



# Dynamic Cutting of a Meshless Model for Interactive Surgery Simulation

Vincent Magnoux<sup>✉</sup> and Benoît Ozell<sup>✉</sup>

Polytechnique Montréal, Montréal, Canada  
benoit.ozell@polymtl.ca

**Abstract.** Virtual reality has become a viable tool for training surgeons for specific operations. In order to be useful, such a simulation need to be as realistic as possible so that a user can believe what they experience and act upon the virtual objects. We focus on simulating surgical operations that require cutting virtual organs. They offer a particular set of challenges with respect to simulation stability, performance, robustness and immersion.

We propose to use a fully continuous movement representation and collision detection scheme between cutting tool and other simulated objects to improve the robustness of the simulation and avoid breaking immersion with errors in topology changes. We also describe a new way to attach the surface of a simulated object to its underlying physical model, consistently while it is being cut. This feature helps maintain immersion by keeping the visual aspect of the object coherent with its physical behavior and by allowing correct transmission of actions from the user on the object.

Our tests show that the proposed tool movement representation properly generates continuous cuts in simulated models, even as they move and deform. It also allows cutting when a moving model comes into contact with a motionless tool, and to model a curved or deforming tool without additional effort. Our surface mapping method results in a visual model that closely follows the movement and deformation of the physical model after it has been cut.

**Keywords:** Surgery simulation · Virtual reality · Cutting simulation · Physically-based simulation · Meshless methods

## 1 Introduction

Surgery simulation technology using virtual reality has become a useful and practical tool for learning specific operations like lung tumor removal [33], aneurysm clipping [2] and many others [20]. As with all virtual environments, an important aim for these simulations is to create a world in which the user can feel immersion and presence as much as possible. This sensation may come from how the world is perceived by the user's senses, such as vision, hearing and touch, and from how the world reacts to the user's actions. The research presented in this paper

focuses on the latter aspect, more specifically on how virtual objects like organs are modified through cutting actions from the user during a surgical operation.

There has been a lot of progress in cutting simulation, both on the physical and the visual aspects, yet there remain open challenges to improve the realism of this interaction. Our goals are to develop a tool representation that increases the *robustness* of the simulation during a cutting action and to maintain *coherence* between the visual surface and the underlying physical model at the same time. Together, these objectives will improve simulation interactivity by providing a more accurate representation of hand movement, avoiding breaks in immersion in a straightforward way and ensuring a realistic reaction of soft tissue to user action.

On the one hand, we improve the robustness by ensuring that any part of a simulated object traversed by a cutting tool will indeed be cut. This means that no geometric element will be missed due to the discrete nature of the simulation or to the movement and deformation of either the object or cutting tool.

On the other hand, we preserve coherence between the visual and physical models by elaborating a set of criteria that determine how the surface mesh moves with the physical model and how it can transmit forces to it.

## 1.1 Background and Related Work

Soft body simulations that allow cutting may be classified according to the physical simulation method, which may either be mesh-based or meshless. A thorough review of the various cutting methods may be found in [35]. Mesh-based cutting involves removing and recreating elements along the tool trajectory [5, 6, 11, 12, 28, 34], while meshless methods rely on updating the connection or visibility between the particles that form the physical object [13, 17, 25, 26, 31]. For both simulation categories, cutting requires modeling the trajectory of the tool as well as finding the intersection of the tool with the object along that trajectory. At the lowest level, this intersection is often computed as the contact between edges or between triangles and points of either model.

**Tool Representation.** Most simulations represent the cutting part of the tool as a one-dimensional edge composed either of a single segment [9, 21, 23] or multiple ones [6, 32]. The area between the position of the cutting edge during the previous frame and its position during the current frame is considered the *swept area* [32]. It may be approximated by a plane [9, 21, 23] or be triangulated to obtain a more closely fitting cutting surface [10, 13, 15, 24, 31].

One major drawback of this approximation is that it does not capture well the movement of the cutting edge when it does not lie in the same plane between two frames. This may cause some contact detection to be missed, eventually resulting in erroneous modifications in the modeled object surface. [21] propose to use a continuous representation of the cutting edge, which partially offsets this problem by providing a more realistic approximation of the edge movement. However, it still fails to take into account the movement and deformation of the object being cut.

There are however approaches that go in a different direction. A single point with an orientation may be used to represent the tool [4,30]. Together, the position and orientation of the point define a plane which can be used to compute level sets at every frame, which in turn determine where different material points are with respect to the tool. In other cases, some represent the tool with a volume and check for an instantaneous (discrete) intersection between the tool volume and the object volume [1,28]. In a similar way, [11] uses as a cutting surface the intersection of the tool volume with the object, represented with voxels – resulting in an approximate cut. Finally, rather than using a well-defined geometric shape, the tool volume may simply be sampled with a set of points [26].

**Surface Mapping.** In general, simulations may either use an implicit or explicit surface representation to draw an object on the screen. However, implicit representations usually require a lot of computing power and are thus not well suited for an interactive simulation [16,22]. Some manage to perform volume- and point-based rendering in real time on parts of a model using surface splatting [17,37] or, more recently, ray casting on the entire object [30].

Explicit surface representations must be somehow attached, or *mapped*, to the physical object model so that they can properly move according to the results of the physical simulation and continue to transmit interaction forces.

For mesh-based physical models, the trivial case consists in using the boundary faces of volume elements to display the surface [5]. However, the resolution of the surface is thus directly tied to that of the physical model. A separate representation allows to display the object in much greater detail than it is possible to physically simulate it.

