Physically-based Simulation of Cuts in Deformable Bodies: A Survey

Jun Wu[†], Rüdiger Westermann[‡], Christian Dick[§]
Computer Graphics & Visualization Group, Technische Universität München, Germany

Abstract

Virtual cutting of deformable bodies has been an important and active research topic in physically-based simulation for more than a decade. A particular challenge in virtual cutting is the robust and efficient incorporation of cuts into an accurate computational model that is used for the simulation of the deformable body.
This report presents a coherent summary of the state-of-the-art in virtual cutting of deformable bodies, focusing on the distinct geometrical and topological representations of the deformable body, as well as the specific numerical discretizations of the governing equations of motion. In particular, we discuss virtual cutting based on tetrahedral, hexahedral, and polyhedral meshes, in combination with standard, polyhedral, composite, and extended finite element discretizations. A separate section is devoted to meshfree methods. The report is complemented with an application study to assess the performance of virtual cutting simulators.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Applications—

1. Introduction

Physically-based, yet efficient and robust simulation of cutting of deformable bodies (also referred to as virtual cutting) has been an important and active research topic in the computer graphics community for more than a decade. It is at the core of virtual surgery simulators, and it is also frequently used in computer animation. A survey of early cutting techniques has been given 10 years ago by Bruyns et al. [BSM*02], and since then a number of significant improvements with respect to physical accuracy, robustness, and speed have been proposed. Our intention in the current state-of-the-art report is to review the basic concepts and principles underlying these techniques.

Virtual cutting involves three major tasks: First, the incorporation of cuts into the computational model of the deformable body, i.e., the update of the geometrical and topological representation of the simulation domain as well as the numerical discretization of the governing equations. Second, the simulation of the deformable body based on this computational model. Third, the detection and handling of collisions. Since techniques for collision detection in virtual cutting are not different to those used in deformable body simulation, their review will not be in the scope of this report. However, for a good overview of the state-of-the-art in this field, including many technical and implementation-specific details, let us refer to the survey by Teschner et al. [TKH*05] and more recent works that consider cutting (e.g., [TMY*11, WDW13]). Physically-based collision handling in virtual cutting applications, on the other hand, is still an open research question, requiring to consider different material properties to predict tissue responses, friction and sliding contacts, as well as accurate force transmission. For a good introduction to the specific problems that have to be addressed to resolve collisions between insertion tools and deformable bodies let us refer to the work by Chentanez et al. [CAR*09].

This report presents a coherent summary of the state-of-the-art in virtual cutting of deformable bodies, focusing on the distinct geometrical and topological representations and the numerical discretizations that have been proposed. The report discusses the different approaches with respect to

- (physical) accuracy, referring to the ability to represent arbitrarily-shaped cuts both in the geometrical and topo-

† jun.wu@tum.de
‡ westermann@tum.de
§ dick@tum.de

DOI: 10.2312/egst.20141033

logical representation as well as in the numerical discretization, and to use physically-based simulation to predict the behavior of the cut object;
- robustness, relating to the numerical stability of the involved algorithms in complicated cutting scenarios, such as thin slicing or repeated cutting at the same location; and
- computational efficiency, which is particularly important in real-time applications such as surgery training and planning, where the update of the computational model as well as the deformation computation must be performed within a very limited time budget.

The techniques we discuss in this report are also employed in fracture simulations. While cutting is the controlled separation of a physical object as a result of an acutely directed force, exerted through sharp-edged tools, fracturing refers to the cracking or breaking of hard objects, under the action of stress. Fracture simulations build on a fracture model, which determines when and where a crack appears, as well as how the crack propagates through the model. To actually realize the crack, the geometrical and topological representation of the object as well as the numerical discretization of the governing equations have to be updated accordingly, and the dynamics simulation of the cut body has to be performed. In this report we focus on reviewing techniques for realizing an actual cut, rather than how the position and shape of a cut is determined. For a thorough introduction to fracture simulation let us refer to [OH99, OBH02].

When comparing the individual approaches used for virtual cutting, one of the most apparent classification criteria is the geometrical and topological representation of the simulation domain. In general, this representation is a spatial discretization, as a spatial discretization of the simulation domain—continuously updated according to the introduced cuts—is required for the numerical discretization of the governing equations.

Most approaches are based on a volumetric mesh representation of the object. Early works in the field [OH99, BMG99, CDA00, NFvdS00, MK00, OBH02] mainly employ tetrahedral meshes, which offer a high degree of flexibility considering the modeling of cuts by splitting elements or/and snapping element vertices onto the cutting surfaces. Unfortunately, these procedures are prone to producing ill-shaped elements, which are numerically unstable. Recent works address this issue by using regular or semi-regular meshes consisting of hexahedral elements [JBB*10, DGW11a, WDW11, SSSH11]. Some works also consider the use of polyhedral meshes [WBG07, MKB*08]. In addition to mesh-based approaches, meshfree approaches based on particles [MKN*04, PKA*05, SOG06, PGCS09] were proposed.

In order to obtain a physically accurate simulation of the deformable body, the large majority of mesh-based approaches employ the finite element method for the numerical discretization of the governing equations. The straightforward approach is to maintain a 1:1 correspondence between computational elements (finite elements) and geometrical elements (cells) of the underlying mesh. The numerical simulation then is mathematically identical to the simulation of an object without cuts. In particular for interactive applications, however, it is highly desirable to decouple the spatial discretization used for the geometrical and topological modeling of cuts from the spatial discretization employed in the numerical simulation, in order to thoroughly balance speed and accuracy. Approaches that are based on this principle are the extended finite element method [JK09, KMB*09] and the composite finite element method [JBB*10, WDW11].

Using implicit time integration schemes, the numerical discretization leads to a large, sparse linear system of equations in each simulation time step. This system can be solved by using standard black box solvers, such as a conjugate gradient solver. A significantly higher computational efficiency can be achieved by means of problem-specific geometric multigrid solvers [GW06], when these solvers are particularly designed for the efficient treatment of the material discontinuities arising in the context of virtual cutting [DGW11a, WDW11].

The remainder of this report is organized as follows: The different mesh representations and the respective adaptation strategies used in virtual cutting are discussed in Section 2. Finite element methods and numerical solvers are discussed in Section 3. Meshfree approaches are reviewed in Section 4. A summary of the surveyed techniques and representative simulation scenarios are presented in Section 5. To demonstrate the performance that can be achieved for virtual cutting on desktop PC hardware, we have performed an application study. The results of this study are presented in Section 6. The report is concluded in Section 7 with a discussion of future research challenges.

2. Mesh-based Modeling of Cuts

Virtual cutting of a deformable body is modeled by manipulating the geometrical and topological representation of the simulation domain. In this section, after briefly discussing the modeling of the cutting process, we focus on mesh-based representations, including tetrahedral, hexahedral, and polyhedral meshes, and we discuss the adaptation of these meshes to cuts.

For rendering and collision handling, a surface representation of the object is required. This representation can be directly obtained from a tetrahedral or polyhedral mesh by determining the element faces lying on the surface. For hexahedral meshes, however, a separate surface representation is mandatory to compensate the jagged simulation domain boundary (staircases) resulting from the hexahedral discretization. To this end, cube-based or dual contouring algorithms that reconstruct a smooth surface from the hexa-

hedral mesh were proposed. Sifakis et al. [SDF07] demonstrated how a lower-resolution tetrahedral mesh representing the simulation domain can be combined with a set of given high-resolution surface meshes (original object surfaces and cutting surfaces) for rendering and collision handling.

2.1. Modeling of the Cutting Process

The cutting process is modeled in simulation practice by detecting intersections between the volumetric mesh that represents the deformable object, and a triangulated surface mesh that represents a cutting surface. The cutting surface is generated from the movement of the cutting tool (scalpel). Specifically, element edges, or links between face-adjacent elements are tested against the cutting surface mesh [BMG99, NFvdS00, WDW13]. To generate sub-mesh cutting effects such as in polyhedral modeling, element faces are also tested against the cutting mesh [WBG07]. Based on these intersections, elements are split and detached accordingly, as described in the following sections. Since cutting only happens locally, a large region of the deformable object can be pruned before elementary intersection tests are performed using bounding volume hierarchies. In progressive cutting, a breadth-first traversal of the volumetric mesh starting from previous intersection points is also useful.

The cutting surface normally is the surface swept by the scalpel's cutting edge between two successive simulation frames. Together with 3D spatial interfaces such as a haptic device, this approach enables a natural interaction with the virtual environment. The scalpel may have a complex geometry for visual rendering, comprising a set of triangles. For simplicity, however, the blade that actually cuts the object is usually represented by a single line segment. For non-interactive applications, the cutting surface can also be predefined in the reference configuration [MBF04, KMB*09]. For example, the cutting surface can be constructed from a contour defined on the surface of the deformable object, which is similar to the guide contours defined during pre-operative surgery planning [WBWD12]. This allows for full control over the cutting surface, simplifies the intersection tests, and avoids the possible problems with temporally discrete intersection testing.

Compared to the physical world, where cuts are induced by the internal stresses resulting from the force interaction between the deformable object and the scalpel, the described approach so far is purely geometry-based. It can be interpreted as modeling an infinitely sharp scalpel, which can induce arbitrary stresses and thus immediately penetrates the object. In contrast, in the physical world, the object would deform under the influence of the increasing force exerted by the scalpel, before the scalpel eventually penetrates. Modeling a more realistic interaction between the object and the scalpel is part of ongoing research, for example in biomedical engineering [CDL07]. Using penalty-force-based collision handling, an initial attempt to simulate this effect is to employ a virtually extended scalpel shape [JBB*10]. The enlarged scalpel penetrates into the deformable object before the real scalpel penetrates, thus leading to a deformation before the object is cut. The enlargement of virtual tools is similarly used in bone drilling simulation [WWWZ10]. Another open problem is how time-continuous intersection testing between the deformable body and the cutting tool can be realized. In current approaches, the deformable body and the scalpel are moved sequentially within each simulation frame, rather than simultaneously. As a consequence, edges/links might be missed by the cutting tool, especially if the object is moving rapidly.

2.2. Tetrahedral Meshes

After a brief review of some of the approaches for generating an initial tetrahedral mesh, we introduce and discuss the following techniques for the incorporation of cuts into tetrahedral meshes (see Figure 1 for a 2D illustration):

- Element deletion [CDA00],
- Splitting along existing element faces [NFvdS00, MG04, LT07],
- Element duplication [MBF04, SDF07],
- Snapping of vertices [NFvdS01, LJD07],
- Element refinement [BMG99, BG00, MK00, BS01, BGTG04, GCMS00],
- Combined snapping of vertices and element refinement [SHGS06].

One challenge is the accurate representation of arbitrarily-shaped cuts, while avoiding the creation of ill-shaped elements [She02], which lead to numerical instabilities during mesh adaptation and deformation computation. The method of element deletion and the method of splitting along existing element faces maintain the well-shaped elements of the original discretization, but they result in jagged surfaces. By means of snapping of vertices or element refinement, or a combination of both, cuts can be accurately represented. However, since the elements are modified, for these methods it is necessary to prevent ill-shaped elements. The method of element duplication provides a good trade-off between accuracy and robustness by embedding an accurate surface into the duplicated elements in their original shapes.

2.2.1. Tetrahedral Mesh Generation

An initial tetrahedral discretization of the simulation domain can be generated from surface meshes [Si06], medical image data [ZBS05], or level sets [TMFB05]. Quality tetrahedral mesh generation itself remains an active research topic. It is well known that ill-shaped elements (e.g., needle elements, or almost planar sliver elements) lead to numerical instabilities [She02].

