Editor: Mike Potel

Efficient Surgical Cutting with Position-Based Dynamics

Iago Berndt
Federal University of Rio Grande do Sul

Rafael Torchelsen
Federal University of Pelotas

Anderson Maciel
Federal University of Rio Grande do Sul

Simulations of cuts on deformable bodies have been an active research subject for more than two decades. Two surveys published 10 years apart[1,2] argued that most of the problems with the three major tasks involved in virtual cutting—collision handling, topology update, and deformation that complies with frequent topology update—have been tackled. The effective solutions developed to date, however, are only applicable in simplified scenarios.

Although the increased complexity of graphical and biomechanical models have improved the visual quality of virtual surgery, most previous works in cutting limit their scope to actions on one organ or even a particular region in one organ. Thus, the current cutting methods cannot scale to the same level of complexity earned by their rendering counterparts. Often, one task (such as collision, rendering, haptics, deformation, or cutting) must be severely simplified to free up flops (floating-point operations per second) for another. Moreover, in the current state of the art, cutting and deforming require updating most of the precomputed data that is used to speed up the other tasks. This scenario indicates that we cannot simply rely on the increasing computational power to enable the next generation of surgery simulators.

Roughly 10 years ago, position-based dynamics (PBD) appeared as a novel paradigm for simulating dynamic systems.[3] Unlike the previously widespread computer graphics approaches based on forces and some successful impulse/velocity-based approaches, PBD directly manipulates positions. This presents many advantages, such as avoiding overshooting problems in explicit integration schemes and making it easier to handle collision constraints.

Since then, PBD has been used to simulate cloth, deformable solids, rigid bodies, and fluids.[4,5] Recently, researchers proposed a unified approach that models gases, liquids, solids, and clothing together.[6] This is done by defining specific positional constraints among groups of particles that are, for all other purposes (such as collision and rendering), equally represented in the data structure. A parallel PBD solver available in the Nvidia FleX developer library has proven efficient enough for some real-time applications. Although the library's main purpose is to support game development, previous works have successfully shown that game-oriented libraries are also suitable for medical simulators.[7]

PBD's ability to unify solid, liquid, gas, and membrane (cloth) is crucial in the context of simulated surgery. Moreover, it is massively parallel, so it can simulate several organs instead of only one, as is common with previous works. It also provides realistic rendering.

In this article, we address common problems with traditional surgery simulation scenarios and propose PBD-based solutions. Specifically, we introduce a skinning scheme that supports interactive tissue cutting with haptic feedback. In addition, we introduce a method to configure cluster constraints with PBD and an approach to simulate electrosurgery. We demonstrate that

 Published by the IEEE Computer Society

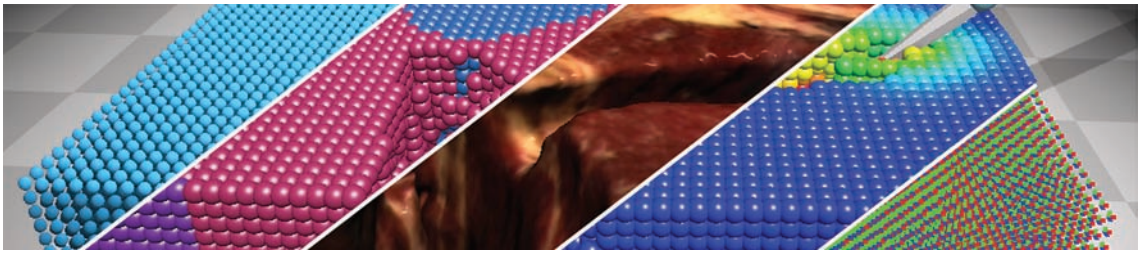


Figure 1. Position-based dynamics (PBD) simulation: (a) object represented by particles, (b) color coding showing the different materials, (c) final rendering, and (d) heat propagation that cuts the body. (e) The orientation of each particle is used to extract the triangular mesh used in the final rendering.

combining these new approaches makes simulated surgery much more plausible.