With mesh-based physical models, this mapping between the surface mesh and physical object is done by selecting to which volume element a surface vertex belongs and computing its barycentric or trilinear coordinates within that element [6,11,12,14,28,29,34]. When the element is displaced or deformed, the new vertex position is computed using the mapping coordinates in the new element configuration.

With meshless models, the mapping is done by selecting a set of appropriate particles and computing a set of weights for each of them relative to the vertex. Instead of weights calculated through barycentric or trilinear coordinates – since there is no geometrically set element – a moving least squares (MLS) scheme is often used [13,15,17,26,31,36]. Other weighted mappings are also possible, based on how the displacement field is sampled in the physical simulation [8,18,25].

## 1.2 Contributions

This paper describes the following contributions:

- A fully continuous cutting interaction between a tool and a surface that includes detecting the cut primitives and modifying the surface mesh in consequence. Both the tool and the object may be deforming during that interaction.

- A small set of criteria that allow mapping the surface on its underlying volume model to maintain visual coherence.

Section 2 describes the details of these contributions, while Sect. 3 demonstrates their utility through a set of examples.

## 2 Method

### 2.1 Tool Representation and Collision Detection

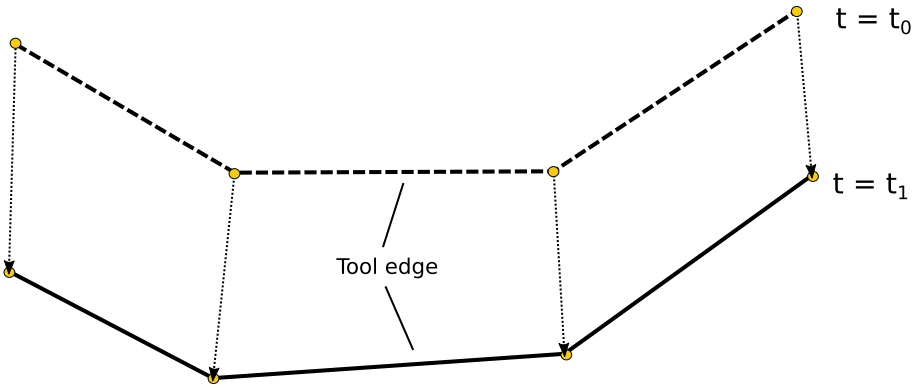
For the purpose of cutting, the interacting part of the tool is represented by a series of points joined by straight line segments, collectively called the *cutting edge*. The only interaction on which we will focus in this paper is for cutting, without exchanging forces between the tool and other simulated objects. The cutting edge is considered ideal, in the sense that it has no volume, cuts as soon as it comes in contact with a simulated object and does not generate any force on it. In addition to the cutting tool, the term *edge* will refer to the primitives used for detecting intersections. For the surface, they are the lines between surface vertices that form triangles and for the underlying physical object, they are the links between integration points and particles that form its topology.

The object itself is deformed and cut using the method described in [3] and [19]. For the purpose of topology modifications, the object is entirely composed of edges, as defined above. Cutting only occurs when edges are intersected, not when a point enters a triangle on the surface. Furthermore, we only mention cutting the object’s surface, but everything discussed in this section applies in an identical way to cutting the edges that define its volume.

As the tool is displaced from one frame to the next, each of its edges form a skew quadrilateral with the four points being the two ends of the edge at the first frame and the two ends at the second frame, as shown in Fig. 1. The set of 3-dimensional quads form the surface swept by the blade during the frame. When detecting collisions, the deformed object’s surface is similarly only considered as a set of edges that move through space between animation frames. The detection is made between each pair of edges that belong to the separate objects, according to the continuous collision algorithm described in [27].

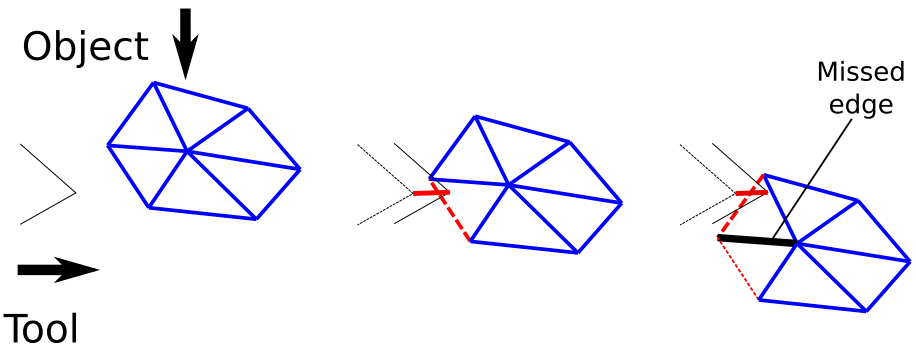
We assume each point of the blade moves in a straight line during a frame. This approximation remains accurate as long as the displacement between frames is relatively short, which is the case when animating the simulation at a rate of 60 frames per second, leaving less than 17 ms per frame.

This operation is more complex than with an explicitly-triangulated swept surface since it requires continuous edge-edge collision tests rather than just a series of discrete triangle-edge tests. However, there are several situations where it provides a correct solution, in contrast to a discrete detection scheme.