2.2.2. Cut Modeling without Creating New Elements

Perhaps the easiest way to incorporate cuts into the deformable body is to separate the material by removing el-

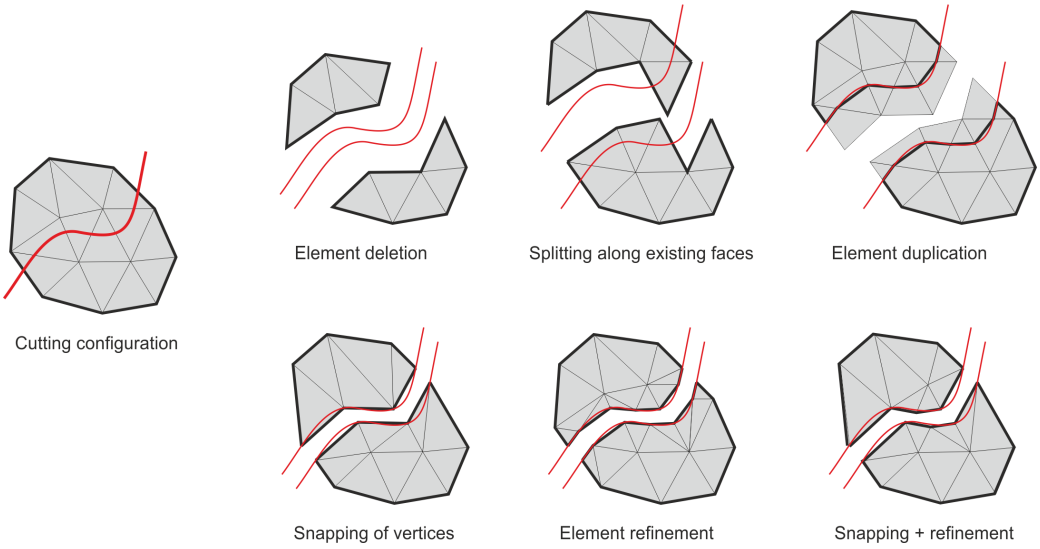

Figure 1: Illustration of different methods for incorporating cuts into a tetrahedral mesh (a triangle mesh in 2D). The red cutting path separates the object into two disconnected parts, which are illustratively displaced to make the discontinuity visible. The surface of the object (bold black line) is given by the set of surface faces of the tetrahedral elements, except for the approach that is based on element duplication, where a separate surface mesh is maintained.

ements that are touched by a cutting tool. While this simple method is widely adopted in real-time simulations (e.g., [CDA00]), it puts severe limitations on the mechanical accuracy and visual quality. First, the newly exposed surface does not conform to the smooth swept surface of a cutting tool, but to the initial discretization of the deformable body, leading to a rather jagged surface. Second, the removal of elements causes a loss of volume, and it leaves unrealistic holes in the object. A remedy to the second problem is to split the object along existing element faces [NFvdS00]. This works fine if the cutting surfaces are known a priori to creating the initial discretization [LT07], i.e., the tetrahedralization takes the pre-recorded cutting surface into account. However, for arbitrary cuts, it still results in a jagged surface. To make the newly created surface conforming to cuts, a simple method is to snap the vertices onto the cutting surface before splitting the object along element faces [NFvdS01]. This modification, however, may create ill-shaped elements, which need further treatment afterwards.

2.2.3. Cut Modeling by Element Refinement

To accurately accommodate complex cuts with a reasonable number of initial elements, it is thus necessary to locally refine tetrahedra. Bielser et al. presented a 1:17 subdivision method for tetrahedral decomposition, by generating a vertex on each edge, and a vertex on each triangle face [BMG99]. The exact placement of these vertices depends on the intersection between the cutting tool and the element. Initially, adjacent elements share their vertices. Cutting is modeled by duplicating vertices appropriately. Figure 2 (left) illustrates the five topologically different configurations of a tetrahedron after introduction of a cut. Among these five configurations, IIIa and IV correspond to complete cuts through the tetrahedron, while the other three correspond to partial cuts. For each of these configurations, the information which vertices have to be duplicated in order to generate the respective topological configuration after performing the 1:17 subdivision, is pre-computed and stored in a look-up table.

To reduce the number of elements compared to a full 1:17 subdivision, Bielser and Gross subdivided only those edges and faces which are part of the cutting surface [BG00]. Mor and Kanade presented a method for progressive cutting that minimizes the number of newly created elements [MK00], and Ganovelli et al. proposed a multi-resolution approach to reduce the number of elements [GCMS00]. Bielser et al. further proposed a state machine to track the topological configuration of each tetrahedron during progressive cutting [BGTG04].

Considering the decomposition of tetrahedron, if the intersection between an edge and the cutting surface is very close to one of the edge's vertices, ill-shaped elements will occur. Steinemann et al. proposed a combination of snapping of vertices and element refinement to solve this problem [SHGS06]. The idea is illustrated in Figure 3 for the 2D case. If a vertex of an intersected edge lies close to the cutting surface (the distance is smaller than a given threshold), the algorithm moves this vertex onto the cutting surface, and separates the material by duplicating the vertex. If the cut-

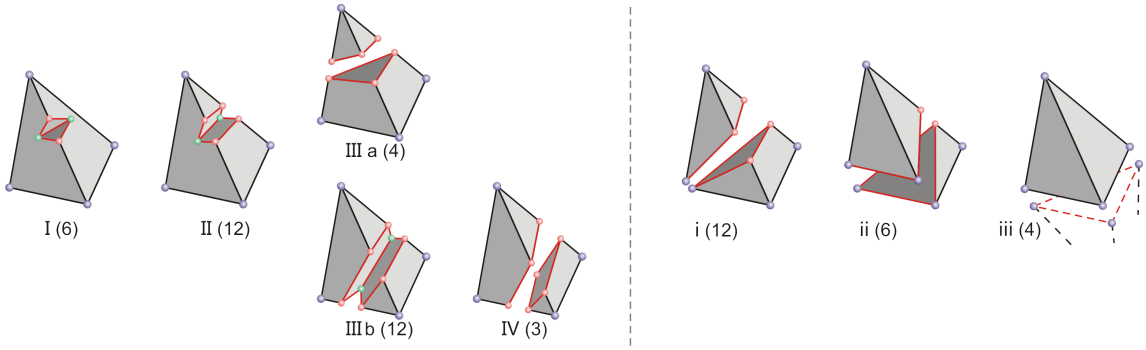

Figure 2: Left: A cut tetrahedron can have five topologically different configurations. The Roman numeral represents the number of disconnected edges. The number in parentheses indicates the number of topologically equivalent configurations by rotation and mirroring operations. Right: In the hybrid cutting approach, three additional topological configurations of a cut tetrahedron are introduced. The small Roman numeral represents the number of existing vertices which are snapped and duplicated.

ting surface intersects an edge close to its midpoint, the edge is split. The method is implemented by extending the set of five topological configurations of a cut tetrahedron, shown in Figure 2 (left), by three additional topological configurations, illustrated in Figure 2 (right). The additional configurations correspond to a complete cut that passes through one, two, or three vertices.

To model a curved cut within a tetrahedral element, given by a sequence of cutting surface triangles, the individual triangles in principle can be successively incorporated into the tetrahedral mesh, leading to a sequence of repeated tetrahedral splits. Since this approach leads to a very large number of tetrahedra along the cut, in practice only a single split of the initial tetrahedron is performed. A curved cut thus is approximated by a only a few tetrahedron faces. The resulting sub-tetrahedra in general are only split if they are intersected by another cut. Also for progressive cutting, the initial tetrahedron is split only once, i.e., when a partial cut is further progressing through a tetrahedron, the current tetrahedral split is undone and replaced by a new split.

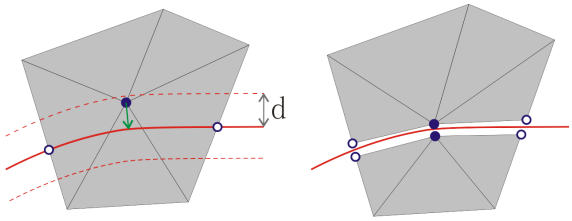

Figure 3: A hybrid cutting approach based on both snapping of vertices and element refinement. If the intersection between an edge and the cutting surface is close to one of the edge's vertices (determined by a threshold d), the vertex is moved onto the cutting surface, in order to prevent the creation of ill-shaped elements. Otherwise the edge is split at the exact intersection point.

2.2.4. Cut Modeling by Element Duplication

Molino et al. proposed the virtual node algorithm to circumvent subsequent numerical problems resulting from ill-shaped elements [MBF04]. The basic idea is to create one or more replicas of the elements that are cut, and to embed each distinct material connectivity component of an element into a unique replica. The replicas comprise both original vertices inside the material (referred to as real nodes) and newly created vertices outside the material (referred to as virtual nodes). Embedding means that the deformation computation is performed on the well-shaped replicas of the original element, and then the displacements of the element's fragments are determined by means of interpolation.

In the initial version of the algorithm, each replica is required to have at least one real node. This was extended by Sifakis et al. to allow for replicas with purely virtual nodes, and thus to support an arbitrary number of fragments within a single tetrahedron [SDF07]. Given a set of triangle surface meshes (original object surfaces and cutting surfaces), enclosed by a tetrahedral mesh that covers the simulation domain, the algorithm first generates a set of non-intersecting polygons from the triangle soup consisting of surface mesh triangles and tetrahedron faces. From these polygons, a polyhedral discretization is determined by examining the connectivity among the polygons. Note that the polyhedra and the tetrahedra per construction do not intersect. Then, for each tetrahedron, the material connectivity components are determined from the polyhedral discretization. For each connectivity component, a duplicate of the tetrahedron is created.

In this way, the algorithm enables to combine a lower-resolution tetrahedral mesh for the representation of the simulation domain with high-resolution surface meshes for rendering and collision handling. Note that by means of the du-

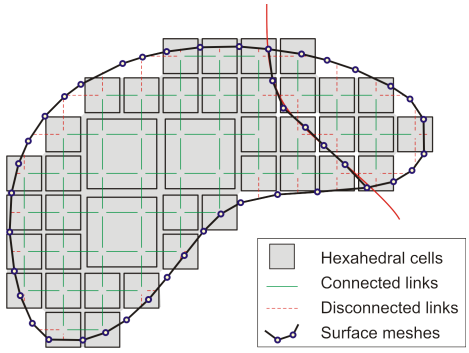

Figure 4: 2D illustration of the modeling of cuts in a linked volume representation. The object is discretized by means of an adaptive octree grid (shaded cells). The cells of this grid are connected by links (green, solid). Cutting is modeled by disconnecting links (red, dashed). A surface mesh (black line and dots) is reconstructed from the dual grid of links.

plication of elements, the volumetric representation and the surface representation are topologically consistent.

2.3. Hexahedral Meshes

A regular or semi-regular hexahedral discretization, generated directly from medical image data [ZBS05] or from polygonal surface meshes by voxelization techniques [ED08, DGBW08], provides an effective means to represent cuts without having to worry about ill-shaped elements [JBB*10, SSSH11]. We discuss the approach of using a linked volume representation, where the connectivity is modeled by links between face-adjacent elements [FG99, DGW11a], and review surface reconstruction techniques to build a smooth surface mesh from the hexahedral grid [WDW11].

2.3.1. Volume Representation

To model cuts in the deformable body, Frisken-Gibson proposed a linked volume representation [FG99]. The basic idea of the linked volume representation is to decompose the object into a set of hexahedral elements, using a uniform hexahedral grid. Face-adjacent elements are connected via links, with six links emanating from each element. Cuts are modeled by marking links as disconnected when they are intersected by the virtual cutting blade. Cuts are thus represented at the resolution of the hexahedral grid.

Since the resolution of a uniform grid is in practice limited by simulation time and memory requirements, an adaptive octree grid for virtual cutting was proposed by Dick et al. [DGW11a] (see Figure 4), which adaptively refines along cuts, down to a certain finest level. Links are still considered on the uniform grid corresponding to this finest level, but are physically stored only for the elements at the finest level.