Current Methods

Many satisfying solutions have been proposed to solve the problem of cutting static/rigid polygonal meshes in computational geometry. However, cutting simulation for deformable bodies involves a higher order of complexity. When dealing with soft bodies, any topological modification caused by cutting will severely limit any assumption used by deformation and collision algorithms to reduce computational cost. This happens because cutting proceeds arbitrarily during real-time interaction, resulting in many possible topological changes.[8]

Computations of soft tissue mechanical responses for surgery simulation and image-guided surgery have been dominated by the finite element method (FEM), which uses a mesh of interconnected elements as a computational grid. However, mesh-based discretization when modeling surgical cutting presents challenges, including high computational cost and the need for remeshing in the vicinity of cutting-induced discontinuity. Many researchers have developed schemes for interactive mesh cutting that attempt to reduce the number of new primitives created, create new primitives with good aspect ratios, avoid a disconnected mesh structure between primitives in the cut path, and represent the path traversed by the tool as accurately as possible. For example, Junjun Pan and his colleagues described an interactive dissection approach for hybrid soft tissue models governed by extended PBD.[9]

In our research, however, we do not assume that a mesh is available. Mesh-free methods are based on a set of moving simulation nodes that interact with each other according to the governing equations of elasticity. To use them effectively in cutting applications, it is necessary to define a discontinuity model and apply a boundary surface definition scheme.[2] Xia Jin and colleagues introduced the meshless total Lagrangian adaptive dynamic relaxation (MTLADR) algorithm, which relies on spatial discretization in a form of a node cloud.[10] The cutting-induced discontinuity is modeled solely through changes in nodal domains of influence. Their work does not address performance, however. To our knowledge, no interactive cutting based on purely mesh-free methods are available in the literature.

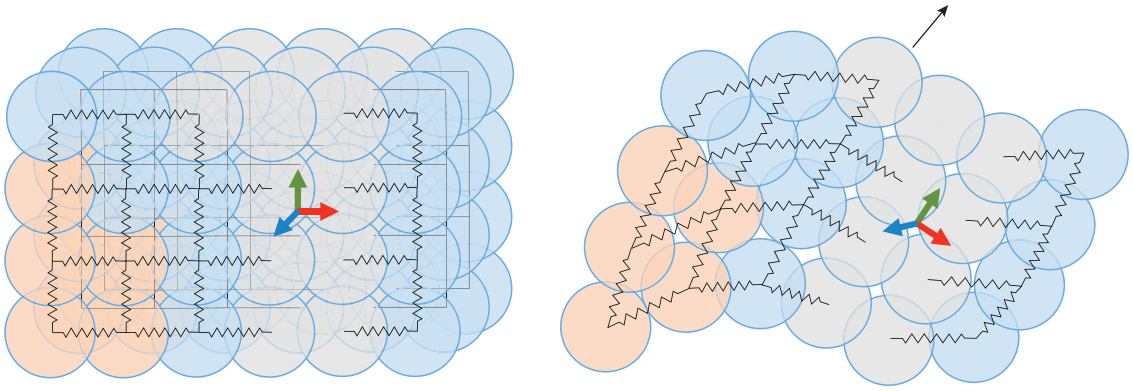
A major element in surgical cutting is defining the cut path. General methods often place seed points or a template, such as a plane or other primitive, to incorporate cuts into the model. In surgery simulations, the usual approach is to move a virtual tool through the object's surface. In their comprehensive survey, Jun Wu and his colleagues assessed the performance of virtual cutting simulators.[2] The most efficient algorithm spends nearly 80 ms per cutting step for coarse resolution versus 2,104 ms for refined resolution. The authors made clear, however, that because of the reduced degrees of freedom (DOF), the simulated deformations cannot exactly match the high-resolution reference.

Finally, many of the previous works in cutting for surgery simulation mention haptics, but no haptic algorithm has been presented to deal with the sense of touch during cutting.

Dynamic Anatomy Model

Our approach uses PBD to model the objects involved in a surgery and their dynamics. We also present the computational flow and the necessary steps to enable the simulation. We focus on detailing an integrated strategy, based on PBD, to simulate soft tissue, bone, and body fluids, including force feedback, tissue cutting, and rendering. Our representation is based on points (particles) illustrated as spheres (see Figure 1). The interaction between particles is configured as constraints within the PBD model. The simulation parameters are application-dependent. These parameters can be obtained from different sources, such as a tomographic dataset, or can be user-defined.