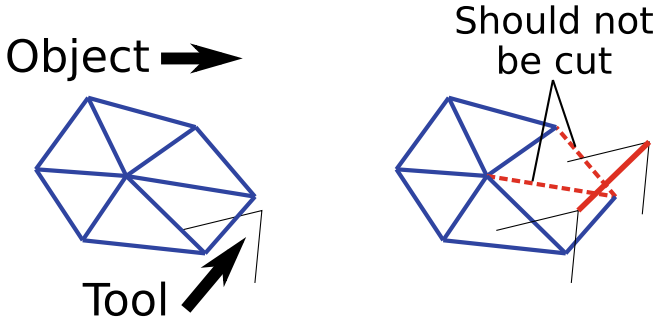
**Fig. 1.** The area swept by the cutting tool is only described in terms of the position of its edge at the previous frame ( $t_0$ ) and its position at the current frame ( $t_1$ ).

*Situation 1.* The continuous method allows for a completely gap-less detection of cut edges, which is especially useful when both objects are moving between frames. For example, an edge from the deformed object could move across an entire swept area quadrilateral during a frame time, leaving it undetected when using a discrete scheme. Figure 2 illustrates such a situation in two dimensions. The cutting tool is displayed as a wedge in this diagram, but only the tip has any physical presence in a 2D setting. Its movement traces a line – rather than a swept area in 3D – which can be used for detecting intersections with surface triangles.



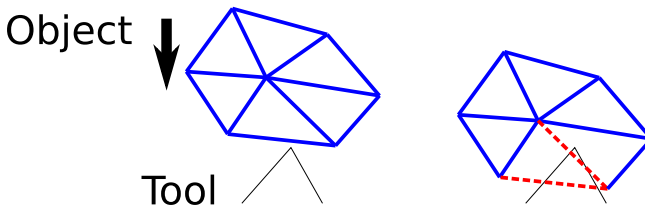
**Fig. 2.** Example of *Situation 1*, in 2D, where a collision detection scheme that is continuous only with the tool movement would fail to detect a certain triangle edge. The large arrows indicate the direction of movement of the tool and object. The red line between tool positions represents the space swept by it between frames (Color figure online).

*Situation 2.* That method also avoids detecting cuts in places where a surface edge enters the swept area *after* the blade has passed, as in Fig. 3. In that case, the tip of the cutting tool was always outside the object, but the line that connects the current position to the previous one still intersects some edges in the object at its current position.



**Fig. 3.** Example of *Situation 2*, where the line spanned by the tip of the cutting tool intersects two of the objects edges at the end of the frame, even though the tool was outside the object at every point in time between the two frames. The large arrows indicate the direction of movement of the tool and object. The red line between tool positions represents the space swept by it between frames (Color figure online).

*Situation 3.* Another case where this description is essential is when the blade is not moving but the deforming object is, whether under gravity or elastic forces. In such a case, the blade does not sweep any area, but the object must still be cut, as shown in Fig. 4.



**Fig. 4.** Example of *Situation 3*, where a simulated object moves downward onto a motionless cutting tool. The tool's edge does not sweep any area, but must still be able to cut the object.

When the tool is moving, the continuous method also provides a slightly more accurate approximation of the swept area as well as a more accurate point of contact, both in space and in time, since it is not flattened by using triangles.

Since the tool may also cut while not moving, we must take particular care with defining some quantities. For example, we might want to know on which

side of the cutting edge lie each pair of vertices. In our method, the new normal on the surface is entirely determined by the position of each intersection – the trajectory of the cutting tool relative to the object. This avoids relying on the normal of a swept surface that does not exist when the tool is motionless.

One point worth mentioning is that with the movement representation described above, the cutting tool itself can be deformed during the simulation without the need to handle that case differently.

## 2.2 Surface Mapping and Remapping

To ensure that the surface remain coherent with the movement of the physical simulated object and thus provide a proper interface with other virtual objects and with the user, we establish a set of constraints that determine how each surface vertex moves with the physical object model. These criteria are tailored to the deformation and cutting method of [19], where the particles are embedded in a regular background integration grid; however, they can be applied almost directly to any method that maps a triangulated surface on a set of particles.

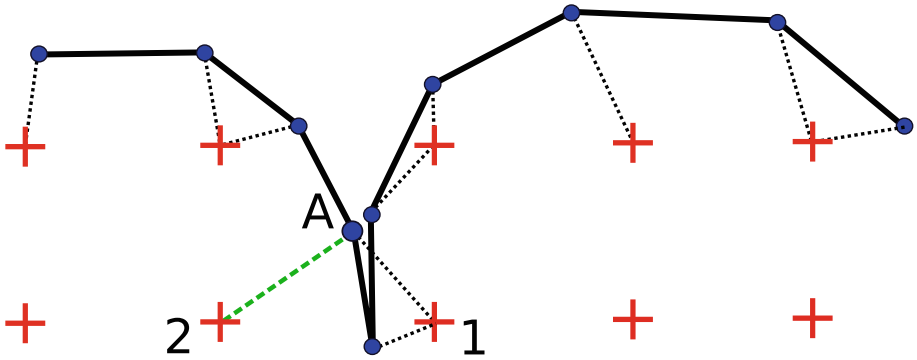
Since the displacement of a material point inside the object is determined by interpolating the displacement of neighboring particles using shape functions computed with an MLS scheme, a natural solution is to use the same scheme for surface vertices. The set of particles that determine the position of a point will be called the *mapping* of that point in the remainder of this paper.

The main challenge with this method consists in choosing the right particles on which to map a vertex. This choice does not simply depends on the distance between the vertex and particles, because we need to take into account the initial topology of the object as well as the cuts introduced with the tool. We may however choose that criterion as a starting point for the set of constraints, since it is how it would be done for a convex object that does not contain any cut.

It should be mentioned that when determining where to map a given vertex, only the rest configuration of the object is considered. While intersections are detected in the current (deformed) configuration, their location in the object at rest can be determined in a straightforward way: since they always occur on edges, between two points, they can be fully described by a proportion along the vector connecting the two points. For simplicity, we consider that proportion to always be the same, whether in the rest or deformed configuration.

