Volume ? (201?), Number ? pp. 1–27 COMPUTER GRAPHICS *forum*

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

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 modeling and 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. Furthermore, we discuss cutting-related research problems such as collision detection and haptic rendering in the context of interactive cutting scenarios. The report is complemented with an application study to assess the performance of virtual cutting simulators.

Keywords: *Virtual Cutting, Deformable Bodies, Physically-based Modeling, Finite Elements, Surgery Simulation*

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 (illustrated in Figure 1): 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 the basic principles underlying techniques for collision detection in virtual cutting in principle are not different to those used in deformable body simulation, this report summarizes only the particular adaptations that have been proposed in the context of interactive cutting simulation. For a broader 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]. The fracture process driven by cutting tools, 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-

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

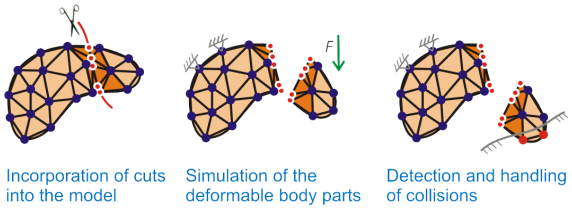

Figure 1: Illustration of the three major tasks involved in mesh-based virtual cutting simulations.

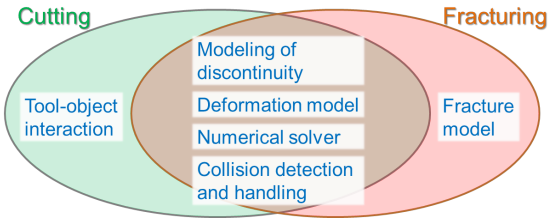

Figure 2: Simulation components in cutting and fracturing. In this report we focus on the components that are common to both simulation tasks.

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 topological 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. The relation between virtual cutting and fracturing is illustrated in Figure 2. 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], and to a recent survey on fracture modeling [MBP14] which discusses geometry- and image-based approaches as well.

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 (e.g., [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].

A topic beyond the scope of this paper is the realistic texturing of the induced cutting surfaces, such that the body's internal structures are displayed. To this end, 3D solid textures are employed, which can be obtained by texture synthesis [PCOS10] or from slice-based real-world data.

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 meshfree approaches are discussed in Section 3 and in Section 4 respectively. Numerical solvers are reviewed in Section 5. A summary of the surveyed techniques and representative simulation scenarios are presented in Section 6, followed by a discussion of techniques for collision detection and haptic rendering in interactive scenarios in Section 7. 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 8. The report is concluded in Section 9 with a discussion of future research challenges. A brief introduction of deformable body simulation using the finite element method and meshfree methods is given in the appendix.

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 hexahedral 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. Geometric 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 happens locally and advances gradually, a large region of the deformable object can be pruned before elementary intersection tests are performed using bounding volume hierarchies, and 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. An 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.

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 preoperative surgery planning [WBWD12]. Prescribing a cutting surface in the reference configuration allows for precisely applying a certain cutting shape, simplifies the intersection tests, and avoids the possible problems with temporally discrete intersection testing.

While the simulation of the progressive cutting process is important for animation and interactive applications, in some special cases such as surgery planning, the dynamic process might be of less importance compared to the finally resulting shapes. In these cases the entire cut can be introduced in a single step, which potentially simplifies remeshing operations.

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 3 for a 2D illustration):