Our goal is not to present an accurate parameter-extraction algorithm to model the anatomy using PBD. Instead, we intend to show that, given



(a) (b)

Figure 2. Material mapping. Each particle represents a portion of a tissue with different characteristics. Applying an external force onto a particle (black arrow) will propagate the effect. The gray particles represent a rigid body, which is a rigid cluster of particles.

a representation of the human body using particles, it is possible to obtain a convincing surgery simulation. Moreover, we intend to show that the application of a specific set of steps using PBD lets us represent a larger than ever region of the human body with high efficiency and visual quality.

Efficient PBD implementations are currently available in engines dedicated to producing visual effects in movies and games, such as the FleX Library. Such implementations help developers build hybrid scenarios, with rigid, soft, cloth, and fluid objects using a unified set of equations. Soft bodies, for example, are obtained by defining springlike constrains based on the Hooke's law between particles. Other positional constraints let us form clusters along adjacent particles to keep a nonrigid set of mass points together. Users define the particle sizes. Large particles result in a loss of fine details on the object, but they allow us to represent larger bodies.

In surgery, dissection (cutting) is crucial. To simulate cuts on a PBD soft body, we need to modify the constraints that maintain the particles' topological organization. Moreover, we must account for the different materials (blood, bones, and surgical instruments) as each plays a role. Finally, realistic rendering and force feedback must be computed, all in a few milliseconds.

Material Mapping

A volumetric model is more suitable with PBD than triangle meshes. To build organs and fluids, we use the photographic images from the Visible Human Project to extract the particles' proprieties. The stack of slices forms a 3D volume that contains the body anatomy. For each voxel, we create a particle that receives the 3D texture coordinate for that location. Some segmentation approach must be applied to tag each particle with a specific tissue (such as blood vessel, bone, muscle, and fat). We used manual segmentation for our case studies, but the segmented dataset from the Visible Human Project could be used or another method could be applied.

Each material is configured with a set of proprieties: how fast it can be cut, its mass, whether it's soft or rigid, and whether it's solid or liquid. This is a simplification of the actual physical properties used to enable real-time manipulation. PBD is a simplification of physics, but larger numbers of parameters and particles lead to increased realism. Figure 2 illustrates particles colored according to the different materials.

Skinning

Although deformation can be simulated with a reduced set of particles, a huge number of particles would be necessary to mime the atoms of real objects and provide a smooth and continuous macroscopic surface for visualization. Because it is impossible to compute the dynamics of such a large number of particles with today's computers, we must use a reduced number of particles combined with some adapted rendering algorithm. For example, to render the tissue in Figure 1c, the particles used are the same used to compute the dynamics (Figure 1a), which is comparatively infinitesimal given the number of actual pixels.

To enable a smooth and continuous surface for rendering based on fewer particles, we use a triangular mesh enclosing the group of particles that describe a body (see Figure 3) in an approach similar to marching cubes (MC). However, because the volume deforms and can be cut, we must recompute the mesh at each frame. Extracting a surface from a volume in motion using previous MC methods can introduce visual perturbation when the surface crossing a MC cell changes, requiring a different triangulation. For example, Figure 3a shows some MC triangulation cases. Note that the

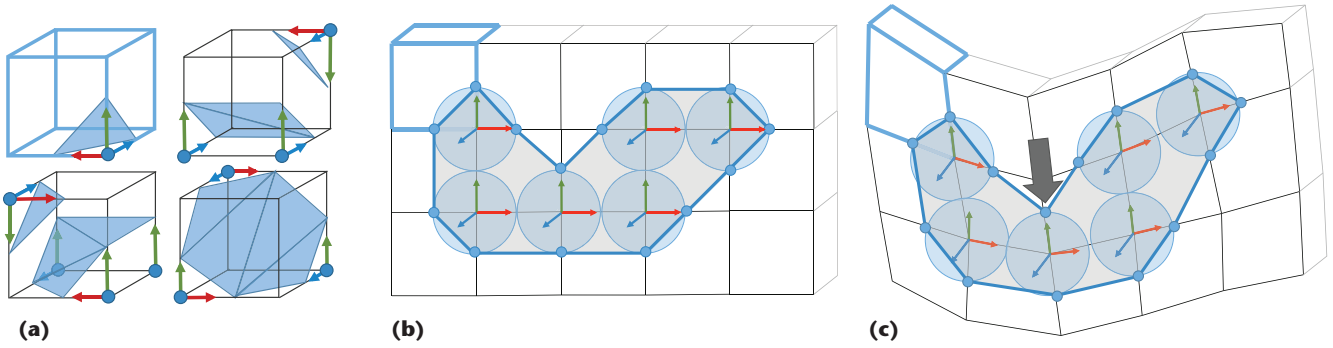


Figure 3. Surface extraction from an object formed by a group of particles. (a) The marching cube examples show that triangulation may differ considerably from one case to the other. (b) Given the initial configuration, changes in the particle's position may imply the use of different cases on consecutive frames, resulting in visual artifacts if a static grid is used. (c) The particle's movement is followed by the cells, which lets us use the same triangulation to extract the surface.