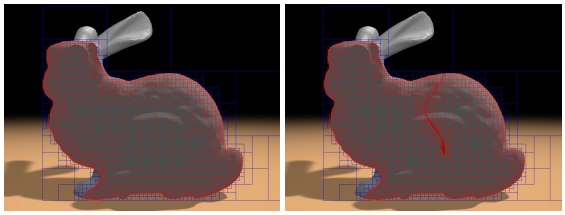

Figure 5: The Stanford bunny model is discretized into a linked octree grid (left), which is refined along the surface and the cuts (right).

The adaptive octree grid is constructed by starting from a coarse uniform grid. Whenever a link on the finest level is intersected by the surface of the deformable object, the incident elements (possibly only one element, when both endpoints of the link are lying within the same element) are refined using a regular 1:8 split. At the finest level, links are marked as disconnected when they are intersected by the object's surface. Elements that are lying outside of the object are removed from the representation. To avoid jumps in the discretization, additional splits are performed to ensure that the level difference between elements sharing a vertex, an edge, or a face is at most one (restricted octree). Cuts are modeled analogously to the modeling of the object surface, i.e., elements are adaptively refined along a cut down to the finest level, where links are marked as disconnected (see Figure 5 for an example). Material properties such as Young's modulus and density are assigned on a per-element basis. To model inhomogeneous materials, the octree mesh can be refined further.

2.3.2. Surface Representation

To render the surface of the deformable object—including the additional surface parts that are generated by cutting—a surface mesh is reconstructed from the volume representation. Wu et al. [WDW11] applied the dual contouring approach [JLSW02] for constructing this surface. Compared to the splitting cubes algorithm [PGCS09], which was used in [DGW11a], dual contouring improves the quality of the generated mesh and reduces the total number of triangles. Dual contouring operates on the (imaginary) grid that is formed by the links between the elements at the finest level. For each link that is cut by the blade, the distances between the intersection point and the link's endpoints as well as the normal of the blade at the intersection point are stored. This information is used to position a surface vertex within each cell that is incident to at least one disconnected link. Since for a cut two surfaces have to be created—one for each material side—this vertex is duplicated, so that for each material component in the cell one vertex exists. The material components in a cell are determined by means of a look-up table, which is indexed by the pattern of connected and disconnected links incident to a cell. After generating the vertices,

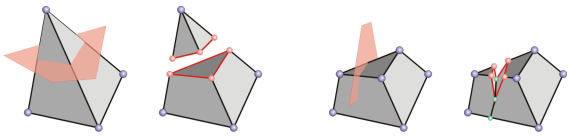

Figure 6: Illustration of cuts in polyhedral elements. Left: A tetrahedron is cut into two parts, resulting in a small tetrahedron and a triangular prism. Right: The triangular prism is partially cut, resulting in two polyhedral elements that are partially connected. Contrary to a tetrahedral discretization, no further subdivision is required.

the surface is spanned by creating two surface patches (2×2 triangles) for each link that is cut. The vertices are finally bound to the nearest element of the respective material part. This binding allows for carrying over the deformation computed at the vertices of the hexahedral simulation mesh to the surface vertices.

2.4. Polyhedral Meshes

When cuts are modeled by element refinement, the resulting elements necessarily must be tetrahedra or hexahedra, when a tetrahedral or hexahedral discretization is used. A polyhedral discretization removes this constraint by allowing the creation of general polyhedra. This potentially enables the modeling of cuts by creating a smaller number of new elements [WBG07, MKB*08].

Without loss of generality, polyhedral modeling of cuts can be realized by starting with a purely tetrahedral discretization of the object. To model a complete cut, upon which an element is split into disconnected parts, two new convex elements are created. These resulting elements are composed of vertices of the initial element and the intersections between its edges and the cutting polygon. As illustrated in Figure 6 (left), a tetrahedron is split into a small tetrahedron and a triangular prism. Note that no re-meshing is needed to decompose the triangular prism into smaller tetrahedra. Figure 6 (right) shows the modeling of a partial cut. The intersections between the element's edges and the cutting polygon, and between the element's faces and the cutting polygon's edges are employed as new vertices.

While polyhedral modeling of cuts potentially leads to simplified operations, similar to tetrahedral meshes, there are practical issues with respect to ill-shaped elements. A fundamental problem is that quality criteria of general polyhedral elements are unclear. Wicke et al. found that in particular sliver polyhedra, which are almost planar, lead to numerical problems during simulation, and applied vertex merging and snapping to remove these slivers [WBG07]. Furthermore, to avoid possible numerical problems, it is required to enforce that the elements are convex. These issues and constraints make quality and efficient polyhedralization non-trivial.

2.5. Discussion on Discretizations

Avoiding ill-shaped elements is still a major challenge when using a tetrahedral discretization, especially under the constraint of the limited time budget in real-time applications. While a polyhedral discretization offers more flexibility with respect to the shape of individual elements, still special care is required to avoid ill-shaped polyhedra, and also to ensure the convexity of the elements. The virtual node algorithm is superior in this aspect, since it embeds possibly ill-shaped fragments into duplicates of original elements, whose quality can be ensured during preprocessing.

Using a semi-regular hexahedral discretization is an effective means to ensure that the elements are well-shaped during dynamic mesh refinement. However, a separate surface representation is required to compensate the jagged nature of the hexahedral grid.

3. Finite Element Simulation for Virtual Cutting

In mesh-based cutting approaches, the finite element method is typically used for the numerical discretization of the governing equations of elasticity. The standard approach is to directly employ the spatial discretization that is induced by the mesh, i.e., to create one computational element (finite element) for each geometrical element (cell). The deformable body simulation then is identical to the case without cuts. General simulation of deformable bodies is for example surveyed in [NMK*06]. A detailed explanation of finite element procedures for mechanics can be found in engineering textbooks (e.g., [Bat96]), while a concise introduction of FEM in medical simulation is given in [BN98].

In this section, we discuss three finite element methods that are specialized for simulating cuts in deformable bodies. In particular, we discuss the extended finite element method, the composite finite element method, as well as the polyhedral finite element method. For the first two methods, separate spatial discretizations are employed for the representation of the simulation domain and for the numerical simulation. This allows for the modeling of complicated-shaped cuts (and also a complicated-shaped original surface of the object), while requiring only a rather small number of computational elements. In this way, these methods enable to carefully balance speed and accuracy, which is particularly important for interactive applications.

For each method, we present its idea and its main components (e.g., the design of shape functions and the construction of element stiffness matrices). In an additional section, we also briefly review the numerical methods for solving the system of equations resulting from finite element discretization and implicit time integration, since the numerical solver is a crucial component considering the overall performance of a cutting application.

3.1. The Extended Finite Element Method

The basic idea of the extended finite element method (XFEM) [BB99] is to model material discontinuities introduced by cuts by adapting the basis functions of the finite dimensional solution spaces [BM97, SCB01]. XFEM was originally invented to accurately simulate material interfaces and crack propagation [MDB99, SMMB00]. The idea was recently utilized for cutting and fracturing deformable objects in graphics applications [LT07, JK09, KMB*09].

In the standard FEM, the displacement within an element is interpolated from the displacements at the element's nodes by using *continuous* shape functions, which are apparently not sufficient to model the discontinuities introduced by cuts. The idea of XFEM is to introduce *discontinuous* enrichment functions, together with additional degrees of freedom (DOFs) assigned to the original nodes. The displacement field $u(x)$ is computed as

$$u(x) = \Phi^e(x)\, u^e + \underbrace{\Psi^e(x)\Phi^e(x)\, a^e}_{enrichment}, \tag{1}$$

where $\Phi^e(x)$ is the standard shape matrix, u^e is the vector of original DOFs, $\Psi(x)^e$ is the element's shape enrichment matrix, which is composed of discontinuous enrichment functions $\psi_i^e(x)$, and a^e represents the newly assigned DOFs.

To make the shape functions fulfill the Kronecker delta property, a good choice for the enrichment functions is the shifted Heaviside function, i.e.,

$$\psi_i^e(x) = \frac{H(x) - H(x_i)}{2}, \tag{2}$$

where x_i is the position of the i-th node, and $H(x)$ is the generalized Heaviside function (also known as the sign function)

$$H(x) = \begin{cases} +1 & \text{if } x \text{ is on the cut's left side;} \\ -1 & \text{if } x \text{ is on the cut's right side.} \end{cases} \tag{3}$$

As illustrated in Figure 7 (for simplicity for a planar element), using the shifted Heaviside function ensures the discontinuity across the cut. Substituting the enrichment functions Eq. 2 into Eq. 1, in this example the displacement field becomes $u(x) = \Phi^e(x)(u_1,\, u_2 + a_2,\, u_3)^{\mathrm{T}}$ on the left side of the cut, and $u(x) = \Phi^e(x)\,(u_1 - a_1,\, u_2,\, u_3 - a_3)^{\mathrm{T}}$ on the right side. Employing the shifted Heaviside function as enrichment functions makes it easy to treat boundary conditions: Since they vanish at the nodes, i.e., $\psi_i^e(x_i) = 0$, the displacement at the position of the i-th node is independent of the additional DOFs a^e. It should be noted that a different selection of the enrichment functions influences the physical meaning of the original and the added DOFs, but leads to the same displacement field.

With the enriched shape functions defined, the enriched element stiffness matrix is computed as

$${}^{\mathrm{x}}K^e = \int_{\Omega^e} ({}^{\mathrm{x}}B^e)^{\mathrm{T}} C ({}^{\mathrm{x}}B^e)\mathrm{d}x, \tag{4}$$

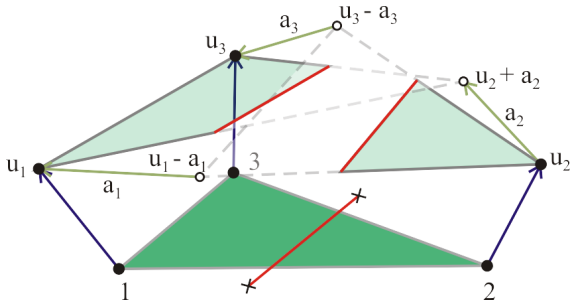

Figure 7: A discontinuous displacement field computed with the extended finite element method. The green triangle domain is divided by a red cut line. The displacements u_i and a_i correspond to the original and added DOFs respectively. Using the shifted Heaviside function as enrichment functions, the displacement field is $u(x) = \Phi^e(x)(u_1,\, u_2 + a_2,\, u_3)^{\mathrm{T}}$ on the left side of the cut, and $u(x) = \Phi^e(x)\,(u_1 - a_1,\, u_2,\, u_3 - a_3)^{\mathrm{T}}$ on the right side.

where C represents the material law, and the enriched element strain matrix is composed according to

$${}^{\mathrm{x}}B^e = \left(B_1^e,\, \ldots,\, B_{n_v}^e,\, \psi_1^e B_1^e,\, \ldots,\, \psi_{n_v}^e B_{n_v}^e\right). \tag{5}$$

The enriched element stiffness matrix has the form

$${}^{\mathrm{x}}K^e = \begin{pmatrix} K^{e,uu} & K^{e,ua} \\ K^{e,au} & K^{e,aa} \end{pmatrix}, \tag{6}$$

where the superscripts ^{u} and ^{a} correspond to the original and the added DOFs, respectively. Details that facilitate implementation, as well as enriched element stiffness matrices for the corotational and non-linear strain formulations were derived by Jeřábková et al [JK09].

While only a single cut is considered above, multiple cuts, in principle, can be supported by further adding more enrichment functions and simulation DOFs. Kaufmann et al. proposed enrichment textures for detailed cutting of shells [KMB*09]. They proposed a harmonic enrichment approach, which uses only one unified kind of enrichment functions to handle multiple, partial, progressive, and complete cuts. While this approach is in general applicable to 3D solids, such a generalization has not been reported yet.