To avoid mapping a vertex to particles that belong to physically separate parts of the object – while still being geometrically close – we first decide to select particles that are in the neighborhood of a single integration point. This guarantees that they will all move in a locally coherent way. The mapping problem is thus reduced to finding a single most suitable integration point for each vertex of the surface mesh. This approach is sound also because the integration elements are more closely related to the geometry of the object – they distribute its volume and mass to the particles. They are all at least partially inside the object, whereas particles may lie slightly outside the surface of the object as described in [3].

When the object contains a cut, or simply a concave portion, the integration point nearest to a certain vertex may actually be in another part of the object (see Fig. 5). To avoid any such wrong mapping, we add as a criterion that the integration point on which a vertex is mapped must lie below the plane defined by the vertex and its normal.

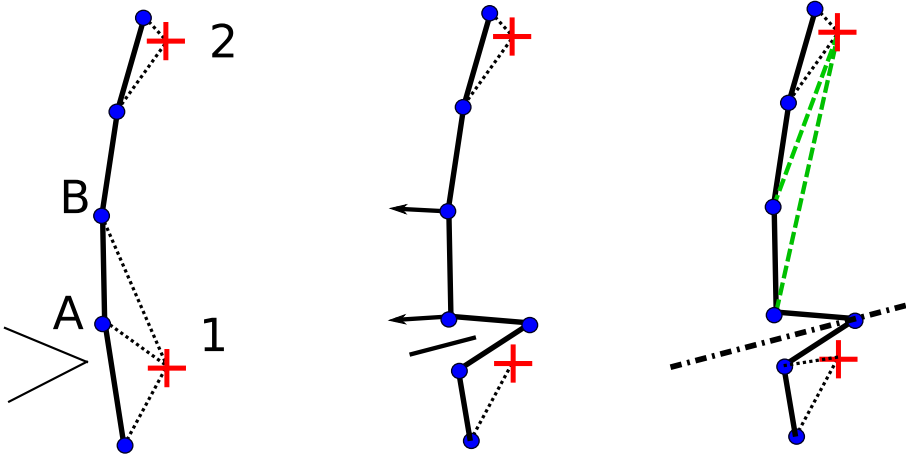


**Fig. 5.** Illustration in 2D of a situation where the nearest integration point to a certain vertex would not be the best mapping. Integration points are displayed as red crosses and vertices as blue circles, with dotted lines indicating the nearest integration point. Vertex A has integration point 1 as its closest, which lies across another surface boundary. We would rather choose an integration point that is *not* separated by another part of the surface, such as point 2, connected to the thicker green line (Color figure online).

There are however many cases where the plane test by itself is not sufficient to determine the right mapping point. For example, in Fig. 6, when the cutting tool passes between a certain vertex and the integration point on which it is mapped, using only the criteria mentioned earlier, we would try to map the vertex to the same integration point. We therefore associate with the vertex an additional plane below which the mapped integration point must also be, whenever a new cut is introduced into the object near that vertex.

The final criterion we need to define allows to take *all* introduced cuts into account – rather than just the most recent one – without specifying an arbitrarily large number of cutting planes. For each vertex, we only choose as potential mapping targets integration points which are connected to a set of particles *similar* to that of neighboring vertices. In other words, we need to consider not just the integration point, but the set of particles on which vertices are mapped. For an integration point to be considered valid, it has to have at least one particle in common with every vertex that form a triangle with the unmapped vertex. This also ensures that vertices that are close to each other will be mapped to integration points that are also close to each other, keeping the displacement of the surface locally coherent.





**Fig. 6.** Illustration of the need for an additional constraint plane when finding a good mapping. *Left:* A cutting tool approaches from the left, below vertex A. *Middle:* The introduced cut has caused vertices A and B to lose their previous mapping. Their initial normal is indicated. Using only that normal would cause them to be mapped on the integration point 1, from which they were just unmapped. *Right:* The introduced cutting plane causes vertices A and B to be mapped to point 2.

In summary, when choosing a new integration point to which a vertex will be mapped, that point must be

- connected to some of the same particles on which neighboring vertices are already mapped;
- below the plane tangent to the surface at that vertex;
- below an additional plane defined when a cut was introduced.

From the integration points that match all these criteria, the one closest to the vertex under consideration is chosen.

### 2.3 Dynamic Simulation

Whenever a part of the object is intersected by the cutting tool, all affected vertices must find a new mapping. We define in the next paragraphs which surface vertices to remap after an interaction with the cutting tool.

When a surface edge is cut, two vertices are added: they must be mapped. The endpoints of the (now cut) edge must be remapped, and receive an additional plane corresponding to the surface normal of the cut (see Fig. 6, right). There is also the case where the blade intersects the segment between a vertex and the integration point on which it is mapped. In that case, the vertex has to be remapped and be associated with an additional plane. That new plane passes through the cut point and has the vertex-integration point line as a normal, as if that line were just another surface edge.

Adjustments to the mapping also need to be made when the underlying physical model is cut. A modified connectivity between particles and integration points changes the eligibility of affected integration points in certain cases, so any vertex that is mapped to these points needs to be remapped. For example, two integration points that share a certain particle may become separated and no longer be considered as neighbors, affecting how nearby vertices must be mapped.

When the mapping of a vertex changes, its position at the next frame may change slightly, even if the object is not moving. Using a fully continuous collision detection scheme prevents any missed surface edge.