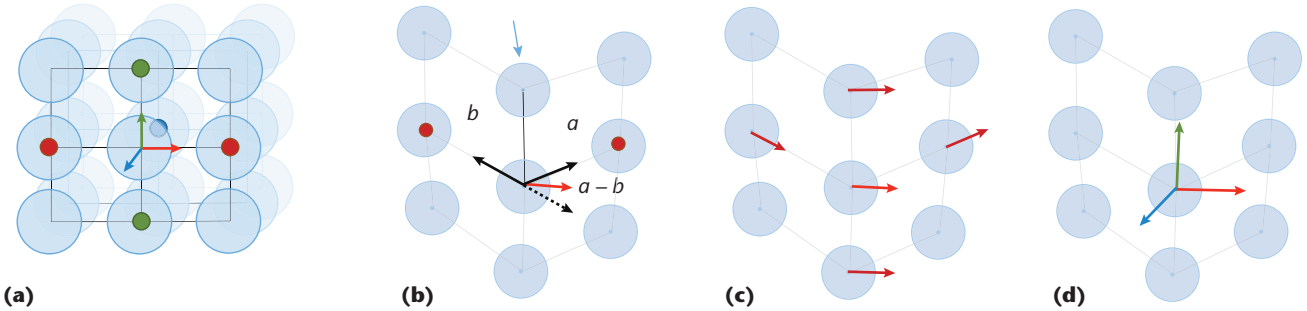


Figure 4. Computing an orientation (directions) for the cell edges based on neighboring particles. (a) In the initial particle distribution, the cells align with the world axis. (b) An external force (arrow) moves the particles, causing them to require a new orientation. Three different directions are computed and then averaged to obtain the final orientation. First, a direction is computed from the vectors formed with the neighbors. (c) Next, another direction is computed, averaging the previous orientations of the neighbors. (d) Finally, when some neighbor particles have been removed during cutting, the component corresponding to the blue arrow cannot be computed accurately in the previous cases. Then, the component can be better estimated from the cross product of the remaining components.

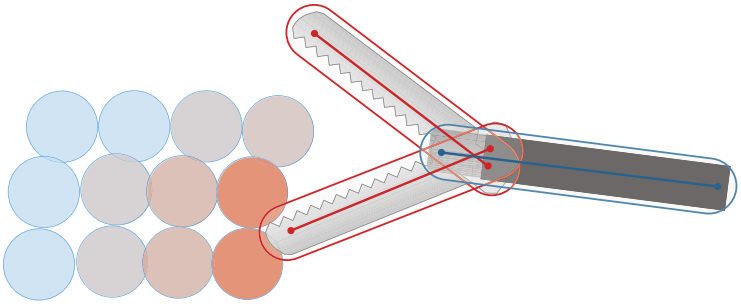
triangulation may differ considerably from one case to the other.

The mesh extracted for rendering must always enclose all the body's particles. Based on this premise, we developed a method to avoid retriangulation until a cut occurs. Figure 3b illustrates the MC grid and the particles' positions; only in this initial configuration are all the cell edges aligned. Figure 3c illustrates a situation after an external force is applied (arrow) that changes the particle positions. This eventually changes the resulting surface (blue polygon). However, the triangulation cases for each cell are the same as before, avoiding abrupt visual perturbation on the surface.

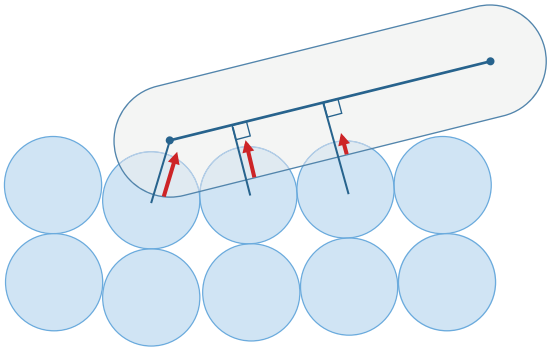
To compute the MC cells, we need the particles' positions and orientations. The orientation is an axis centered within the particle and used to define the cell edges. It is based on the neighboring particles, which are defined in the initial body construction. Only a cut can change this topological bind. The intent is to allow cells to reshape following the movement of the particles directly involved and the surrounding particles.