3.2. The Composite Finite Element Method

The idea of the composite finite element method (CFEM) [HS97, SW06] is to approximate a high-resolution finite element discretization of a partial differential equation by means of a smaller set of coarser elements. Preusser et al. used the composite finite element method to resolve complicated simulation domains with only a few degrees of freedom, and also to improve the convergence of geometric multigrid methods by an effective representation of complicated object boundaries at ever coarser scales [PRS07, LPR*09]. In computer graphics, Nesme et al. em-

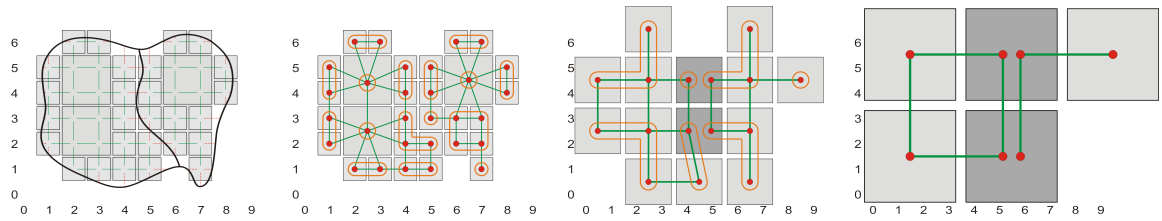

Figure 8: Hierarchical construction of the composite finite element model. Left: 2D illustration of the adaptive linked volume representation, consisting of a set of octree cells (gray), which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary and along the newly generated surfaces due to cutting (bold black line). Middle left to right: Iterative coarsening of the finite element model. The underlying graph representation is indicated by red vertices and green edges. In each block of 2^3 cells of the underlying lattice, the connected components (orange) are determined, and the elements of each connected component are replaced by a separate composite finite element. Duplicated elements are shaded in dark gray.

ployed CFEM as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation [NKJF09].

Recently composite finite elements were leveraged in the context of virtual cutting to reduce the number of simulation DOFs [JBB*10, WDW11]. The adoption of CFEM for cutting simulation is motivated by the following facts. First, using hexahedral discretizations (see Section 2.3), an accurate representation of complex cuts typically requires a high-resolution octree grid. Creating a hexahedral simulation element for each octree cell would lead to a very high number of DOFs, exceeding the number of DOFs that can be simulated in real-time. Second, due to its simplicity, the regular or semi-regular hexahedral grid enables an efficient construction of composite finite elements.

A composite finite element is obtained by combining a set of small standard finite elements into a single larger element. In particular, the shape functions of the composite finite element are assembled from the shape functions of the individual elements. In the following, we discuss the geometrical and topological composition, and the numerical composition of the stiffness matrices.

Geometrical and topological composition Building upon the hexahedral grid, composite elements are constructed by merging the elements in a block of 2^3 cells of the underlying lattice into one composite element. This straightforward approach, however, may merge topologically disconnected elements into one composite element, neglecting the fact of the material's discontinuity. To accurately represent the topology and thus to enable a physically correct simulation, an individual composite element is created for each mechanically disconnected part, as demonstrated in [NKJF09, DGW11a, WDW11]. As a consequence, multiple composite finite elements may exist at the same location. An illustration of this process, which can be iteratively repeated to create ever coarser composite elements, is given in Figure 8.

Numerical composition The stiffness matrices for the composite elements are assembled from those of the underlying standard elements. Using composite finite elements, the DOFs are located at the vertices of these composite elements. The displacements at the vertices of the underlying hexahedral finite elements are determined by trilinear interpolation from the vertices of the composite finite elements. This is described by the equation $u = I\tilde{u}$, where $\tilde{u}$ and u denote the linearizations of the displacements at the vertices of the composite finite elements and the underlying hexahedral finite elements, respectively, and the interpolation matrix I expresses the trilinear interpolation from the composite element vertices.

It can be derived that the stiffness matrix $\tilde{K}$ for the composite finite element discretization has the form

$$\tilde{K} = I^{\mathrm{T}} K I, \tag{7}$$

where K is the stiffness matrix for the underlying hexahedral finite element discretization. $\tilde{K}$ is assembled from the composite element stiffness matrices $\tilde{K}^c$, which are computed as

$$\tilde{K}^c_{mn} = \sum_{e \text{ in } c} \sum_{i=1}^{8} \sum_{j=1}^{8} w^{c\to e}_{m\to i} w^{c\to e}_{n\to j} K^e_{ij}, \quad m,n = 1,\dots 8. \tag{8}$$

Here, the first sum iterates over the hexahedral elements e that are merged into the composite element c. Note that the element matrices are interpreted as 8×8 matrices with each entry being itself a 3×3 matrix. Thus, $\tilde{K}^c_{mn}$ and K^e_{ij} denote 3×3 blocks of scalar entries.

The trilinear interpolation weights $w^{c\to e}_{m\to i}$ from the vertices $m = 1,\dots 8$ of the composite element c to the vertices $i = 1,\dots,8$ of the hexahedral element e are defined as

$$w^{c\to e}_{m\to i} = (1 - \frac{|x^c_m - x^e_j|}{s^c})(1 - \frac{|y^c_m - y^e_j|}{s^c})(1 - \frac{|z^c_m - z^e_j|}{s^c}), \tag{9}$$

where (x^c_m, y^c_m, z^c_m) and (x^e_i, y^e_i, z^e_i) are the coordinates of the vertices, and s^c denotes the edge length of the composite element.

3.3. The Polyhedral Finite Element Method

To avoid the re-meshing process in standard finite elements, Wicke et al. proposed to directly work on more general convex polyhedral elements [WBG07]. Martin et al. extended this method to support arbitrary convex and concave polyhedral elements with planar (not necessarily triangulated) faces [MKB*08]. Kaufmann et al. further applied discrete discontinuous Galerkin FEM to arbitrary polyhedra [KMBG08]. These approaches are collectively named here as polyhedral finite element method (PFEM).

Shape functions for polyhedra A key component in PFEM is valid shape functions defined on the polyhedral domain. They should fulfill the properties of positivity and reproduction of linear polynomials, as required for the convergence of the finite element method [WBG07].

Wicke et al. employed the mean value interpolation function, which is defined as a normalized weight function for each vertex x_i of a convex polyhedron with k vertices according to

$$\phi_i(x) = \frac{w_i(x)}{\sum_{l=1}^{k} w_l(x)}. \tag{10}$$

Enumerating x_i's edge-adjacent vertices by x_j, the weight w_i is defined as a weighted sum of ratios of signed tetrahedra volumes by

$$w_i(x) = \sum_j \left(\frac{c_{j,j+1}}{V_{i,j,j+1}} + \frac{c_{i,j} V_{j-1,j+1,j}}{V_{i,j-1,j} V_{i,j,j+1}} \right), \tag{11}$$

where $V_{a,b,c}$ represents the volume of the tetrahedron spanned by x_a, x_b, x_c and x, and $c_{a,b}$ is computed as

$$c_{a,b}(x) = \frac{\|(x_a - x) \times (x_b - x)\|}{6} \arccos\left(\frac{(x_a - x)^{\mathrm{T}} (x_b - x)}{\|x_a - x\| \|x_b - x\|} \right). \tag{12}$$

Martin el al. used harmonic shape functions as a generalization of linear tetrahedral shape functions to general polyhedral elements. A shape function is called harmonic if its Laplacian vanishes within the element. With its value fully determined at the nodes (constrained by the Kronecker delta property), the harmonic shape function is uniquely determined. Since closed form expressions for harmonic shape functions do not exist for general polyhedra, they numerically computed the solution of the Laplacian equation using the method of fundamental solutions.

Computation of element matrices In contrast to finite element methods based on tetrahedral and hexahedral elements, an analytical evaluation of the element stiffness matrices for polyhedral elements is non-trivial. To efficiently integrate $(B^e)^{\mathrm{T}} C B^e$ over a polyhedral domain, Wicke et al. approximated the integrals using a small set of sample points p heuristically placed throughout the element, in particular, one integration sample p_i per vertex of the element, and one sample p_f per triangle of the element faces. In their implementation, the per-vertex samples are placed between the element centroid c and the vertex x_i, at $p_i = 0.8x_i + 0.2c$, while the per-triangle samples are located between the element centroid c and the face centroid c_f, at $p_f = 0.9c_f + 0.1c$. Their simulation results show that the exact location of the samples has little influence, and that the difference compared to employing around 10,000 samples per element is subtle.

Integrating over this set of sample points, the element stiffness matrix K^e has the form

$$K^e = \sum_i \frac{\mu_i^e}{2} (B^e(p_i))^{\mathrm{T}} C B^e(p_i) + \sum_f \frac{\kappa_f^e}{2} (B^e(p_f))^{\mathrm{T}} C B^e(p_f). \tag{13}$$

Here, μ_i^e and κ_f^e represent the volume fractions associated with the per-vertex integration sample p_i and with the per-triangle sample p_f, respectively. Specifically, enumerating x_i's edge-adjacent vertices by x_j, the volume fraction μ_i^e is defined as

$$\mu_i^e = \frac{\sum_j V(x_i, x_j, x_{j+1}, c)}{3V^e}. \tag{14}$$

For a triangle face with vertices x_{j_1}, x_{j_2}, and x_{j_3}, the volume fraction κ_f^e is defined as

$$\kappa_f^e = \frac{V(x_{j_1}, x_{j_2}, x_{j_3}, c)}{V^e}. \tag{15}$$

3.4. Discussion on Finite Element Methods

Both the extended finite element method and the composite finite element method are based on using distinct spatial discretizations for the representation of the simulation domain and the numerical discretization. In particular, the spatial discretization that is employed for the numerical discretization does not need to be aligned at the simulation domain boundary. Compared to the standard finite element methods, this enables to reduce the number of computational elements/DOFs along the boundary, and thus to balance speed and accuracy. Both approaches are based on using duplicated DOFs at the same location in order to correctly model the topology of the simulation domain. Whereas the extended finite element method directly duplicates the DOFs at the vertices of the original element, the composite finite element method is based on duplicating elements, which implicitly leads to a duplication of DOFs.

It is worth noting that the virtual node algorithm [MBF04, SDF07] described in Section 2.2.4 is related to these approaches, in that it is also based on the duplication of elements and thus DOFs in order to correctly represent the topology of the simulation domain. However, since in the virtual node algorithm the duplicated elements are assigned the (standard) element matrices of the original elements before cutting, the distribution of the material to the distinct sides of a cut is not modeled accurately. This is in contrast to the extended and the composite finite element method, where the material boundaries (including those resulting from cutting) are accurately represented by using special-

ized element matrices, i.e., the construction of these matrices takes the exact material boundaries into account.

3.5. Numerical Solvers

To solve the sparse linear system of equations resulting from finite element discretization and implicit time integration, an efficient numerical solver is required. Here it is worth noting that in the particular application of virtual cutting, the initialization time of the solver plays a significant role. Since the system matrices have to be re-assembled in every simulation step due to the use of the corotational strain formulation and the handling of topology changes, initialization of the solver has to be performed in every simulation step, too.