**Animation Loop.** The order in which the main steps of the simulation are carried out is important to obtain a consistent interaction between the dynamic tool and the object. An animation step in our current implementation consists of four operations:

1. Process user input. This only consists in moving the cutting tool to its new position, as determined by a haptic device, for example.
2. Detect collisions. These collisions are currently only for cutting.
3. Perform topology changes, based on the intersections detected in the previous step. This results in updated data structures describing the object – physical model, surface and mapping between the two.
4. Solve the dynamic system. This system incorporates internal elastic forces, gravity and external forces. This step also updates the surface vertex and integration point positions according to the new particle positions given by the system solution, as described in [3,19].

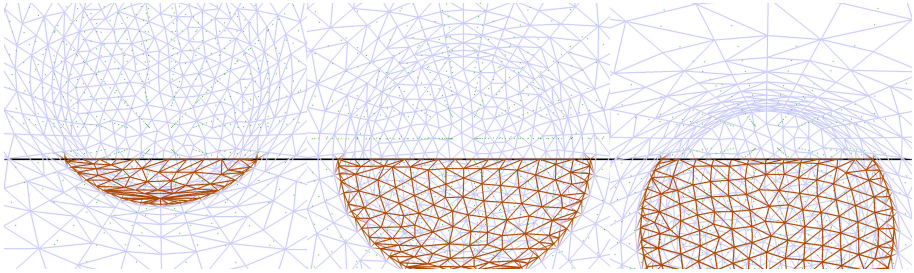
In such a scenario, input by the user is considered to occur during the displacement of surface positions, even if it is only processed after. Any movement from the user that happens during the animation step will have an effect based on the *next* surface and object position.

### 3 Results

The applicability of our method is demonstrated through an implementation using the SOFA framework [7]. We show examples of various situations described in the previous section where it is useful, both on simple geometries and on organ models. The proper functioning of the surface mapping criteria is apparent in some of these scenes, but is most visible in the accompanying video.

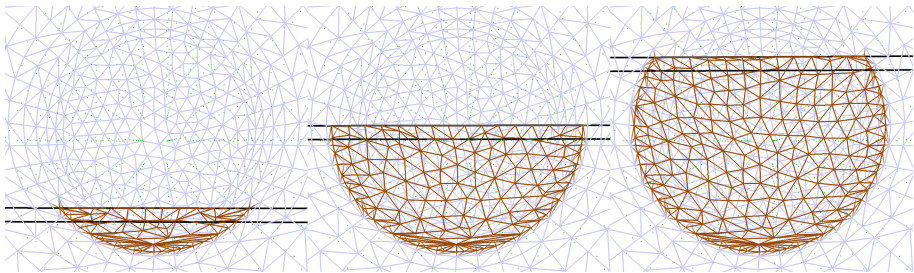
The most obvious advantage of having a fully continuous collision detection scheme, is that we can still properly detect tool-object intersections when the blade is static and the object is moving. This can occur for example under the action of gravity or elastic forces. This capability contrasts with the usual method where only the cutting edge is considered to be moving from one frame to the next. Figure 7 shows three different frames of a simulation where a circular

cylinder is falling on a simple straight cutting edge. Even if the edge does not sweep any area, the object can still be cut. This example also shows how the surface position remains consistent with the position of the object. Surface vertex positions are entirely dependent on the position of the underlying physical model, not shown in this picture.



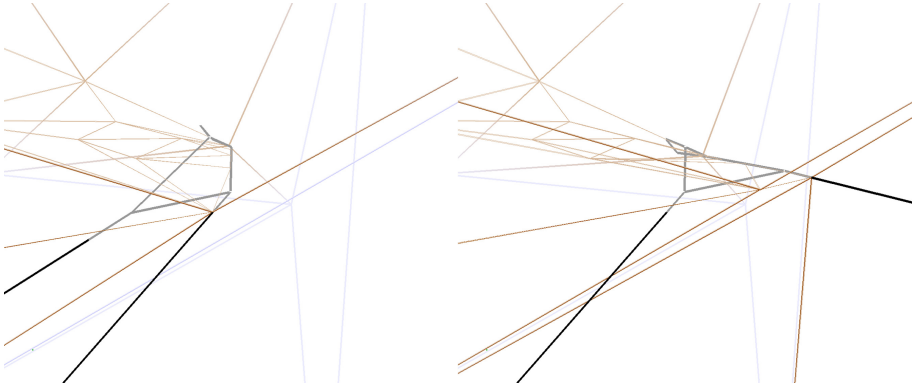
**Fig. 7.** A circular cylinder is moving downward and being cut by a horizontal edge that remains in the same place. The surface of the cylinder (including the newly-generated part) properly follows the movement of the underlying physical model.

As a comparison, Fig. 8 shows a similar scene, this time where the circular cylinder is static and the cutting edge is moving upward. One can see that the resulting cut surface is very similar to that of Fig. 7. The surface generation method remains identical; the only difference is the sequence and speed at which triangle edges are intersected.