Initially, the MC grid is regular, as in Figure 3b; all particles have the same orientation. During simulation, the particle positions change, and the orientation is recomputed. The axes remain aligned with the cells' edges after deformation, as in Figure 3c. As we mentioned earlier, the goal is to continue extracting a surface around the body (group of particles) without using a different triangulation. We accomplish this by maintaining the isosurface crossing the same edges of an MC cell. The surface will bend, following particle movement as the cells reshape, but this results in a smooth surface deformation instead of an abrupt change. This is an extremely important feature because human users will notice an abrupt perturbation during simulation and could misread it as a medical problem. This intentionally irregular distribution is a unique peculiarity and a benefit in terms of our method's stability and performance compared with finite elements.[3]

After the orientation computation (Figure 4), we generate the triangular mesh using the MC triangulation cases only on the cells with at least one



(a)



(b)

Figure 5. Tool interaction simulation. (a) Particles touching the tool receive heat that is transmitted to the next particle. A cut is made when a particle hits a specific heat threshold, depending on the material. (b) We compute the force feedback (arrows) as the result of all vectors departing from the center of the particles under collision with the tool and the closest point in the tool.

missing particle from the eight possible options. Also, for each generated vertex in the mesh, we associate the 3D texture coordinate from the particle that originated the cell edge where the vertex is generated.

Interactive Cutting

During laparoscopy surgery, the surgeon manipulates organs and tissues using various instruments. We modeled some of these tools (grasper, cautery hook, needle tip cautery, and scalpel) and coupled the graphical model of the tools with a simpler physical model that is used for collision detection. This is necessary because the haptic feedback must run at approximately 900 Hz to provide a smooth feel. Figure 5a depicts one of these tools and its collision capsules.

In the real world, each of these tools is a potential electrode for monopolar electrosurgery. In this type of surgery, another larger electrode is placed on a large flat area of the body (usually the back) to close the circuit. The surgeon uses a pedal to activate the selected electrocautery function when necessary. Upon activation, an alternated current flows through the patient's body, encountering higher resistance at the tool tip location. This resistance causes heat to be produced inside the living cells. Depending on the type of current used, heat is produced at different speeds, and a cautery tool can function as a coagulation device (slow heat) or a cutting device (fast heat). If the intracell temperature reaches 70°C to 80°C, protein denaturation will occur and coagulation (white coagulation) will result. If the temperature reaches 90°C, the cells will lose their water content (dehydration) but will keep their architecture (desiccation). When the temperature rapidly reaches 100°C, the water contained in the cells boils. The subsequent formation of vapor and the quick expansion of the intracell volume create a cell "explosion," so the cut effect is determined by this vaporization.

Figure 5a illustrates our simulation of this process. Heat is produced at the contacting particles and dissipates throughout the tissue. The heat propagation is governed by

$$\text{newHeat}_i = h_i \text{dissipation}_i + \sum_{j \in \text{neighbors}} (h_j - h_i) \frac{(\text{cond}_j + \text{cond}_i)}{2} \Delta t.$$

The speed of heat generation is empirical in our model. However, as with the other parameters, a differential model based on physics could be used for higher physical accuracy. We modeled each material (including muscle, fat, and bone) with a different heat-response threshold. Although we used empirical parameters to define these thresholds as well, they are physically supported by the material's density and the amount of water present. When the temperature reaches the threshold, the particle starts to lose mass until it disappears. A cut can be made faster by increasing the current passing through the tool or on materials with a lower heat threshold. Because the electrosurgery process always removes material, the cut accuracy depends on the particle density and tool resolution.

Force Feedback

A Phantom Omni, which also gives force feedback, provides the input for the tools. During simulation, a collision test is computed in parallel to allow a higher frame rate, which is needed for the haptic feedback.

We compute the force feedback as the result of all vectors departing from the center of the particles under collision with the tool and the closest point in the tool. Figure 5b illustrates this process. The resulting vector direction gives the direction of the feedback and the amount of penetration gives the amount of force. The force feedback uses the same process to handle solids and fluids be-

cause both are modeled as particles. This allows the feedback to account for blood and flesh.