3.5.1. Direct Solvers

Direct methods solve for an exact solution of the linear system by a finite sequence of operations. Zhang et al. proposed to solve the equation system resulting from static finite element simulation of linear materials by pre-computing the inverse of the global stiffness matrix [ZWP05]. Topological modifications add new rows and columns into this matrix, and its inverse can be re-computed efficiently via the Sherman-Morrison-Woodbury formulae [Hag89]. Lee et al. made use of condensation [BNC96, WH05] to further accelerate the inversion process [LPO10]. By considering only the surface nodes of the volumetric object in the computation of the inverse, significantly improved computation times could be achieved. In a similar manner, Lindblad and Turkiyyah updated the inverse of the stiffness matrix in the extended finite element method [LT07]. Turkiyyah et al. proposed to use progressive updates of the Cholesky factorization to simulate cutting of a 2D mesh [TKAN09], and Courtecuisse et al. updated the inverse of the compliance matrix after cutting to model the contact [CJA*10]. Recently, sparse direct solvers were applied to corotational elastodynamics with consistent topology [HLSO12].

3.5.2. Iterative Solvers

Direct solvers using matrix inversion and factorization do not scale well in the number of DOFs because of their extensive memory and computation requirements. Furthermore, such solvers cannot trade accuracy for speed, which is required in interactive applications to guarantee prescribed response rates. Iterative solvers, on the other hand, provide an effective means to balance speed and accuracy. Iterative solvers use an initial guess to generate a sequence of progressively accurate approximations to the exact solution of a system of equations.

The major concern of iterative solvers is the convergence rate that can be achieved per given time interval. Nienhuys and van der Stappen [NFvdS00, NFvdS01] used a conjugate gradient (CG) solver [She94]. CG solvers can be parallelized efficiently using OpenMP [CAR*09] and CUDA [CJA*10]. Dick et al. proposed a geometric multigrid solver for cutting simulation, based on a hexahedral discretization of the simulation domain [DGW11a]. The main idea of multigrid is to improve the convergence of a basic iterative method by global correction, accomplished by solving the system on a hierarchy of successively coarser grids. It is optimal in the sense that it exhibits asymptotic linear runtime in the number of unknowns. A detailed comparison of different solvers in the context of virtual cutting has revealed significantly improved convergence rates of multigrid methods compared to a Cholesky solver and a CG solver with Jacobi pre-conditioner. In particular it was demonstrated that the convergence rate of multigrid methods does not depend on the smoothness of the object boundary, which was commonly referred to as a main weakness of multigrid methods.

4. Meshfree Methods

In contrast to finite element methods, meshfree methods (also known as meshless methods) do not require a simulation grid. Instead, the material is represented by a set of moving simulation nodes, which interact with each other according to the governing equations of elasticity. The advantage of meshfree methods is that they do not need an explicit encoding of the material topology and can be used even in scenarios where the connectivity of the nodes is difficult to maintain without introducing errors [NTV92]. On the downside, meshfree methods need to compute node-to-node adjacency in every simulation step, making it necessary to maintain and update an additional search data structure.

In computer graphics, Desbrun and Cani are among the first to employ the concepts underlying meshfree methods for deformable body simulation. They animated soft substances that can split and merge by combining particle systems with inter-particle forces [DG95]. Müller et al. proposed point-based animation for a wide spectrum of volumetric objects [MKN*04]. Meshfree methods have also been used in offline fracturing [PKA*05] and interactive cutting [SOG06, PGCS09].

In meshfree methods, the object is sampled at a set of simulation nodes x_i. The deformation field is approximated by $u(x) = \sum_i \phi_i(x) u_i$, where u_i are the displacement vectors at the simulation nodes and ϕ_i are shape functions. The shape functions proposed in the computer graphics literature are usually constructed using the moving least squares (MLS) approximation [LS81]. Alternative designs of shape functions for meshfree methods can be found in engineering textbooks (e.g., [FM03]). The shape functions are weighted by a polynomial kernel $w(x, x_i, r_i)$, which rapidly decays with increasing distance between the simulation node x_i and the point x where the function is to be evaluated. The radius of influence r_i should be sufficiently small to adequately discretize displacement gradients. It is typically chosen in such a way that the influence region includes a constant number of neighbors of the node x_i.

Modeling discontinuity Meshfree methods model the material discontinuities caused by cutting (as well as initial surface concavities) by augmenting the shape functions. A straightforward way is by introducing the visibility criterion [BLG94], i.e., if a point x is invisible from the simulation node x_i due to the newly created surface, the shape function $\phi_i(x)$ is assigned a value of zero. A drawback of this solution is that it introduces an artificial discontinuity (see Figure 9 (a), the two sides of the ray starting from the discontinuity tip p are classified as visible and invisible respectively), which affects negatively numerical convergence and stability. To cope with this, Pietroni et al. extended the visibility criterion by introducing the concept of visibility disk and augmented the shape function by the ratio of the visible region within the disk [PGCS09]. While this approach alleviates the discontinuity caused by the binary-valued visibility criterion, a rigorous definition of the visibility disk has not been proposed so far.

The discontinuity can also be modeled by defining different distance measures for quantification of the distance between x and x_i, which is then considered in the weights of the shape functions. The transparency method [OFTB96] adds to the Euclidean distance between x and x_i a factor that depends on the distance from the discontinuous tip to the intersection of the line segment, $\overline{pa}$ in Figure 9 (b). This distance measure was used in offline simulations [PKA*05,GLB*06].

The diffraction method [OFTB96], which considers the diffraction of rays around the discontinuity tip, weights the Euclidean distance between x and x_i by their distances to the discontinuity tip, $\overline{px_i}$ and $\overline{px}$ in Figure 9 (c). Note that the diffraction and transparency methods were designed for simple 2D domains where the discontinuity tip is well defined. For efficient evaluation of diffraction distances in 3D, Steinemann et al. proposed the use of a visibility graph for estimating the distance along fully visible paths between two points [SOG06]. The distance is chosen as the shortest path in a pre-computed visibility graph, see Figure 9 (d), $x_i \rightarrow x_a \rightarrow x_b \rightarrow x$. Upon cutting, the intersected edges of the visibility graph are removed from the graph, and the shortest paths in the graph are updated accordingly.

Boundary surface Meshfree methods do not naturally provide a representation of the boundary. To create new surfaces after cutting, Steinemann et al. proposed to explicitly triangulate the swept surface of a cutting tool [SOG06]. The swept surface is trimmed with respect to the original surface of the object. In the context of fracturing, Pauly et al. modeled explicitly advancing crack fronts by continuously adding surface samples during crack propagation [PKA*05]. Instead of creating new surfaces explicitly, Pietroni et al. maintained a uniform hexahedral grid that is embedded into the simulation domain, and reconstructed from this grid a mesh corresponding to an implicit surface in the volume [PGCS09].

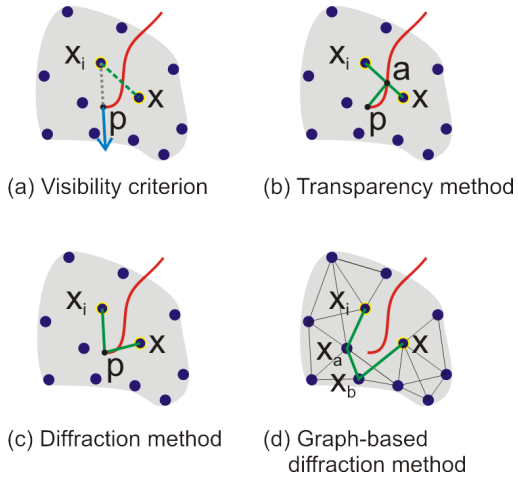

Figure 9: Discontinuity modeling in meshfree methods. The object (gray region) is sampled at a set of simulation nodes (blue dots). (a) The visibility criterion assigns a zero value to the shape function since $\overline{x_i x}$ intersects the cut (the red curve). (b) The transparency method enhances the Euclidean distance $\overline{x_i x}$ with the distance from the discontinuity tip to the intersection, $\overline{pa}$. (c) The diffraction method considers the distances from the tip to both nodes, $\overline{px_i}$ and $\overline{px}$. (d) The diffraction distance is approximated by the shortest path in a visibility graph, $\overline{x_i x_a}$, $\overline{x_a x_b}$, and $\overline{x_b x}$.

5. Summery of Techniques for Cutting Simulation

Since there exist so many different techniques for simulating cuts in deformable bodies, in the following we try to provide a comprehensive overview of these techniques according to a few specific categories. The overview is presented in Table 1. In particular, we classify techniques according to the discretization and the modeling of cuts (Geometry), the deformable model (Deformation), the time integration and numerical solver (Solver), and the intended application scenario (Scenario). We further give some remarks on specific properties of these techniques.

In Table 1 the different techniques are grouped into six categories. In the first category are techniques building on tetrahedral discretizations. It can be observed that these techniques are intended primarily for medical applications. The second and third categories comprise techniques building upon the virtual node algorithm and extended finite elements, respectively. In the fourth category are techniques using hexahedral discretizations. Into the fifth and sixth category, respectively, fall papers using polyhedral discretizations and meshfree methods.

We show representative simulation results for each group in Figures 12 and 13. Figure 12 shows scenarios simulated offline by (a) the virtual node algorithm, (b) the polyhedral finite element method, (c) the extended finite element method, (d) the hexahedral finite element method on an oc-

Reference	Geometry	Deformation	Solver	Scenario	Remark
Bielser et al. [BMG99, BG00, BGTG04]	Tet., refinement	Mass-spring	Explicit/Semi-implicit	Interactive	Basic tet. refinement
Cotin et al. [CDA00]	Tet., deletion	Tensor-mass	Explicit	Interactive	Hybrid elastic model
Mor & Kanade [MK00]	Tet., refinement	FEM	Explicit	Interactive	Progressive cutting
Nienhuys et al. [NFvdS00, NFvdS01]	Tet., boundary splitting/snapping	FEM	Static (CG solver)	Interactive	FEM with a CG solver
Bruyns et al. [BSM*02]	Tet., refinement	Mass-spring	Explicit	Interactive	An early survey
Steinemann et al. [SHGS06]	Tet., refinement + snapping	Mass-spring	Explicit	Interactive (Fig. 13 a)	Hybrid cutting
Chentanez et al. [CAR*09]	Tet., refinement	FEM	Implicit (CG solver)	Interactive (Fig. 13 d)	Needle insertion
Courtecuisse et al. [CJA*10, CAK*14]	Tet., deletion/refinement	FEM	Implicit (CG solver)	Interactive (Fig. 13 c,e)	Surgery applications
Molino et al. [MBF04]	Tet., duplication	FEM	Mixed explicit/implicit	Offline	Basic virtual node algorithm
Sifakis et al. [SDF07]	Tet., duplication	FEM		Offline (Fig. 12 a)	Arbitrary cutting
Jeřábková & Kuhlen [JK09]	Tet.	XFEM	Implicit (CG solver)	Interactive	Introduction of XFEM
Turkiyyah et al. [TKAN09]	Tri.	2D-XFEM	Static (direct solver)	Interactive	XFEM with a direct solver
Kaufmann et al. [KMB*09]	Tri./Quad.	2D-XFEM	Semi-implicit	Offline (Fig. 12 c)	Enrichment textures
Frisken-Gibson [FG99]	Hex., deletion	ChainMail	Local relaxation	Interactive	Linked volume
Jeřábková et al. [JBB*10]	Hex., deletion	CFEM		Interactive	CFEM
Dick et al. [DGW11a]	Hex., refinement	FEM	Implicit (multigrid)	Offline/Interactive (Fig. 12 d)	Linked octree, multigrid solver
Seiler et al. [SSSH11]	Hex., refinement	FEM	Implicit	Interactive	Octree, surface embedding
Wu et al. [WDW11, WBWD12, WDW13]	Hex., refinement	CFEM	Implicit (multigrid)	Interactive (Fig. 13 b, f)	Collision detection for CFEM
Wicke et al. [WBG07]	Poly., splitting	PFEM	Implicit	Offline (Fig. 12 b)	Basic polyhedral FEM
Martin et al. [MKB*08]	Poly., splitting	PFEM	Semi-implicit	Offline	Harmonic basis functions
Pauly et al. [PKA*05]	Particles, transparency	Meshfree	Explicit	Offline	Fracture animation
Steinemann et al. [SOG06]	Particles, diffraction	Meshfree		Offline/Interactive (Fig. 12 e)	Splitting fronts propagation
Pietroni et al. [PGCS09]	Particles, visibility	Meshfree		Interactive	Splitting cubes algorithm

Table 1: An overview of cutting techniques outlined in this report.

tree grid, and (e) the meshfree method. Figure 13 shows interactive simulation scenarios in various medical contexts, such as (a) ablating a polyp in a hysteroscopy simulator, using element refinement together with snapping of vertices, (b) virtual soft tissue cutting and shrinkage simulation, by modeling the residual stress in biological tissues, (c) real-time simulation of a brain tumor resection, using an asynchronous pre-conditioner, (d) needle insertion in a prostate brachytherapy simulator with a parallelized CG solver, (e) real-time simulation of laparoscopic hepatectomy dealing with complex contacts, and (f) haptic-enabled real-time virtual cutting of high-resolution soft tissues, using composite finite elements and a multigrid solver.