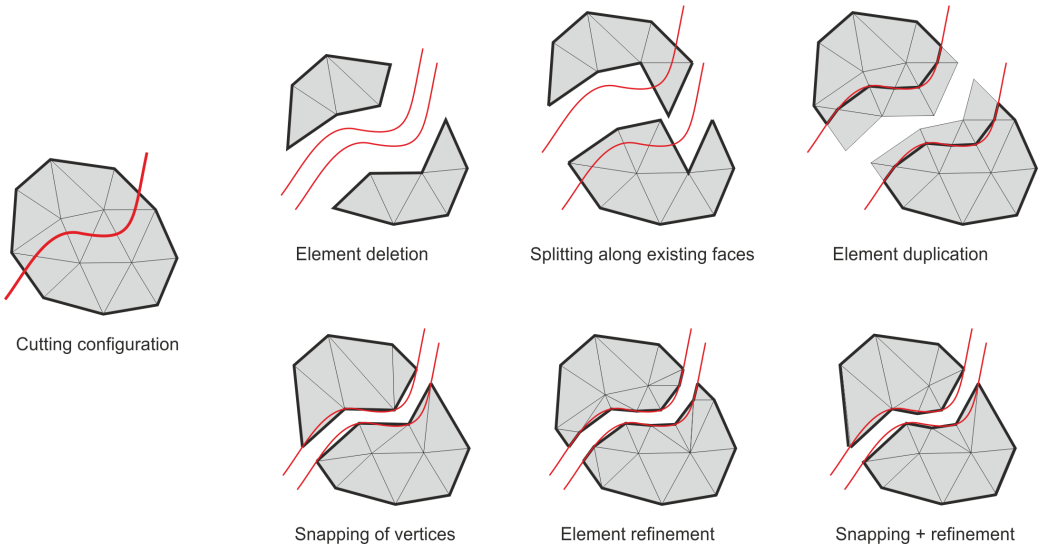

Figure 3: Illustration of different methods for incorporating cuts into a tetrahedral mesh (a triangle mesh in 2D). The red cutting path separates the object into two disconnected parts, which are illustratively displaced to make the discontinuity visible. The surface of the object (bold black line) 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.

- 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, LZW*14], 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 elements 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 sub-

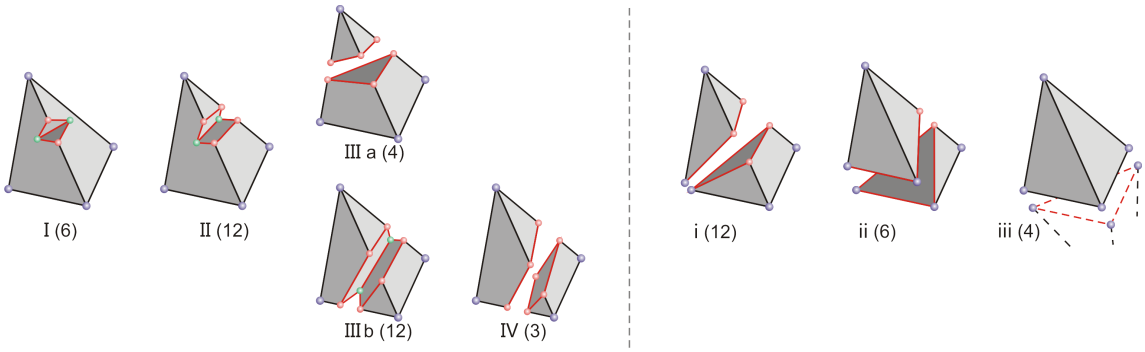

Figure 4: Left: A cut tetrahedron can have five topologically different configurations. The Roman numeral represents the number of disconnected edges. The number in parentheses indicates the number of topologically equivalent configurations by rotation and mirroring operations. Right: In the hybrid cutting 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.

division 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 4 (left) illustrates the five topologically different configurations of a tetrahedron after introduction of a cut. Among these five configurations, III a 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 5 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 cutting 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 4 (left), by three additional topological configurations, illustrated in Figure 4 (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.

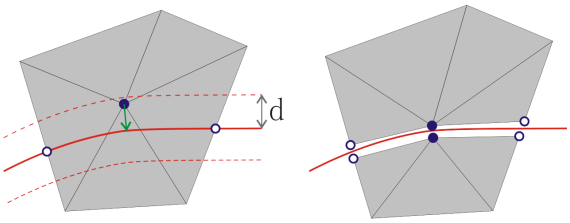

Figure 5: A hybrid cutting approach based on both snapping of vertices and element refinement. If the intersection between an edge and the cutting surface is close to one of the edge's vertices (determined by a threshold d), the vertex is moved onto the cutting surface, in order to prevent the creation 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 duplication of elements, the volumetric representation and the surface representation are topologically consistent.

Wang et al. redeveloped the virtual node algorithm [WJST14]. Their version allows for cuts passing through mesh vertices or lying on mesh edges and faces (without the need of ambiguous perturbation of the cutting surfaces as in the original version), enables multiple cuts per tetrahedron face (at a lower algorithmic complexity than [SDF07]), and includes a mesh intersection routine that is provably robust in the context of floating point rounding errors.

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].