Results

To provide face validity for our methods, we analyzed common surgery scenarios simulated using our framework. Here, we provide an objective assessment of the performance obtained with these scenarios, providing visual simulation results for each. We also compare our results with those from recent previous works.

Surgical Scenario

In our face validation, the goal was to demonstrate the proposed approach's ability to simulate several key scenarios in real time (more than 30 frames per second). The first scenario, depicted in Figure 6, consists of a piece of fatty muscle where straight cuts are made by varying the electrocautery parameters. The three different cuts were obtained by setting different blends (on/off cycles of the high-frequency current) to provide more cutting (left) or more coagulation (right). Note the similarity between the cuts in the real photograph and in the simulation.

The second scenario provided a testbed for multiple materials, including bone, muscle, fat, and blood (fluid). Figures 7a to 7d show a sequence of cutting interactions. The tissue volume is built from the Visible Photographic female (VIP-woman) dataset. We selected the shoulder region for this scenario because it contains a variety of tissues (bones, fat, and muscle). Each tissue is set

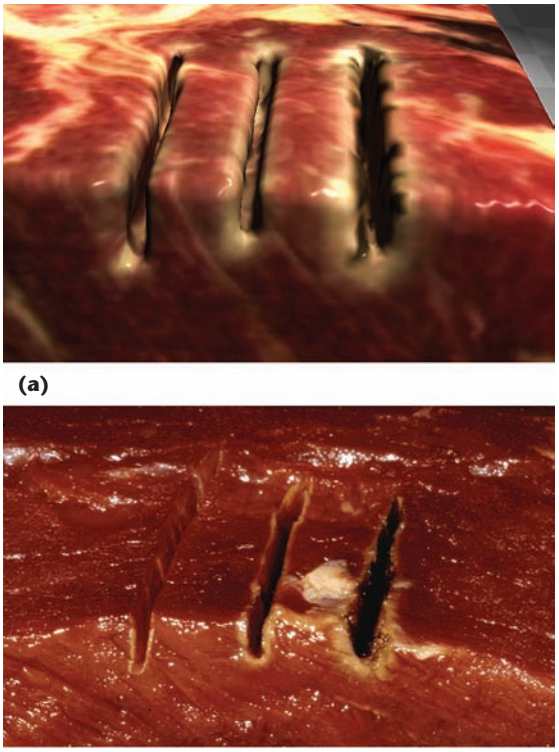


Figure 6. Comparing the (a) simulated electrocautery with (b) a real photograph. Three different cuts are obtained by setting different blends (on/off cycles of the high-frequency current): from left to right, continuous cut (100 percent and 0 percent), blend 2 (80 percent and 20 percent), and blend 3 (60 percent and 40 percent). The total amount of current must be the same for all blends, so voltage is proportionally increased in blends 2 and 3.

