employ a pixel buffer object (PBO) as a bridge to transfer image data from CUDA buffer to texture cache, which can perform fast pixel data transfer through direct memory access without involving any CPU cycle.

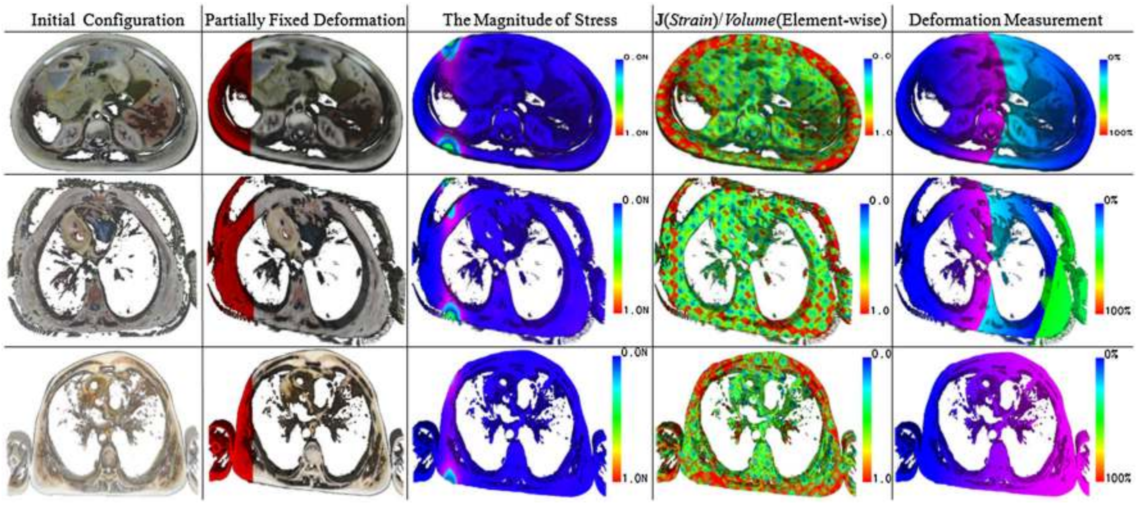


Figure 8. Deformation of partially fixed tissues/organs under the influence of gravity. The fixed regions are labeled by red; the second column shows the deformation results; and the third to the fifth columns quantitatively measure the space-varying stress and strain distribution as well as the global deformation ratio in a color-encoded way.

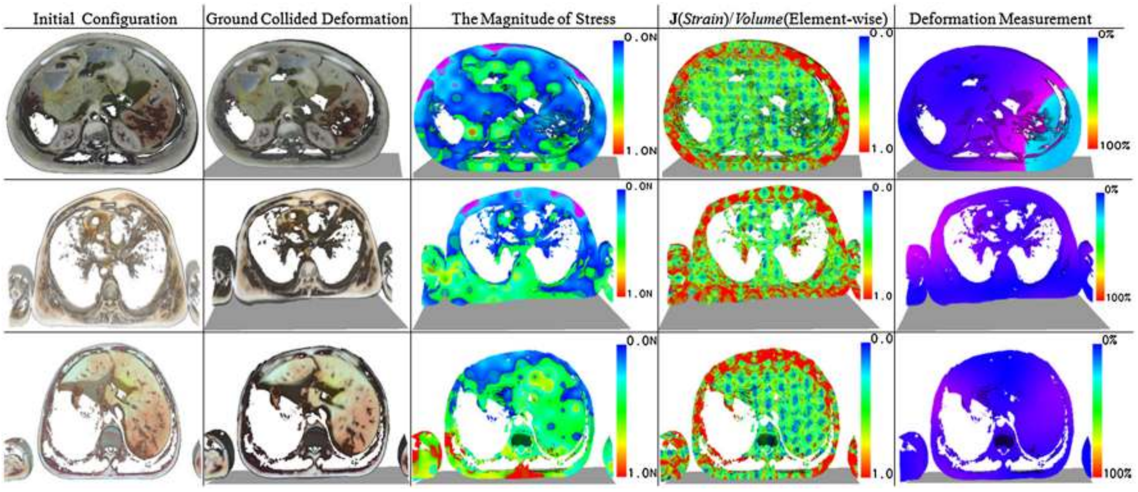


Figure 9. Deformation under the influence of gravity and the collision between tissues and constrained plane, wherein the vertical displacement is not taken into account when computing deformation ratio.