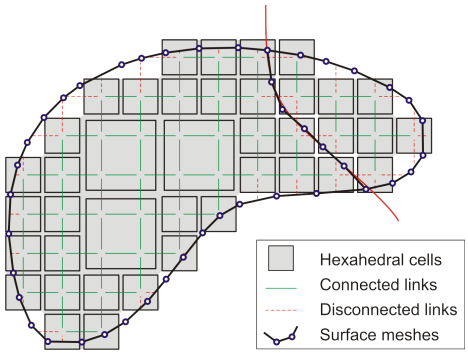

Figure 6: 2D illustration of the modeling of cuts in a linked volume representation. The object is discretized by means of an adaptive octree grid (shaded cells). The cells of this grid are connected by links (green, solid). Cutting is modeled by disconnecting links (red, dashed). A surface mesh (black line and dots) is reconstructed from the dual grid of links.

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 6), 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. 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

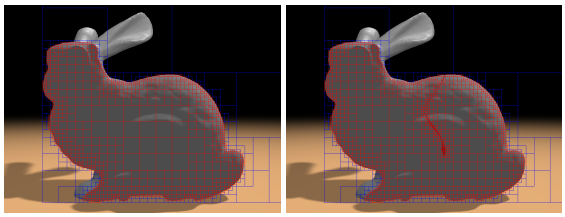

Figure 7: The Stanford bunny model is discretized into a linked octree grid (left), which is refined along the surface and the cuts (right).

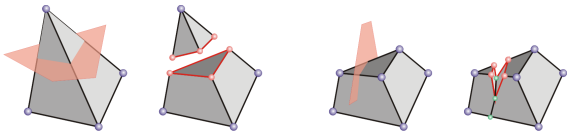

Figure 8: Illustration of cuts in polyhedral elements. Left: A tetrahedron is cut into two parts, resulting in a small tetrahedron and a triangular prism. Right: The triangular prism is partially cut, resulting in two polyhedral elements that are partially connected. Contrary to a tetrahedral discretization, no further subdivision is required.

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 7 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, 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.

In the discussed approaches, the resulting surface is reconstructed from the underlying hexahedral grid. This design thus avoids the explicit cutting of the surface mesh. It has also been adopted for fracturing simulation [HJST13]. A different strategy is to explicitly cut the surface mesh, separated from the hexahedral simulation grid. This strategy was followed by Seiler et al. for the simulation of punching operations [SSSH11], which are commonly seen in endoscopic surgery.

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 8 (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 8 (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. Another solution to handle ill-shaped elements is to treat them separately with an alternative numerical strategy, for example with a geometric deformation model. This was studied by Fierz et al. [FSHH12].

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 in computer graphics is for example surveyed in [NMK*06]. For an introduction of finite elements for elasticity let us refer to Appendix A.2.

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 9 (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

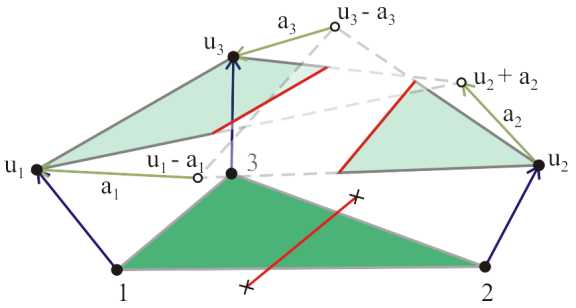

Figure 9: A discontinuous displacement field computed with the extended finite element method. The green triangle domain is divided by a red cut line. The displacements u_i and a_i correspond to the original and added DOFs respectively. Using the shifted Heaviside function as enrichment functions, the 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.

element stiffness matrix is computed as

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

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

$$^{\mathrm{x}}B^e = \left(B^e_1,\, \ldots,\, B^e_{n_v},\, \psi^e_1 B^e_1,\, \ldots,\, \psi^e_{n_v} B^e_{n_v}\right). \tag{5}$$

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. employed 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 10.

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.

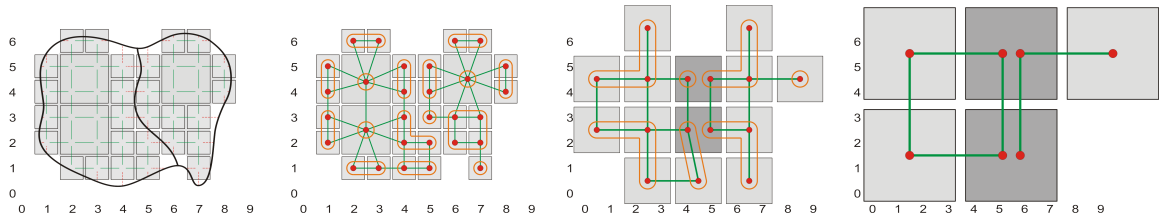

Figure 10: Hierarchical construction of the composite finite element model. Left: 2D illustration of the adaptive linked volume representation, consisting of a set of octree cells (gray), which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary 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.