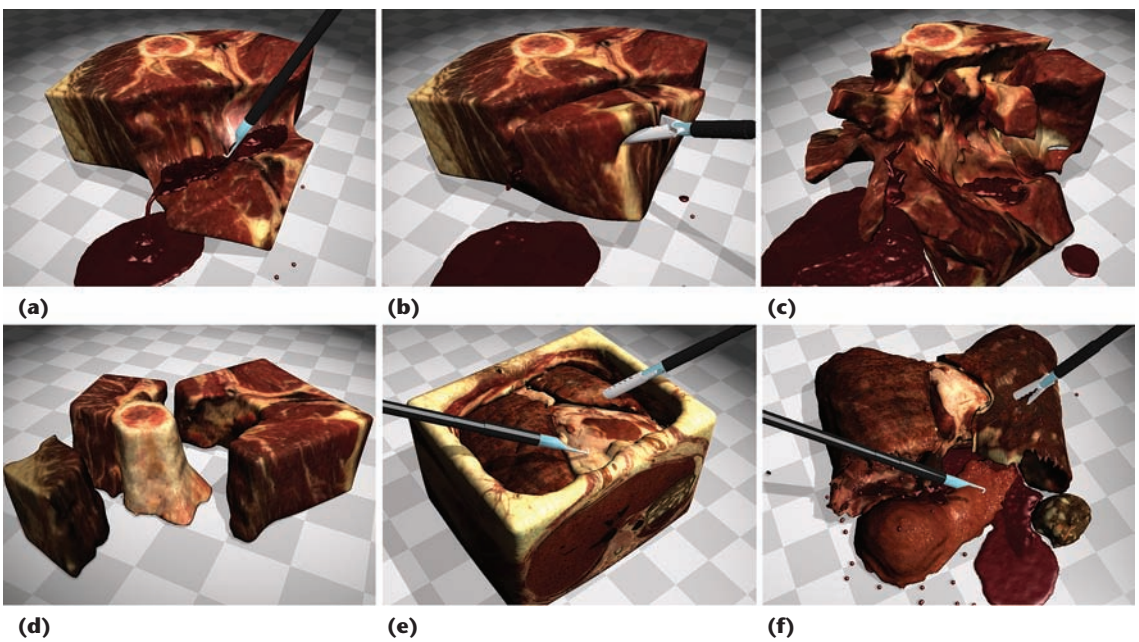


Figure 7. Visual results from a cutting simulation with multiple materials: (a) cutting through muscle and fat causes bleeding; (b) after cutting, a grasper pinches the tissue; (c) tissue after multiple cuts; (d) separating bone from surrounding tissues; (e) thorax reconstructed from the dataset; and (f) thorax after bone removal. In the last two examples (e and f), all the organs and other anatomical structures are present. In all cases, everything in the simulation generates haptic feedback and is interactive, dynamic, and subject to cutting. Nearly 10,000 particles and 30,000 constraints were simulated in real time.

Table 1. Complexity and performance values from different scenarios.

Figure	Triangles	Springs	Particles			Frame time	
			Soft	Fluid	Rigid	GPU (ms)	Total (ms)
7a	74,038	215,323	74,038	5,155	0	22	30
7b	74,038	215,323	74,038	6,582	0	23	32
7c	67,065	189,399	67,065	7,205	0	23	33
7d	21,236	78,624	24,217	0	3,771	21	29
7e	103,288	271,575	99,176	0	0	22	34
7f	55,644	194,262	63,043	6,422	0	22	34

Table 2. Simulation times for our method and previous works.*

Elements	Method		
	Pan method (ms)	Wu method (ms)	Our method (ms)
2,385	8.1		4.2
4,079	19.3		5.3
13,559		78.2	8.0
40,080		613.3	12.8

The Wu method[2] used composite finite elements, and the Pan method[9] used PBD to model body dynamics and cutting.

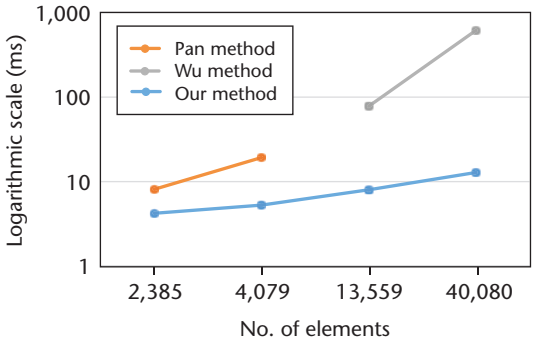


Figure 8. Comparison between our simulation and previous works. The Wu method[2] used composite finite elements, and the Pan method[9] used PBD to model body dynamics and cutting. Given the same number of elements (particles) in the model, our performance is higher and scales better. Our method is also faster and scales better as the model grows more complex.

with different response parameters to heat. That is, fat cuts easily, whereas bone barely responds to electrocautery, allowing a sharp dissection around it (see Figure 7d). For the blood-flow simulation, drops of fluid are generated at a specific point inside the tissue. After generation, blood interacts with the other tissues and tools; blood also produces haptic feedback.

In the third scenario, we demonstrate the level of complexity our framework can handle in real time. Figures 7e and 7f show screenshots from a bimanual interaction with a complete model of a thorax. This model was also reconstructed from the VIP-woman dataset. It includes the organs of the thorax (lungs, heart, liver, and spleen), vessels, and musculoskeletal structure. In this scenario, we also included heartbeats. The heartbeats are not intended to accurately simulate the heart motion. Rather, we apply periodic particle impulses toward the center of the organ, which helps demonstrate our framework's ability to handle complex scenarios in real time. (See the supplemental online video for a demonstration of the dynamic behavior in this scenario: https://youtu.be/7PxWbHDTgVg.)