Besides, because image-order approaches, such as GPU-based ray casting, usually need much iteration to fetch 3-D texture and blend the color for a single pixel, it can not be afforded here because of the already-existing expensive NFEM simulation. Thus, 3-D texture alone is not enough; we should create object-order primitives by decomposing the volume into slice stacks of textured polygons, and we always create a six-vertex polygon for each slice in order to improve the uniformity of CUDA-based data structure. If the intersecting polygon consists of less than six vertices, one or more duplicate vertices (located on the dotted line in Figure 7) will be generated to occupy the corresponding location. Then, the generated polygon data is directly mapped to a vertex buffer object (VBO) from CUDA buffer, which can drastically increase the rendering efficiency. Finally, with the transfer function controlled by the user, we can achieve the interactive volume rendering performance while undergoing physics-based image manipulation. And the CUDA-based rendering implementation is detailed in Algorithm 3.

6. EXPERIMENTAL RESULTS

We have developed a prototype system using C++ and CUDA. All the experiments run on a laptop with NVIDIA GeForce 330 M GPU, Intel Core (TM) i7 CPU (1.6 GHz, two cores), and 4G RAM, and all the volumetric datasets are from Chinese Visible Human Project (CVHP).

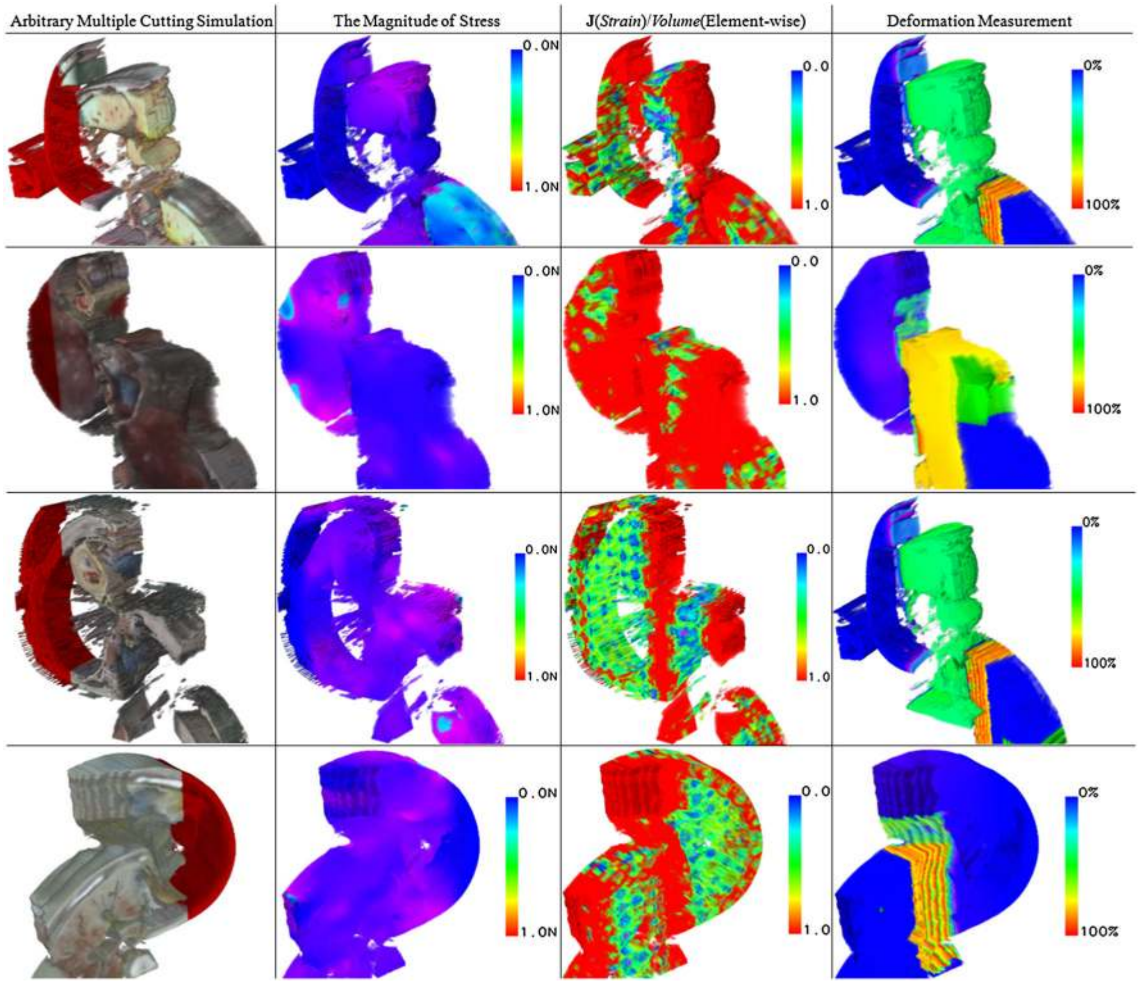

Arbitrary Multiple Cutting Simulation
The Magnitude of Stress
J(*Strain*)/*Volume*(Element-wise)
Deformation Measurement
0.0N
1.0N
0.0
1.0
0%
100%

Figure 10. Cutting simulation and accompanying deformation of partially fixed tissues/organs under the influence of gravity. Our method supports arbitrary cutting surface and can also accommodate multiple cuts, while the anatomical structures on the dynamic cutting surface can be synchronously visualized in a physiologically plausible way.

In all the following experiments, we scale each organ into a cube of 8dm^3. Because our finite elements are label-aware, it allows us to set tissue-specific physical properties according to different element types. Meanwhile, we also note that the adopted tissue-specific properties are experientially set while not physically accurate, because currently, it is extremely hard for us to get the true values of such physical properties regardless of how we would choose to make use of *in vivo* or *in vitro* methods. In all our experiments, the tissue density parameters are set to be within $[800\text{kg/m}^3, 1200\text{kg/m}^3]$, the parameters μ are set to be within $[100, 500]$, while the parameters λ are set to be within $[4000, 5500]$. In fact, strain in nature is a tensor, and thus, we employ $J(\text{strain})/\text{volume}$ to measure the element-wise strain afforded in different local regions. According to Equation (3), we define the measurement $M_e = J(\text{strain})/\text{volume}$ as

$$M_e = \frac{{}^tJ}{V_e} = \frac{det({}^t_0\mathbf{X})}{V_e}, \tag{19}$$

where V_e represents the volume of an element e. To quantitatively illustrate the space-varying deformation degree of the elements and clearly show their relative differences, the measurement results are normalized to [0, 1] and color-encoded in the fourth column. Meanwhile, the accompanying stress distribution is shown in the third column. It should be noted that the element-wise stress is in nature a tensor. According to Equation 7, it is calculated through volume integral and can be discretized as six force vectors. Thus, by summing all the corresponding forces in the tetrahedra that share the same vertex, we can finally get the composition of the discretized stress forces for each vertex in the proxy mesh. In our experiments, we vertex-wisely measure the composition of the discretized stress with their magnitude. Besides, to quantitatively measure the global scale deformation with respect to its undeformed configuration, we list the deformation ratio (the ratio of displacement to its maximal dimension) for each experiment result in the fifth columns of Figures 8, 9, and 10.

Table I. Time performance (in millisecond).

VIS	NTE	PG	PS	TU	CH	DVR
$512^2 \times 82$	6775	578	14	14	1	48
$512^2 \times 80$	7515	577	15	16	2	49
$512^2 \times 86$	11940	890	17	17	2	74
$512^2 \times 94$	10095	795	17	17	2	71
$512^2 \times 82$	8630	655	15	17	2	53

VIS, volumetric image size; NTE, number of tetrahedron element; PG, proxy generation; PS, physical simulation; TU, topological updating; CH, collision handling; DVR, dynamic volume rendering.

The second column of Figure 8 shows the deformation results of partially fixed organs (marked in red) under the influence of gravity. From the color-encoded measurements, we can observe that our method can easily simulate large deformation (the deformation percentage is above 10%) while keeping physiologically plausible anatomical structures during deformation. It can be seen that tissues nearby the fixed parts usually suffer from bigger stress, which coincides well with the fact.