6. Application Study on Cutting Simulation

In the following application study we intend to shed light on the performance of physically-based cutting simulation and, by this, to assess the model resolution that can be handled in interactive scenarios requiring update rates of 20-30 Hz. We restrict ourselves to the analysis of one specific simulation approach for which a highly optimized implementation is available. Even though this approach has limitations, we believe that it allows for a very good estimation of the simulation efficiency that can be achieved when the model of linear elasticity is used.

Figure 10 shows our experiment setup in which we simulate a cut in a linear elastic liver model. All experiments were performed on a standard desktop PC equipped with an Intel Xeon X5560 processor (a single core was used) and 8 GB main memory.

We analyze three variants of the hexahedral finite element approach proposed by Dick et al [DGW11a]. It uses the corotational formulation of finite elements, which simulates linear dependencies between the components of stress and strain, and considers the geometric non-linearity by respecting per-element rotations in the strain computation. While finite element discretizations enable high accuracy, hexahedral elements are well suited for constructing a mesh hierarchy that can be used by a geometric multigrid solver to achieve optimal convergence rates. On the other hand, since hexahedral simulation elements are not aligned with the object boundaries, approaches using unstructured tetrahedral simulation grids might be favorable when smooth boundary-aware discretizations of the simulation domain are required.

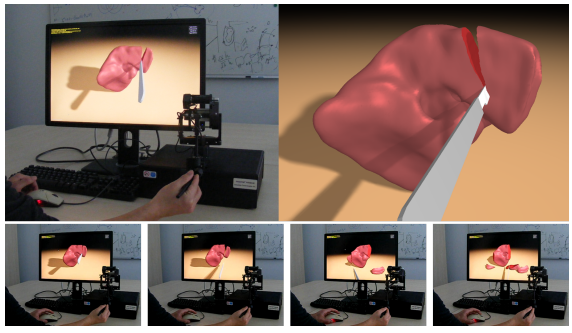

Figure 10: Left: Experiment setup. To cut a liver model the user manipulates a haptic device that is mapped to a scalpel. Right: High quality surface rendering. Bottom: A sequence of images from a live recording, available at http://wwwcg.in.tum.de/research/research/projects/real-time-haptic-cutting.html.

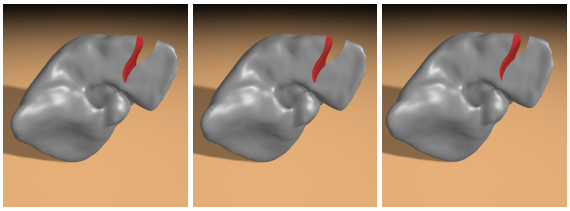

Figure 11: From left to right: Simulation of cuts using finite elements on a uniform hexahedral grid, finite elements on an adaptive octree grid, and composite finite elements on an adaptive octree grid.

	Uniform	Adaptive	Composite (2 levels)
Coarse resolution		21×21×25	21×21×25
Refined resolution	82×83×100	82×83×100	82×83×100
# Cells (initial)	173 843	40 080	3 439
# DOFs (initial)	566 493	129 162	13 557
# Cells (added due to cut)	0	1 596	39
# DOFs (added due to cut)	2 037	6 438	318
Octree subdivision (t_1)	0	13.29	13.39
Surface meshing (t_2)	1.26	1.26	1.24
FE matrices (t_3)	29.57	7.05	20.99
Multigrid hierarchy (t_4)	40.34	10.09	2.06
Solver (t_5)	2 033.09	581.66	40.61
Simulation per cut ($\sum_{i=1}^{5} t_i$)	2 104.26	613.35	78.29

Table 2: Timings (in milliseconds) for cutting simulations using finite elements on a *uniform* hexahedral grid, finite elements on an *adaptive* octree grid, and *composite* finite elements on an adaptive octree grid.

For instance, to perform accurate collision detection and response. In all of our experiments, a high-resolution surface is generated from the cut object using the dual contouring algorithm on the hexahedral grid, and this surface is used for rendering and collision detection [WDW13].

Our first variant uses finite elements on a uniform hexahedral grid and realizes cuts by simply disconnecting elements along element faces. A high-resolution finite element model consisting of 173,843 hexahedral elements (566,493 DOFs) on a $82 \times 83 \times 100$ uniform grid is used as our reference solution (see Fig. 11 (left)). We simulate the cut by instantly bringing the cutting tool into its final position and performing all required operations like finding and disconnecting edges in the simulation grid, FE matrix assembly, and numerical multigrid solver execution. In particular, we perform 2 V-cycles including 4 pre- and post-smoothing Gauss-Seidel relaxation steps, which reduces the error to below 1%. Note that while cutting on a uniform grid does not add new elements, the number of DOFs is increased, since some of the originally shared element vertices become separated vertices due to the cut.

The second variant uses finite elements on an adaptive octree grid. This grid is constructed by starting with a uniform coarse hexahedral grid, which is adaptively refined along the initial object boundary and the cut, until a user-selected level is reached. For this variant we have set the resolution of the initial coarse grid and the refinement depth such that in the refined regions the grid resolution of the first variant is reached. The third variant uses the same adaptive grid as the second one, but instead of standard finite elements, it uses composite finite elements at the resolution of the initial coarse grid [WDW11]. Since the refined elements are used only to correct the coarse grid simulation, considerably higher performance is achieved. The last variant is intended to demonstrate the trade-offs between highest accuracy and speed in interactive scenarios when employing the principle of homogenization for linear elasticity [KMOD09]. Since the adaptive variants restrict the element refinement to a user-selected depth, alternative (offline) approaches like extended finite elements [JK09, KMB*09] can be favorable in applications where cuts should be modeled at sub-grid accuracy.

Figure 11 (middle) shows the same cut as before, but now the second variant is used to simulate the cut. We start with a $21 \times 21 \times 25$ uniform grid. Initially, this grid is refined adaptively along the object boundary via two levels of subdivision. When the cut is simulated, the grid is further refined along the cut using the same refinement depth, resulting in 41,676 hexahedral elements (135,600 DOFs).

In the last experiment (see Fig. 11 (right)) we start with the same initial grid as in the second experiment, and we apply exactly the same adaptive grid refinement along the object boundary and the cut. However, the adaptively generated elements are not considered as DOFs in the simulation, but they are used to assemble the coarse grid matrices according to their contributions. Thus, the simulation is performed using 3,439 composite elements (13,557 DOFs).

Table 2 lists the times that are consumed by the different processes in each experiment. It can be seen that even though a large number of DOFs can be simulated in roughly 2 seconds using a uniform simulation grid, the grid resolution has to be reduced about a factor of 4 in each dimension to make the uniform grid suitable for interactive scenarios. Via the adaptive octree grid the cut can be simulated at almost no visual difference to the high-resolution reference solution. Due to the restriction of element refinements along the initial object boundary and the cut, the overall simulation time can be reduced by a factor of 3.5. Using composite finite elements, the number of simulation elements to be considered by the numerical solver can be decreased further, making this approach suitable for interactive scenarios. Despite the low number of DOFs to be solved for, the simulation result is in very good agreement with the results generated by the other variants. It is clear, however, that due to the reduced

number of DOFs, the simulated deformations cannot exactly match the high-resolution reference in general.

Table 2 further indicates that surface meshing does not have any impact on the overall performance. This is because it works only on the boundary elements. Since the effective resolution of the boundary elements is the same in all three experiments, surface meshing always consumes the same amount of time. It can be seen that in addition to the time consumed by the multigrid solver, especially in the interactive variant the adaptive grid refinement (t_1) and the assembly of the finite element matrices (t_3) take up a considerable amount of the overall time. In this variant, the time required to generate the multigrid hierarchy (t_4) is rather low due to the low resolution of the simulation grid. This variant requires a grid hierarchy from the finest level to the coarse simulation level as well for assembling the FE matrices on the simulation grid. The time for updating this part of the hierarchy is counted in t_3 for this variant.

7. Discussion and Conclusion

In this report we have reviewed the current state-of-the-art in computer-aided simulation of cuts in deformable bodies. We have discussed distinct geometry and topology representations, specifically-tailored finite element approaches, and meshfree methods, with respect to accuracy, robustness, and computational efficiency.

The analysis of current techniques indicates a clear trend towards physically-based simulations. From our experience this trend is driven by the application domains in which virtual cutting is applied. Especially in virtual surgery simulators, which are used for training and preoperative planning, doctors are more and more demanding for reliable simulations that can accurately predict the behavior of soft tissue undergoing cuts and deformations thereof. Thus, going beyond this STAR we see the urgent need for a benchmark that is tailored to the problem of virtual cutting simulation and that can be used to assess simulation techniques quantitatively.

Furthermore, especially in medical applications the accurate modeling of real-world material is becoming of ever increasing importance. Going beyond the model of linear elasticity and homogeneous material, soft tissues exhibiting nonlinear, anisotropic, viscoelastic and even viscoplastic behavior [Hum03] need to be considered by interactive simulators in the future. However, even though it is known in principle how to model such tissue types physically, we see the efficient numerical simulation of these types as one of the most important research questions for the future.

One possibility to achieve interactive cutting simulation even for complex tissue types is the use of dedicated parallelization strategies on multi-core and multi-GPU architectures. On single GPUs, the parallelization of deformable body simulation has already shown significant performance improvements [DGW11b, CJA*10]. The layout of numerical solvers across multiple CPU or GPU nodes, however, is extremely challenging. In particular for the parallelization of an initially sequential solver, typically frequent communication between the nodes is required, letting bandwidth and latency become quickly the bottleneck. It is therefore required to develop parallel solvers that are particularly tailored to such computing architectures by reducing the communication between the nodes. A promising approach that needs further investigation are domain decomposition methods, which divide the problem into subproblems that can be solved independently.

Another direction of research is the development of approaches for modeling the physical interaction between a scalpel and soft tissues accurately [MH01, CDL07, MRO08]. For simplicity, most virtual cutting techniques assume that the material is separated as long as it is swept by a cutting tool. In the physical world, however, we can observe that there is a deformation of soft tissues before a cut happens. The simulation of this kind of tool-object interaction may benefit from general contact resolution techniques [HVS*09, AFC*10, SH12].

Finally, the discussion of techniques in this report has revealed that two major, and somehow opposing, requirements reflect in the design of cutting techniques. On the one hand, one seeks to use unstructured spatial object discretizations to accurately model a cut. This has led to geometric techniques using tetrahedral or polyhedral meshes, which are re-meshed irregularly along the cut. On the downside, the re-meshing step becomes very elaborate for arbitrary cutting paths, and it increases the number of simulation elements significantly.