Performance

We implemented the simulations for these three scenarios using C++ and OpenGL. For the performance results, we used an Intel i7 CPU and an Nvidia GTX 780 GPU.

Table 1 shows the number of triangles generated to render each scene, the number of constraints connecting particles, and the number of particles describing the soft, fluid, and rigid regions. It also shows the total computation time and the time to compute the particle dynamics on the GPU.

Table 2 shows that, given the same number of elements (particles) in the model, our method achieved better performance and scaled better than the most competitive previous works.[2,9] The PBD-based methods, such as ours and the Pan method,[9] are faster than the FEM, although the latter is more physically correct. Interestingly, in addition to being the fastest, our simulation performance decreases slowly as the model complexity grows, which Figure 8 clearly shows. This demonstrates that our approach is scalable.

Although most previous works on simulating soft tissue cuts focused on improving the performance and accuracy of physically correct models, they are only able to handle a single organ or a few small organs at interactive rates. Any further improvement will greatly depend on

the availability of more powerful hardware. The next generation of surgical simulators, however, will benefit more from plausible scenarios that are complete in terms of anatomy, behavior, and interaction, even if they are not guaranteed to be physically correct.

References

1. C.D. Bruyns et al., "A Survey of Interactive Mesh-Cutting Techniques and a New Method for Implementing Generalized Interactive Mesh Cutting Using Virtual Tools," *J. Visualization and Computer Animation*, vol. 13, no. 1, 2002, pp. 21–42.
2. J. Wu, R. Westermann, and C. Dick, "A Survey of Physically Based Simulation of Cuts in Deformable Bodies," *Computer Graphics Forum*, vol. 34, no. 6, 2015, pp. 161–187.
3. M. Müller et al., "Position Based Dynamics," *J. Visual Comm. and Image Representation*, vol. 18, no. 2, 2007, pp. 109–118.
4. M. Macklin and M. Müller, "Position Based Fluids," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, article no. 104.
5. J. Bender et al., "A Survey on Position-Based Simulation Methods in Computer Graphics," *Computer Graphics Forum*, vol. 33, no. 6, 2014, pp. 228–251.
6. M. Macklin et al., "Unified Particle Physics for Real-Time Applications," *ACM Trans. Graphics*, vol. 33, no. 4, 2014, article no. 153.
7. A. Maciel et al., "Using the PhysX Engine for Physics-Based Virtual Surgery with Force Feedback," *Int'l J. Medical Robotics and Computer-Assisted Surgery*, vol. 5, no. 3, 2009, pp. 341–353.
8. K.-S. Choi, "Interactive Cutting of Deformable Objects Using Force Propagation Approach and Digital Design Analogy," *Computers & Graphics*, vol. 30, no. 2, 2006, pp. 233–243.
9. J. Pan et al., "Real-Time Haptic Manipulation and Cutting of Hybrid Soft Tissue Models by Extended Position-Based Dynamics," *Computer Animation and Virtual Worlds*, vol. 26, nos. 3–4, 2015, pp. 312–335.
10. X. Jin et al., "Meshless Algorithm for Soft Tissue Cutting in Surgical Simulation," *Computer Methods in Biomechanics and Biomedical Eng.*, vol. 17, no. 7, 2014, pp. 800–811.

Iago Berndt *is a master's student in computer science in the Institute of Informatics at the Federal University of Rio Grande do Sul (UFRGS), Brazil. His research interests include real-time computer graphics, surgery simulation, and 3D user interfaces. Berndt has a BS in computer science from the Federal University of the Southern Border (UFFS). Contact him at iago.berndt@inf.ufrgs.br.*

Rafael Torchelsen *is an associate professor in the Center of Technological Development at the Federal University of Pelotas (UFPel), Brazil. His research interests include real-time computer graphics, GPUs, and human-computer interfaces. Torchelsen has a PhD in computer science from UFRGS. Contact him at rafael.torchelsen@inf.ufpel.edu.br.*

Anderson Maciel *is an associate professor in the Institute of Informatics at UFRGS, Brazil. His research interests include soft tissue and surgery simulation, virtual reality, and human-computer interfaces. Maciel has a PhD in computer science from the Swiss Federal Institute of Technology (EPFL). Contact him at amaciel@inf.ufrgs.br.*

Contact department editor Mike Potel at potel@wildcrest.com.