Figure 9 shows the deformation results under the influence of gravity and subject to the constraints on the ground. Because of the collision response when touching the constrained plane, the stress should spread quickly as long as collision occurs. The stress measurement correctly presents such effects (see the third column of Figure 9). And it is expected to undergo large-scale deformation when the soft tissues fall onto the constrained ground; the space-varying deformation ratio measurements correctly show such facts. Here, it should be noted that the computation of deformation ratio in Figure 9 does not take the vertical displacement into account, because the vertical displacement contains the free-fall transformation from their original position to the plane.

Figure 10 demonstrates the arbitrary multiple cutting simulation over partially fixed tissues. During simulation, the cut parts will have a free fall under the influence of gravity. The experiments show that our method can accurately demonstrate the patient-specific anatomical structures on the deformed cut surfaces, which are currently hard to achieve in most of the existing surgery simulators. Meanwhile, it shows that our method is stable and robust to sustain arbitrary and multiple cuts. In addition, during the cutting process, self-collision may occur because the cut parts are not rigid bodies and will undergo deformation. The integration of collision handling can effectively and accurately simulate these cases, which can be seen from the color-coded quantitative element-wise stress and strain measurements. For more vivid results, please refer to our supplementary video.

Furthermore, to fully analyze the time performance of our method, Table I lists the volumetric image size and the corresponding number of tetrahedron element, and documents the time testing for most of the dominating steps, which include proxy generation, physical simulation, topological updating, collision handling, and dynamic volume rendering, wherein proxy generation is only executed in the pre-processing stage. It shows that dynamic volume rendering is a little time-consuming. However, even on a commodity laptop, we can still achieve interactive performance. Therefore, with the help of a more powerful computer or a multi-GPU scheme, our framework has great potentials to afford even larger data size and achieve higher fidelity for physical simulation, as well as real-time performance efficiency.

7. CONCLUSION AND DISCUSSION

We have detailed a comprehensive and fully CUDA-accelerated volumetric image manipulation framework to address a suite of research challenges in image-based interactive surgical simulation. Our framework supports instant utilization of patient-specific clinical images, physics-based large-scale non-linear deformation, arbitrarily cutting simulation, synchronized collision handling, and realistic visualization of cut surface and volume, all of which collectively have a broader appeal to both image-based animation and physics-based simulation in the virtual world. Extensive experiments on various medical images together with their quantitative physical behavior measurements have demonstrated its superior performance.

The paper's current foci are on the seamless image manipulation framework for physically plausible soft tissue deformation and cutting simulation. Although our current framework can offer to set different material properties for each finite element, there still exist some open problems yet to be improved. First, it is difficult to accurately capture material parameters of heterogeneous tissues regardless of making use of *in vivo* or *in vitro* methods. Second, given the volumetric images, it remains hard to design a universally-applicable method to accurately segment different kinds of tissues (especially for the tiny or thin-shell ones) automatically. Third, given the accurate material parameters and heterogeneous segmentation results, it is challenging to achieve accurate registration.

Our technical vision is to achieve label-aware and even physiological property-coupled surgical simulation directly over clinical images in the near future. Our immediate research efforts are geared toward incorporating blooding and suturing simulation into our current framework in order to enhance its realism during surgical training and then propelling its utility in the clinical setting.

ACKNOWLEDGEMENTS

This research is supported in part by National Natural Science Foundation of China (No. 61190120, 61190121, and 61190125) and National Science Foundation of USA (No. IIS-0949467, IIS-1047715, and IIS-1049448). Thanks to the Chinese Visible Human Project for providing the volumetric images. We would like to thank the anonymous reviewers for their very constructive comments and suggestions that helped greatly improve this paper's quality.