**Fig. 8.** A horizontal cutting edge moves upward in an fixed circular cylinder and cuts it.

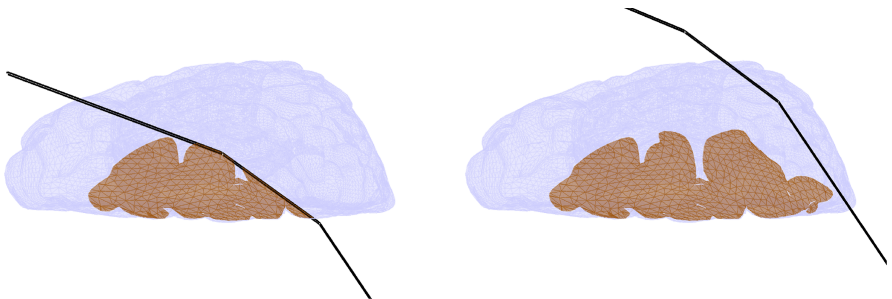
Another major advantage of the fully continuous motion modeling is that the cutting edge will never miss a triangle edge because of their movement between frames – barring any numerical error. This can occur in many situations, especially when the blade movement is almost parallel to a surface edge, or when it undergoes a twisting motion relative to the surface. Figure 9 displays such a situation, where a curved blade is rotating while advancing and cuts a surface edge that is almost parallel to its local motion.



**Fig. 9.** Cutting edge (in black) that is curved and that twists as it moves forward.

In a surgery simulation, most simulated objects are more complex than those shown so far. Our proposed tool representation and surface mapping methodology are not affected by the complexity of the model being cut, in theory. The difficulty lies mostly in how to manage the cuts that are detected. Once the surface and physical model are cut, the mapping can proceed as with any other model. Figure 10 presents a brain hemisphere being cut by a slightly curved blade. This model has many grooves and folds that the blade enters and leaves simultaneously or at different times. This gives rise to situations where the cutting edge traverses several separate parts of the model in a given frame. Our method handles these multiple fronts merging and splitting as the blade moves forward the same way as it does a single front.

Figure 11 shows that a single object may be cut multiple times with the same tool. In the first image, the cut surface appears to be slightly curved, because the scene was animated during the cutting operation and the liver was slowly tilting

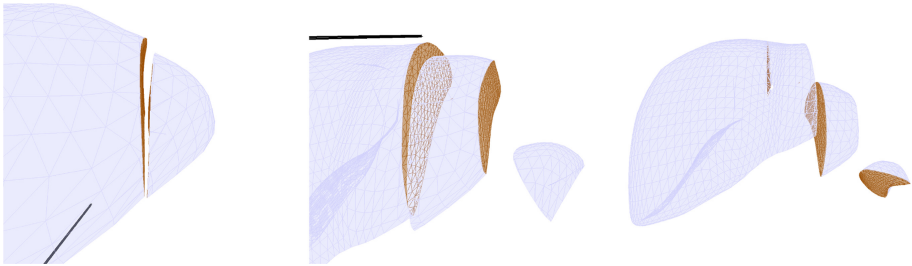


**Fig. 10.** Cutting a more complex model that may appear in a surgical simulation scenario. The cutting tool is able to deal with the numerous folds present in a human brain.

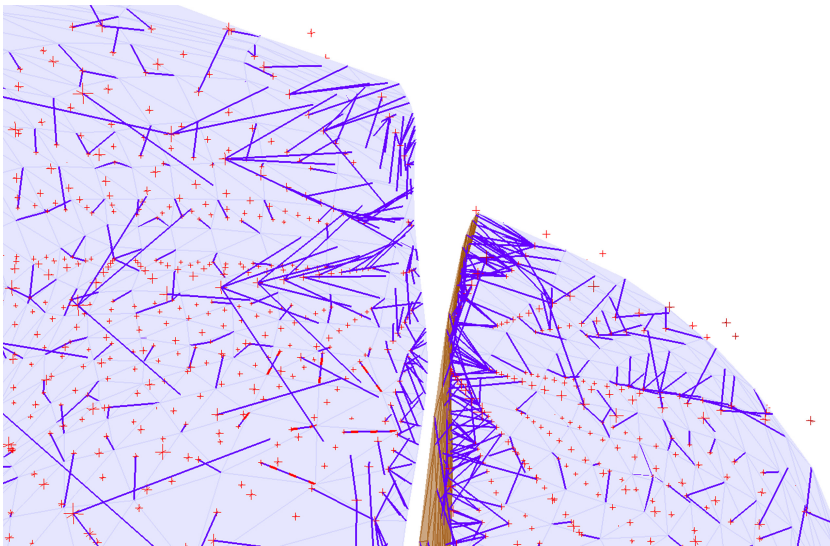
down as the blade went through it. Figure 11 also shows that the displacement of the surface remains consistent with the cuts introduced by the tool.

Figure 12 offers a more detailed perspective on how surface vertices are mapped on the underlying physical model. On the cut surface, it shows that each mapped vertex is linked to an integration point on its own side of the cut. That point is generally the closest that fits all mapping criteria, taking the new cut into account.

**Limitations.** The correctness of mapping a surface based on the criteria described in Sect. 2 depends greatly on the relative size of the surface mesh



**Fig. 11.** Illustration of a series of cuts in the same object. A slight curve is noticeable in the first cut and is caused by the movement of the liver under the effect of gravity while the cut was performed.



**Fig. 12.** A view of the liver cut scenario that displays the integration points (red crosses) and highlights to which point each vertex is mapped (blue lines). Each vertex is properly mapped to a point on its own side of the cut (Color figure online).

and that of the physical model on which it is mapped. The surface mesh *must* be at least as dense as the integration point grid. Otherwise, it will not be possible to find a point near a vertex that is somewhat close to every triangle formed by that vertex.

A related issue is that of the smallest possible cut in a model. While it is possible to create and display a surface with an arbitrary level of detail, it is not possible to cut these details away from the rest of the simulated object if there are not enough integration elements under them. If a sufficiently small part of the surface is separated from the rest of the object, it will not be possible to map it to the underlying physical model. With the scheme used for this research for distributing particles and integration points, the thinnest part of an object that can be fully separated from the rest and still be properly mapped is about the width of two integration elements.