On the other hand, to achieve high performance of the numerical solver used to simulate the dynamic behavior of the cut body, structured simulation grids have turned out to be favorable. Scalable solvers exhibiting linear complexity in the number of supporting vertices have been achieved via geometric multigrid methods on semi-regular hexahedral grids. While building geometric multigrid hierarchies on hexahedral grids is simple, on unstructured grids the construction of such hierarchies is extremely complicated and very time-consuming. In general, however, elements in hexahedral grids are not aligned with the object boundaries, introducing modeling inaccuracies along these boundaries.

In our opinion it is one of the most interesting questions whether adaptive spatial discretizations can be found that give rise to efficient numerical solution techniques at the same time allowing for an accurate alignment of simulation elements along the object boundaries.

Author Biographies

Jun Wu is a research assistant in the Computer Graphics and Visualization Group at the Technische Universität München, Germany. He received a B.Eng. in astronautical engineering

in 2006 and a Ph.D. in mechanical engineering in 2012, both from Beihang University, Beijing, China. His research interests include physically-based modeling and interactive simulation, haptic rendering, and their applications in surgery simulation.

Rüdiger Westermann studied computer science at the Technical University Darmstadt, Germany. He pursued his doctoral thesis on multiresolution techniques in volume rendering, and he received a Ph.D. in computer science from the University of Dortmund, Germany. In 1999, he was a visiting professor at the University of Utah in Salt Lake City, and he became an assistant professor at the University of Stuttgart, Germany. In 2000, he was appointed an associate professor at the Technical University Aachen, Germany, where he was head of the Scientific Visualization and Imaging Group. In 2002, he was appointed the chair of Computer Graphics and Visualization at the Technische Universität München. His research interests include uncertainty and large data visualization, large-scale physically-based simulation, interactive and GPU techniques in computer graphics and visualization.

Christian Dick is a PostDoc in the Computer Graphics and Visualization Group at the Technische Universität München, Germany. He received a diploma in computer science in 2007 and a Ph.D. in 2012, both from Technische Universität München. His research interests include physics-based simulation of deformable objects and fluids, simulation and visualization in the context of medical surgery planning and training, and visualization of very large scientific data sets such as terrain data.

References

[AFC*10] Allard J., Faure F., Courtecuisse H., Falipou F., Duriez C., Kry P. G.: Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29*, 4 (July 2010), 82:1–82:10. 15

[Bat96] Bathe K.-J.: *Finite Element Procedures*. Prentice Hall, 1996. 7

[BB99] Belytschko T., Black T.: Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering 45*, 5 (1999), 601–620. 8

[BG00] Bielser D., Gross M.: Interactive simulation of surgical cuts. In *Proceedings of Pacific Graphics* (2000), pp. 116–442. 3, 4, 13

[BGTG04] Bielser D., Glardon P., Teschner M., Gross M.: A state machine for real-time cutting of tetrahedral meshes. *Graphical Models 66*, 6 (2004), 398 – 417. 3, 4, 13

[BLG94] Belytschko T., Lu Y. Y., Gu L.: Element-free galerkin methods. *International Journal for Numerical Methods in Engineering 37*, 2 (1994), 229–256. 12

[BM97] Babuška I., Melenk J. M.: The partition of unity method. *International Journal for Numerical Methods in Engineering 40*, 4 (1997), 727–758. 8

[BMG99] Bielser D., Maiwald V. A., Gross M. H.: Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum 18*, 3 (1999), 31–38. 2, 3, 4, 13

[BN98] Bro-Nielsen M.: Finite element modeling in surgery simulation. *Proceedings of the IEEE 86*, 3 (1998), 490–503. 7

[BNC96] Bro-Nielsen M., Cotin S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum 15*, 3 (1996), 57–66. 11

[BS01] Bruyns C. D., Senger S.: Interactive cutting of 3D surface meshes. *Computers & Graphics 25*, 4 (2001), 635 – 642. 3

[BSM*02] Bruyns C. D., Senger S., Menon A., Montgomery K., Wildermuth S., Boyle R.: A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The Journal of Visualization and Computer Animation 13*, 1 (2002), 21–42. 1, 13

[CAK*14] Courtecuisse H., Allard J., Kerfriden P., Bordas S. P., Cotin S., Duriez C.: Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis 18*, 2 (2014), 394 – 410. 13, 19

[CAR*09] Chentanez N., Alterovitz R., Ritchie D., Cho L., Hauser K. K., Goldberg K., Shewchuk J. R., O'Brien J. F.: Interactive simulation of surgical needle insertion and steering. *ACM Trans. Graph. 28*, 3 (July 2009), 88:1–88:10. 1, 11, 13, 19

[CDA00] Cotin S., Delingette H., Ayache N.: A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer 16*, 8 (2000), 437–452. 2, 3, 4, 13

[CDL07] Chanthasopeephan T., Desai J., Lau A.: Modeling soft-tissue deformation prior to cutting for surgical simulation: Finite element analysis and study of cutting parameters. *IEEE Transactions on Biomedical Engineering 54*, 3 (2007), 349–359. 3, 15

[CJA*10] Courtecuisse H., Jung H., Allard J., Duriez C., Lee D., Cotin S.: GPU-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology 103*, 2-3 (2010), 159 – 168. 11, 13, 15, 19

[DG95] Desbrun M., Gascuel M.-P.: Animating soft substances with implicit surfaces. In *Proceedings of SIGGRAPH* (1995), pp. 287–290. 11

[DGBW08] Dick C., Georgii J., Burgkart R., Westermann R.: Computational steering for patient-specific implant planning in orthopedics. In *Proceedings of Visual Computing for Biomedicine* (2008), pp. 83–92. 6

[DGW11a] Dick C., Georgii J., Westermann R.: A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (2011), 1663–1675. 2, 6, 9, 11, 13, 19

[DGW11b] Dick C., Georgii J., Westermann R.: A real-time multigrid finite hexahedra method for elasticity simulation using CUDA. *Simulation Modelling Practice and Theory 19*, 2 (2011), 801–816. 15

[ED08] Eisemann E., Décoret X.: Single-pass GPU solid voxelization for real-time applications. In *Proceedings of Graphics Interface* (2008), pp. 73–80. 6

[FG99] Frisken-Gibson S.: Using linked volumes to model object collisions, deformation, cutting, carving, and joining. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 333–348. 6, 13

[FM03] Fries T.-P., Matthies H. G.: Classification and overview of meshfree methods. *Technical University of Braunschweig* (2003). 11

[GCMS00] Ganovelli F., Cignoni P., Montani C., Scopigno R.: A multiresolution model for soft objects supporting interactive cuts

and lacerations. *Computer Graphics Forum 19*, 3 (2000), 271–281. 3, 4

[GLB*06] Guo X., Li X., Bao Y., Gu X., Qin H.: Meshless thin-shell simulation based on global conformal parameterization. *IEEE Transactions on Visualization and Computer Graphics 12* (2006), 375–385. 12

[GW06] Georgii J., Westermann R.: A multigrid framework for real-time simulation of deformable bodies. *Computer & Graphics 30* (2006), 408–415. 2

[Hag89] Hager W. W.: Updating the inverse of a matrix. *SIAM Review 31*, 2 (1989), 221–239. 11

[HLSO12] Hecht F., Lee Y. J., Shewchuk J. R., O'Brien J. F.: Updated sparse cholesky factors for corotational elastodynamics. *ACM Transactions on Graphics 31*, 5 (Oct. 2012), 123:1–13. 11

[HS97] Hackbusch W., Sauter S.: Composite finite elements for the approximation of PDEs on domains with complicated microstructures. *Numerische Mathematik 75*, 4 (1997), 447–472. 8

[Hum03] Humphrey J.: Review paper: Continuum biomechanics of soft biological tissues. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 459*, 2029 (2003), 3–46. 15

[HVS*09] Harmon D., Vouga E., Smith B., Tamstorf R., Grinspun E.: Asynchronous contact mechanics. *ACM Trans. Graph. 28*, 3 (July 2009), 87:1–87:12. 15

[JBB*10] Jeřábková L., Bousquet G., Barbier S., Faure F., Allard J.: Volumetric modeling and interactive cutting of deformable bodies. *Progress in Biophysics and Molecular Biology 103*, 2-3 (2010), 217 – 224. 2, 3, 6, 9, 13

[JK09] Jeřábková L., Kuhlen T.: Stable cutting of deformable objects in virtual environments using XFEM. *IEEE Computer Graphics and Applications 29*, 2 (2009), 61–71. 2, 8, 13, 14

[JLSW02] Ju T., Losasso F., Schaefer S., Warren J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (2002), 339–346. 6

[KMB*09] Kaufmann P., Martin S., Botsch M., Grinspun E., Gross M.: Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. 28*, 3 (July 2009), 50:1–50:10. 2, 3, 8, 13, 14, 19

[KMBG08] Kaufmann P., Martin S., Botsch M., Gross M.: Flexible simulation of deformable models using discontinuous galerkin FEM. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), pp. 105–115. 10

[KMOD09] Kharevych L., Mullen P., Owhadi H., Desbrun M.: Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph. 28*, 3 (July 2009), 51:1–51:8. 14

[LJD07] Lim Y.-J., Jin W., De S.: On some recent advances in multimodal surgery simulation: A hybrid approach to surgical cutting and the use of video images for enhanced realism. *Presence: Teleoper. Virtual Environ. 16*, 6 (Dec. 2007), 563–583. 3

[LPO10] Lee B., Popescu D. C., Ourselin S.: Topology modification for surgical simulation using precomputed finite element models based on linear elasticity. *Progress in Biophysics and Molecular Biology 103*, 2-3 (2010), 236 – 251. 11

[LPR*09] Liehr F., Preusser T., Rumpf M., Sauter S., Schwen L. O.: Composite finite elements for 3D image based computing. *Computing in Visualization and Science 12*, 4 (2009), 171–188. 8

[LS81] Lancaster P., Salkauskas K.: Surfaces generated by moving least squares methods. *Mathematics of Computation 37*, 155 (1981), 141–158. 11

[LT07] Lindblad A., Turkiyyah G.: A physically-based framework for real-time haptic cutting and interaction with 3D continuum models. In *Proceedings of ACM symposium on Solid and Physical Modeling* (2007), pp. 421–429. 3, 4, 8, 11

[MBF04] Molino N., Bao Z., Fedkiw R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 385–392. 3, 5, 10, 13

[MDB99] Moës N., Dolbow J., Belytschko T.: A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering 46*, 1 (1999), 131–150. 8

[MG04] Müller M., Gross M.: Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), pp. 239–246. 3

[MH01] Mahvash M., Hayward V.: Haptic rendering of cutting: A fracture mechanics approach. *Haptics-e 2*, 3 (2001), 1–12. 15

[MK00] Mor A. B., Kanade T.: Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention* (2000), pp. 598–607. 2, 3, 4, 13

[MKB*08] Martin S., Kaufmann P., Botsch M., Wicke M., Gross M.: Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum 27*, 5 (2008), 1521–1529. 2, 7, 10, 13

[MKN*04] Müller M., Keiser R., Nealen A., Pauly M., Gross M., Alexa M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 141–151. 2, 11

[MRO08] Misra S., Ramesh K. T., Okamura A. M.: Modeling of tool-tissue interactions for computer-based surgical simulation: A literature review. *Presence: Teleoper. Virtual Environ. 17*, 5 (Oct. 2008), 463–491. 15

[NFvdS00] Nienhuys H.-W., Frank van der Stappen A.: Combining finite element deformation with cutting for surgery simulations. In *EuroGraphics short presentations* (2000), pp. 43–52. 2, 3, 4, 11, 13

[NFvdS01] Nienhuys H.-W., Frank van der Stappen A.: A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001*, Niessen W., Viergever M., (Eds.), vol. 2208 of *Lecture Notes in Computer Science*. 2001, pp. 145–152. 3, 4, 11, 13

[NKJF09] Nesme M., Kry P. G., Jeřábková L., Faure F.: Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3 (July 2009), 52:1–52:9. 9