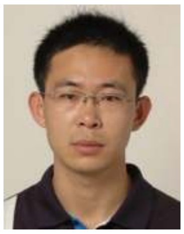
REFERENCES

1. Engel K, Hadwiger M, Kniss JM, Rezk-Salama C, Weiskopf D. *Real Time Volume Graphics*. A K Peters, Ltd., 888 Worcester Street Suite 230, Wellesley, MA, United States, 2006.
2. Miller K, Taylor Z, Nowinskin WL. Towards computing brain deformations for diagnosis, prognosis and neurosurgical simulation. *Journal of Mechanics in Medicine and Biology* 2005; **5**(1): 105–121.
3. Nakao M, Hung KWC, Yano S, Yoshimura K, Minato K. Adaptive proxy geometry for direct volume manipulation, In *Proceedings of IEEE pacific visualization symposium*, 2010; 161–168.
4. Mosegaard J, Herborg P, Sørensen TS. A GPU accelerated spring mass system for surgical simulation. *Studies in Health Technology and Informatics* 2005; **111**: 342–348.
5. Dick C, Georgii J, Westermann R. A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Transactions on Visualization and Computer Graphics* 2011; **17**(11): 1663–1675.
6. Soler L, Marescaux J. Patient-specific surgical simulation. *World Journal of Surgery* 2008; **32**(2): 208–212.
7. Hung KWC, Nakao M, Yoshimura K, Minato K. Background-incorporated volumetric model for patient-specific surgical simulation: a segmentation-free, modeling-free framework. *International Journal of Computer Assisted Radiology and Surgery* 2011; **6**(1): 35–45.
8. Bruckner S, Gröller ME. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**(5): 1077–1084.
9. Wittek A, Miller K, Kikinis R, Warfield Sk. Patient-specific model of brain deformation: application to medical image registration. *Journal of Biomechanics* 2007; **40**(4): 919–929.
10. Fong P. Sensing, acquisition, and interactive playback of data-based models for elastic deformable objects. *International Journal of Robotics Research* 2009; **28**(5): 630–655.
11. Bao C, Wang BL. A open source based general framework for virtual surgery simulation, In *Proceedings of international conference on biomedical engineering and informatics*. IEEE Computer Society, Washington, DC, USA, 2008; 575–579.
12. Sutherland L, Middleton P, Anthony A, Hamdorf J, Cregan P, Scott D, Maddern G. Surgical simulation: a systematic review. *Annals of Surgery* 2006; **243**(3): 291–300.
13. Plass A, Scheffel H, Alkadhi H, Kaufmann P, Genoni M, Volkmar F, Grüenfelder J. Aortic valve replacement through a minimally invasive approach: preoperative planning, surgical technique, and outcome. *Annals of Thoracic Surgery* 2009; **88**(6): 1851–1856.
14. Satava RM. Historical review of surgical simulation: a personal perspective. *World Journal of Surgery* 2008; **32**(2): 141–148.
15. Burger K, Kruger J, Westermann R. Direct volume editing. *IEEE Transactions on Visualization and Computer Graphics* 2008; **14**(6): 1388–1395.
16. Taylor ZA, Cheng M, Ourselin S. High-speed nonlinear finite element analysis for surgical simulation using graphics processing units. *IEEE Transactions on Medical Imaging* 2008; **27**(5): 650–663.
17. Mcguffin MJ, Tancau L, Balakrishnan R. Using Deformations for browsing volumetric data, In *Proceedings of 14th IEEE Visualization*. IEEE Computer Society, Washington, DC, USA, 2003; 401–408.
18. Correa CD, Silver D, Chen M. Discontinuous displacement mapping for volume graphics, In *Proceedings of the Fifth Eurographics / IEEE VGTC Workshop on Volume Graphics,* Eurographics Association, Boston, Massachusetts, USA, 2006; 9–16.
19. Correa CD, Silver D, Chen M. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**(5): 1069–1076.
20. Masutani Y, Inoue Y, Kimura F, Sakuma I. Development of surgical simulator based on FEM and deformable volume rendering. In *Medical Imaging 2004: Visualization, Image-Guided Procedures, and Display*, Fitzpatrick J.-M., Sonka M (eds). Society of Photo Optical, 1000 20th Street, Bellingham, WA, United States, 2004; 500–507.
21. Takayama K, Okabe M, Ijiri T, Igarashi T. Lapped solid textures: filling a model with anisotropic textures. *ACM Transactions on Graphics* 2008; **27**(3): 1–9.
22. Correa CD, Silver D, Chen M. Volume deformation via scattered data interpolation, In *Proceedings of IEEE VGTC Workshop on Volume Graphics,* Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007; 91–98.
23. Chang J, Zhang JJ. Mesh-free deformations. *Computer Animation and Virtual Worlds* 2004; **15**(3-4): 211–218.
24. Pauly M, Keiser R, Adams B, Dutré P. Meshless animation of fracturing solids. *ACM Transactions on Graphics* 2005; **24**(3): 957–964.
25. Jin X, Joldes GR, Miller K, Yang KH, Wittek A. Meshless algorithm for soft tissue cutting in surgical simulation. *Computer Methods in Biomechanics and Biomedical Engineering* 2012; **PMID**: 22974246.
26. Miller KS, Joldes GR, Lance D, Wittek A. Total lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Communi-*

cations in Numerical Methods in Engineering 2007; **23**(2): 121–134.