## 4 Conclusion and Future Work

We have described and demonstrated the use of a dynamic tool representation that allows to interactively cut a deforming object in a variety of situations: with a static, moving or deforming tool on a static, moving or deforming surface. We have also described a method to successfully map the cut surface of the object on its underlying cut physical model, itself based on a set of particles and a background mesh.

Together, these two methods lead to a more robust and coherent simulation, improving the immersion and interactivity of the virtual environment in a surgical setting. This work forms a component that will become part of an existing simulation system.

That system displays a surgical scene on a 3D screen, offering sufficient visual immersion for accomplishing tasks on the relatively small work area of the surgery site. It is also equipped with a haptic device that provides a virtual surgical tool as the single means through which the user can interact with the scene. Using that system will enable us to make our method available for testing by surgeons and will be the subject of future research.

**Acknowledgement.** This work is funded by the Natural Sciences and Engineering Research Council (NSERC) under grant 501444-16, in collaboration with OSSimTech.

## References

1. Agus, M., Giachetti, A., Gobbetti, E., Zanetti, G., Zorcolo, A.: Real-time haptic and visual simulation of bone dissection. Presence: Teleoper. Virtual Environ. **12**(1), 110–122 (2003). <https://doi.org/10.1162/105474603763835378>
2. Alaraj, A., et al.: Virtual reality cerebral aneurysm clipping simulation with real-time haptic feedback. Neurosurgery, 1 (2015). <https://doi.org/10.1227/NEU.0000000000000583>

3. Brunet, J.N., Magnoux, V., Ozell, B., Cotin, S.: Corotated meshless implicit dynamics for deformable bodies. In: International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 91–100. Pilsen, Czech Republic, May 2019. <https://doi.org/10.24132/CSRN.2019.2901.1.11>
4. Cheng, Q., Liu, P.X., Lai, P., Xu, S., Zou, Y.: A novel haptic interactive approach to simulation of surgery cutting based on mesh and meshless models. *J. Healthcare Eng.* **2018**, 1–16 (2018). <https://doi.org/10.1155/2018/9204949>
5. Cotin, S., Delingette, H., Ayache, N.: A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *Vis. Comput.* **16**(8), 437–452 (2000). <https://doi.org/10.1007/PL00007215>
6. Dick, C., Georgii, J., Westermann, R.: A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Trans. Visual. Comput. Graph.* **17**(11), 1663–1675 (2011). <https://doi.org/10.1109/TVCG.2010.268>
7. Faure, F., et al.: Sofa: a multi-model framework for interactive physical simulation. In: *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, pp. 283–321. Springer, Heidelberg (2012). [https://doi.org/10.1007/8415\\_2012.125](https://doi.org/10.1007/8415_2012.125)
8. Faure, F., Gilles, B., Bousquet, G., Pai, D.K.: Sparse meshless models of complex deformable solids. In: *ACM SIGGRAPH 2011 Papers*. pp. 73:1–73:10. SIGGRAPH 2011, ACM, New York, NY, USA (2011). <https://doi.org/10.1145/1964921.1964968>
9. Gutiérrez, L.F., Ramos, F.: XFEM framework for cutting soft tissue including topological changes in a surgery simulation. In: *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pp. 275–283. Angers, France, January 2010. <https://doi.org/10.5220/0002836402750283>
10. Holgate, N., Joldes, G.R., Miller, K.: Efficient visibility criterion for discontinuities discretised by triangular surface meshes. *Eng. Anal. Boundary Elements* **58**, 1–6 (2015). <https://doi.org/10.1016/j.enganabound.2015.02.014>
11. Jeřábková, L., Bousquet, G., Barbier, S., Faure, F., Allard, J.: Volumetric modeling and interactive cutting of deformable bodies. *Progress Biophys. Molecular Biol.* **103**(2–3), 217–224 (2010). <https://doi.org/10.1016/j.pbiomolbio.2010.09.012>
12. Jia, S.-Y., Pan, Z.-K., Wang, G.-D., Zhang, W.-Z., Yu, X.-K.: Stable real-time surgical cutting simulation of deformable objects embedded with arbitrary triangular meshes. *J. Comput. Sci. Technol.* **32**(6), 1198–1213 (2017). <https://doi.org/10.1007/s11390-017-1794-z>
13. Jung, H., Lee, D.Y.: Real-time cutting simulation of meshless deformable object using dynamic bounding volume hierarchy. *Comput. Animat. Virt. Worlds* **23**(5), 489–501 (2012). <https://doi.org/10.1002/cav.1485>
14. Kim, Y., et al.: Deformable mesh simulation for virtual laparoscopic cholecystectomy training. *Vis. Comput.* **31**(4), 485–495 (2014). <https://doi.org/10.1007/s00371-014-0944-3>
15. Li, S., Zhao, Q., Wang, S., Hao, A., Qin, H.: Interactive deformation and cutting simulation directly using patient-specific volumetric images. *Comput. Animat. Virt. Worlds* **25**(2), 155–169 (2014). <https://doi.org/10.1002/cav.1543>
16. Li, Y., et al.: Surface embedding narrow volume reconstruction from unorganized points. *Comput. Vis. Image Understand.* **121**, 100–107 (2014). <https://doi.org/10.1016/j.cviu.2014.02.002>
17. Luo, J., Xu, S., Jiang, S.: A novel hybrid rendering approach to soft tissue cutting in surgical simulation. In: *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)*, pp. 270–274, October 2015. <https://doi.org/10.1109/BMEI.2015.7401514>