[NMK*06] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836. 7

[NTV92] Nayroles B., Touzot G., Villon P.: Generalizing the finite element method: Diffuse approximation and diffuse elements. *Computational Mechanics 10*, 5 (1992), 307–318. 11

[OBH02] O'Brien J. F., Bargteil A. W., Hodgins J. K.: Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH* (2002), pp. 291–294. 2

[OFTB96] Organ D., Fleming M., Terry T., Belytschko T.: Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Computational Mechanics 18*, 3 (1996), 225–235. 12

[OH99] O'Brien J. F., Hodgins J. K.: Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH* (1999), pp. 137–146. 2

[PGCS09] Pietroni N., Ganovelli F., Cignoni P., Scopigno R.: Splitting cubes: a fast and robust technique for virtual cutting. *The Visual Computer 25*, 3 (Feb. 2009), 227–239. 2, 6, 11, 12, 13

[PKA*05] Pauly M., Keiser R., Adams B., Dutré P., Gross M., Guibas L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (July 2005), 957–964. 2, 11, 12, 13

[PRS07] Preusser T., Rumpf M., Schwen L. O.: Finite element simulation of bone microstructures. In *Proceedings of the 14th Workshop on the Finite Element Method in Biomedical Engineering, Biomechanics and Related Fields* (July 2007), pp. 52–66. 8

[SCB01] Strouboulis T., Copps K., Babuska I.: The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering 190*, 32-33 (2001), 4081–4193. 8

[SDF07] Sifakis E., Der K. G., Fedkiw R.: Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 73–80. 3, 5, 10, 13, 19

[SH12] Spillmann J., Harders M.: Robust interactive collision handling between tools and thin volumetric objects. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (Aug. 2012), 1241–1254. 15

[She94] Shewchuk J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep., Carnegie Mellon University, 1994. 11

[She02] Shewchuk J.: *What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)*. Tech. rep., University of California at Berkeley, 2002. 3

[SHGS06] Steinemann D., Harders M., Gross M., Szekely G.: Hybrid cutting of deformable solids. In *Virtual Reality Conference* (2006), pp. 35–42. 3, 4, 13, 19

[Si06] Si H.: *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, 2006. http://tetgen.org. 3

[SMMB00] Sukumar N., Moës N., Moran B., Belytschko T.: Extended finite element method for three-dimensional crack modelling. *International Journal for Numerical Methods in Engineering 48*, 11 (2000), 1549–1570. 8

[SOG06] Steinemann D., Otaduy M. A., Gross M.: Fast arbitrary splitting of deforming objects. In *Proceedings of ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 63–72. 2, 11, 12, 13, 19

[SSSH11] Seiler M., Steinemann D., Spillmann J., Harders M.: Robust interactive cutting based on an adaptive octree simulation mesh. *The Visual Computer 27*, 6-8 (2011), 519–529. 2, 6, 13

[SW06] Sauter S., Warnke R.: Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing 77*, 1 (2006), 29–55. 8

[TKAN09] Turkiyyah G., Karam W. B., Ajami Z., Nasri A.: Mesh cutting during real-time physical simulation. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling* (2009), pp. 159–168. 11, 13

[TKH*05] Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Fuhrmann A., Cani M.-P., Faure F., Magnenat-Thalmann N., Strasser W., Volino P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (2005), 61–81. 1

[TMFB05] Teran J., Molino N., Fedkiw R., Bridson R.: Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers 21*, 1 (2005), 2–18. 3

[TMY*11] Tang M., Manocha D., Yoon S.-E., Du P., Heo J.-P., Tong R.-F.: VolCCD: Fast continuous collision culling between deforming volume meshes. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 111:1–111:15. 1

[WBG07] Wicke M., Botsch M., Gross M.: A finite element method on convex polyhedra. *Computer Graphics Forum 26*, 3 (2007), 355–364. 2, 3, 7, 10, 13, 19

[WBWD12] Wu J., Bürger K., Westermann R., Dick C.: Interactive residual stress modeling for soft tissue simulation. In *Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine* (2012), pp. 81–89. 3, 19

[WDW11] Wu J., Dick C., Westermann R.: Interactive high-resolution boundary surfaces for deformable bodies with changing topology. In *Proceedings of 8th Workshop on Virtual Reality Interaction and Physical Simulation* (2011), pp. 29–38. 2, 6, 9, 13, 14

[WDW13] Wu J., Dick C., Westermann R.: Efficient collision detection for composite finite element simulation of cuts in deformable bodies. *The Visual Computer 29*, 6-8 (2013), 739–749. 1, 3, 13, 14

[WH05] Wu W., Heng P. A.: An improved scheme of an interactive finite element model for 3d soft-tissue cutting and deformation. *The Visual Computer 21*, 8-10 (2005), 707–716. 11

[WWD14] Wu J., Westermann R., Dick C.: Real-time haptic cutting of high resolution soft tissues. *Studies in Health Technology and Informatics (Proc. Medicine Meets Virtual Reality 21) 196* (2014), 469–475. 19

[WWWZ10] Wu J., Wang D., Wang C. C. L., Zhang Y.: Toward stable and realistic haptic interaction for tooth preparation simulation. *Journal of Computing and Information Science in Engineering 10*, 2 (2010), 021007:1–9. 3

[ZBS05] Zhang Y., Bajaj C., Sohn B.-S.: 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering 194*, 48-49 (2005), 5083–5106. 3, 6

[ZWP05] Zhong H., Wachowiak M. P., Peters T. M.: Adaptive finite element technique for cutting in surgical simulation. In *Medical Imaging 2005: Visualization, Image-Guided Procedures, and Display*, Robert L. Galloway J., Cleary K. R., (Eds.), vol. 5744 of *Proc. SPIE*. 2005, pp. 604–611. 11

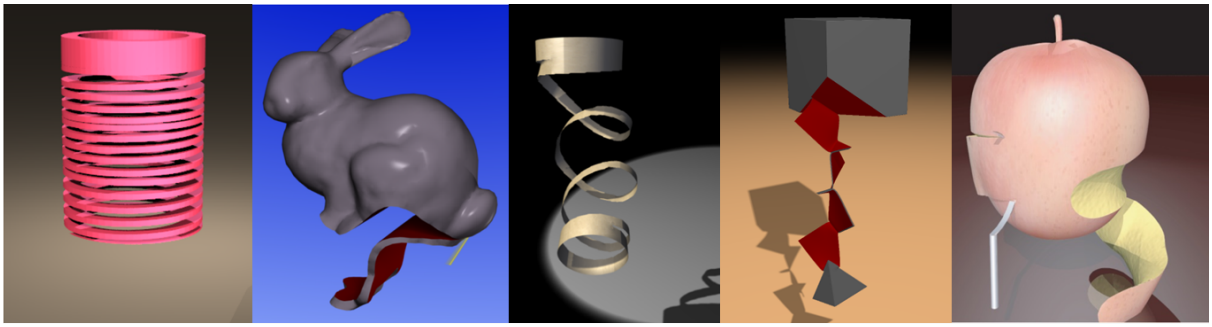

Figure 12: Offline progressive cutting scenarios simulated by (a) the virtual node algorithm on a tetrahedral mesh (image courtesy of Sifakis et al. [SDF07] ©2007 ACM), (b) the polyhedral finite element method (image courtesy of Wicke et al. [WBG07] ©2007 WILEY), (c) the extended finite element method on quads (image courtesy of Kaufmann et al. [KMB*09] ©2009 ACM), (d) the hexahedral finite element method on an adaptive octree grid [DGW11a], and (e) the meshfree method (image courtesy of Steinemann et al. [SOG06] ©2006 Eurographics). Copyrighted materials, image a, b, c, and e, are reprinted with permissions from ACM, WILEY, ACM, and Eurographics, respectively.

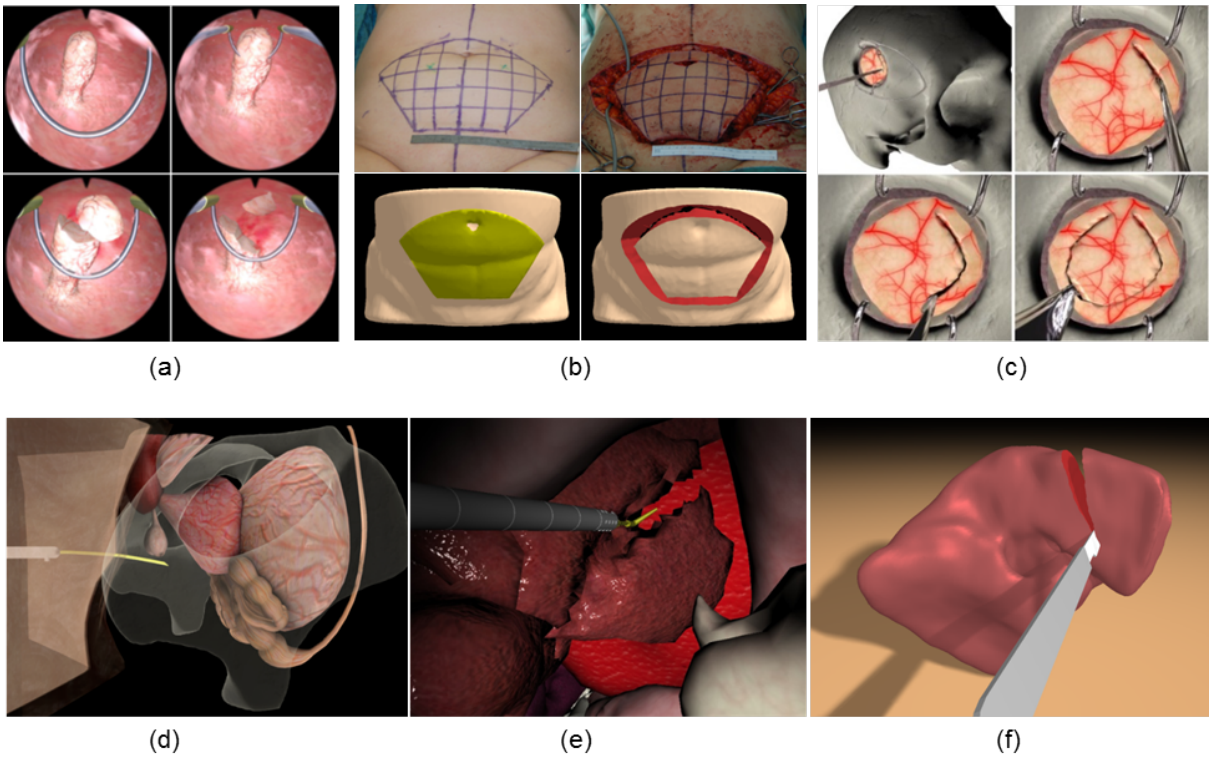

Figure 13: Interactive simulation in medical contexts. (a) Ablating a polyp in a hysteroscopy simulator (image courtesy of Steinemann et al. [SHGS06] ©2006 IEEE). (b) Virtual soft tissue cutting and shrinkage simulation [WBWD12] (abdomen photographs courtesy of Dr. med. Laszlo Kovacs). (c) Real-time simulation of a brain tumor resection (image courtesy of Courtecuisse et al. [CAK*14] ©2014 Elsevier). (d) Needle insertion in a prostate brachytherapy simulator (image courtesy of Chentanez et al. [CAR*09] ©2009 ACM). (e) Real-time simulation of laparoscopic hepatectomy (image courtesy of Courtecuisse et al. [CJA*10] ©2010 Elsevier). (f) Haptic-enabled virtual cutting of high-resolution soft tissues [WWD14]. Copyrighted materials, image a, c, d, and e, are reprinted with permissions from IEEE, Elsevier, ACM, and Elsevier, respectively.