27. Noe KO, Sorensen TS. Solid mesh registration for radiotherapy treatment planning. *Lecture Notes in Computer Science* 2010; **5958**: 59–70.
28. Taylor ZA, Comas O, Cheng M, Passenger J, Hawkes DJ, Atkinson D, Ourselin S. On modelling of anisotropic viscoelasticity for soft tissue simulation: numerical solution and GPU execution. *Medical Image Analysis* 2009; **13**(2): 234–244.
29. Fortmeier D, Mastmeyer A, Handels H. GPU-based visualization of deformable volumetric soft-tissue for real-time simulation of haptic needle insertion. In *Bildverarbeitung Für Die Medizin 2012, Informatik Aktuell*, Tolxdorff T, Deserno T. M., Handels H, Meinzer H.-P. (eds). Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2012; 117–122.
30. Fierz B, Spillmann J, Harders M. Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects, In *Proceedings of ACM SIGGRAPH*. ACM, New York, NY, USA, 2011; 257–266.
31. Wicke M, Botsch M, Gross M. A finite element method on convex polyhedra. *Computer Graphics Forum* 2007; **26**(3): 355–364.
32. Chao I, Pinkall U, Sanan P, Schröder P. A simple geometric model for elastic deformations. *ACM Transactions on Graphics* 2010; **29**(4): 8:1–38:6.
33. Fierz B, Spillmann J, Aguinaga I, Harders M. Maintaining large time steps in explicit finite element simulations using shape matching. *IEEE Transactions on Visualization and Computer Graphics* 2012; **18**(5): 717–728.
34. Jerabkova L, Bousquet G, Barbier S, Faure F, Allard J. Volumetric modeling and interactive cutting of deformable bodies. *Progress of Biophysics and Molecular Biology* 2010; **103**(2-3): 217–224.
35. Forest C, Delingette H, Ayache N. Proceedings of medical image computing and computer-assisted intervention. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention*. Springer Berlin Heidelberg, London, UK, 2002; 235–244.
36. Sifakis E, Der KG, Fedkiw R. Arbitrary cutting of deformable tetrahedralized objects, In *Proceedings of eurographics symposium on computer animation*, 2007; 73–80.
37. Ganovelli F, Ponchio F, Rocchini C. Fast tetrahedron-tetrahedron overlap algorithm. *ACM Journal of Graphics Tools* 2002; **7**(2): 17–26.

AUTHORS' BIOGRAPHIES



Shuai Li is an assistant professor at the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. He received his BS in Computer Science from Ocean University of China, MS and PhD in Computer Science at Beihang University. His research interests include computer graphics, real-time and realistic rendering, physics-based modeling and simulation, medical image processing, computer vision, and virtual surgery.

Qinping Zhao is a professor in Computer Science School and the Director of the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. He is now the president of Chinese Association for System Simulation and the vice-chairman of China Association for Science and Technology. He got his BS and MS in Electronics Engineering at Taiyuan University of Technology and PhD in Computer Science at Nanjing University. His research interests are on virtual reality, computer simulation, artificial intellegence, computer graphics, image processing, and computer vision.

Shengfa Wang is a post-doctoral of the School of Computer Science and Technology at Dalian University of Technology and works in the School of Software Technology at Dalian University of Technology. He got his BS and PhD in Computational Mathematics at Dalian University of Technology. His research interests include computer graphics, diffusion geometry, and differential geometry processing and analysis.

Aimin Hao is a professor in the Computer Science School and the Associate Director of the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. He got his BS, MS, and PhD in Computer Science at Beihang University. His research interests are on virtual reality, computer simulation, computer graphics, geometric modeling, image processing, and computer vision.

Hong Qin is a full professor of Computer Science in the Department of Computer Science at Stony Brook University (SUNY). He received his BS and his MS in Computer Science from Peking University, China. He received his PhD in Computer Science from the University of Toronto. Currently, he serves as an associate editor for The Visual Computer, Graphical Models, and Journal of Computer Science and Technology. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer aided geometric design, human-computer interaction, visualization, and scientific computing.