18. Luo, R., Xu, W., Wang, H., Zhou, K., Yang, Y.: Physics-based quadratic deformation using elastic weighting. *IEEE Trans. Visual. Comput. Graph.* **24**(12), 3188–3199 (2018). <https://doi.org/10.1109/TVCG.2017.2783335>
19. Magnoux, V., Ozell, B.: Real-time visual and physical cutting of a meshless model deformed on a background grid. *Comput. Animat. Virt. Worlds*, e1929 (2014). <https://doi.org/10.1002/cav.1929>
20. Malukhin, K., Ehmann, K.: Mathematical modeling and virtual reality simulation of surgical tool interactions with soft tissue: a review and prospective. *J. Eng. Sci. Med. Diagnost. Therapy* **1**(2), 020802 (2018). <https://doi.org/10.1115/1.4039417>
21. Mor, A.B., Kanade, T.: Modifying soft tissue models: progressive cutting with minimal new element creation. In: Delp, S.L., DiGoia, A.M., Jaramaz, B. (eds.) *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2000*, pp. 598–607. *Lecture Notes in Computer Science*, Springer, Heidelberg (Oct 2000). [https://doi.org/10.1007/978-3-540-40899-4\\_61](https://doi.org/10.1007/978-3-540-40899-4_61)
22. Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., Alexa, M.: Point based animation of elastic, plastic and melting objects. In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 141–151. SCA 2004, Eurographics Association, Aire-la-Ville, Switzerland (2004). <https://doi.org/10.1145/1028523.1028542>
23. Nakao, M., Minato, K., Kume, N., Mori, S.i., Tomita, S.: Vertex-preserving cutting of elastic objects. In: *2008 IEEE Virtual Reality Conference*, pp. 277–278, March 2008. <https://doi.org/10.1109/VR.2008.4480799>
24. Nienhuys, H.W.: Cutting in deformable objects. *Utrecht University Repository* (2003). <http://dspace.library.uu.nl/handle/1874/882>
25. Pan, J., Yan, S., Qin, H., Hao, A.: Real-time dissection of organs via hybrid coupling of geometric metaballs and physics-centric mesh-free method. *Vis. Comput.* **34**(1), 105–116 (2016). <https://doi.org/10.1007/s00371-016-1317-x>
26. Peng, Y., Li, Q., Yan, Y., Wang, Q.: Real-time deformation and cutting simulation of cornea using point based method. *Multimedia Tools Appl.* **78**(2), 2251–2268 (2018). <https://doi.org/10.1007/s11042-018-6343-4>
27. Provot, X.: Collision and self-collision handling in cloth model dedicated to design garments. In: Thalmann, D., van de Panne, M. (eds.) *Computer Animation and Simulation 1997*. pp. 177–189. Eurographics, Springer, Vienna (1997). [https://doi.org/10.1007/978-3-7091-6874-5\\_13](https://doi.org/10.1007/978-3-7091-6874-5_13)
28. Qian, K., Jiang, T., Wang, M., Yang, X., Zhang, J.: Energized soft tissue dissection in surgery simulation. *Comput. Animat. Virt. Worlds* **27**(3–4), 280–289 (2016). <https://doi.org/10.1002/cav.1691>
29. Seiler, M., Steinemann, D., Spillmann, J., Harders, M.: Robust interactive cutting based on an adaptive octree simulation mesh. *The Visual Computer* **27**(6–8), 519–529 (2011). <https://doi.org/10.1007/s00371-011-0561-3>
30. Shi, W., Liu, P.X., Zheng, M.: Cutting procedures with improved visual effects and haptic interaction for surgical simulation systems. *Comput. Methods Prog. Biomed.* **184**, 105270 (2020). <https://doi.org/10.1016/j.cmpb.2019.105270>
31. Si, W., Lu, J., Liao, X., Wang, Q.: Towards interactive progressive cutting of deformable bodies via phyxel-associated surface mesh approach for virtual surgery. *IEEE Access* **6**, 32286–32299 (2018). <https://doi.org/10.1109/ACCESS.2018.2845901>
32. Steinemann, D., Otaduy, M.A., Gross, M.: Splitting meshless deforming objects with explicit surface tracking. *Graph. Models* **71**(6), 209–220 (2009). <https://doi.org/10.1016/j.gmod.2008.12.004>



33. Tai, Y., et al.: Development of haptic-enabled virtual reality simulator for video-assisted thoracoscopic right upper lobectomy. In: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3010–3015. IEEE, Miyazaki, Japan, October 2018. <https://doi.org/10.1109/SMC.2018.00511>
34. Wu, J., Westermann, R., Dick, C.: Real-time haptic cutting of high-resolution soft tissues. In: Studies in Health Technology and Informatics (Proceedings of the Medicine Meets Virtual Reality 2014) vol. 196, pp. 469–475 (2014). <https://doi.org/10.3233/978-1-61499-375-9-469>
35. Wu, J., Westermann, R., Dick, C.: A survey of physically based simulation of cuts in deformable bodies. In: Computer Graphics Forum, vol. 34, pp. 161–187. Wiley Online Library (2015). <http://onlinelibrary.wiley.com/doi/10.1111/cgf.12528/full>
36. Zerbato, D., Fiorini, P.: A unified representation to interact with simulated deformable objects in virtual environments. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 2710–2717, May 2016. <https://doi.org/10.1109/ICRA.2016.7487432>
37. Zou, Y., Liu, P.X., Wu, D., Yang, X., Xu, S.: Point primitives based virtual surgery system. IEEE Access **7**, 46306–46316 (2019). <https://doi.org/10.1109/ACCESS.2019.2909061>