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. approx-

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

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. From computational mechanics, reviews of meshfree methods for cutting and fracturing can be found in [NRBD08, RBZ10]. 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], the basics of which are summarized in Appendix A.3. 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.

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 11 (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 11 (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 11 (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 11 (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].

While meshfree methods were intentionally proposed to efficiently handle large deformations and topological changes, more recent works has demonstrated performance gains by augmenting meshfree methods with explicitly stored connectivity information. For example, a graph connecting simulation nodes can be employed to evaluate material distance [SOG06] or encode visibility [JL12]. In these two approaches, cutting is modeled by removing corresponding edges in the graph. Another example is the embedding of a uniform hexahedral grid into a meshfree simulation in order to construct the newly exposed cutting surfaces [PGCS09].

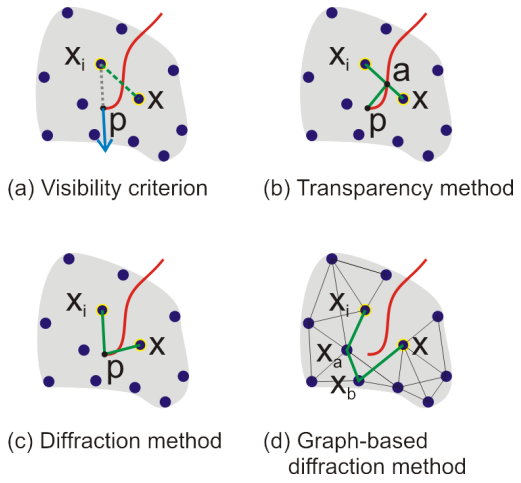

Figure 11: Discontinuity modeling in meshfree methods. The object (gray region) is sampled at a set of simulation nodes (blue dots). (a) The visibility criterion assigns a zero value to the shape function since $\overline{x_i x}$ intersects the cut (the red curve). (b) The transparency method enhances 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. Numerical Solvers

To solve the sparse linear system of equations $Ax = b$ 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 topological changes, the initialization of the solver has to be performed in every simulation step, too.

5.1. Direct Solvers

Direct methods determine an exact solution of the linear system of equations by a finite sequence of operations. Targeting static finite element simulations using linear elasticity without corotational strain (i.e., the matrix A changes only if cuts are applied), Zhong et al. [ZWP05] proposed to solve the equation system by pre-computing the inverse A^{-1}. In their approach, a cut is modeled by deleting elements, and by adapting the rows and columns of A corresponding to

their incident vertices. The manipulation of A is expressed in the form $A + UV^{\mathrm{T}}$, so that the inverse can be updated via the Sherman-Morrison-Woodbury formula [Hag89]

$$(A + UV^{\mathrm{T}})^{-1} = A^{-1} - A^{-1}U(I + V^{\mathrm{T}}A^{-1}U)^{-1}V^{\mathrm{T}}A^{-1}. \quad (16)$$

If the number of affected elements and the number of non-zero entries in the right-hand side vector b are limited by a given constant, the update of A^{-1} and the evaluation of $x = A^{-1}b$ have linear run-time in the number of vertices. Lee et al. [LPO10] made use of condensation [BNC96, WH05] to further accelerate the inversion process. By considering only the surface vertices of the volumetric object in the computation of the inverse, significantly improved computation times can be achieved.

In a similar manner, Lindblad and Turkiyyah updated the inverse of the stiffness matrix in the extended finite element method [LT07]. Using XFEM, the dimension of A increases due to the newly assigned DOFs (see Eq. 6), and A changes to $\begin{bmatrix} A & A^{ua} \\ A^{au} & A^{aa} \end{bmatrix}$, where the superscripts ^{u} and ^{a} correspond to the original and the added DOFs, respectively. The inverse of the new matrix can be computed from the inverse of the original matrix using

$$\begin{bmatrix} A & A^{ua} \\ A^{au} & A^{aa} \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}A^{ua}D^{-1}A^{au}A^{-1} & -A^{-1}A^{ua}D^{-1} \\ -D^{-1}A^{au}A^{-1} & D^{-1} \end{bmatrix}, \quad (17)$$

where $D = A^{aa} - A^{au}A^{-1}A^{ua}$.

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].

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 generate a sequence of increasingly more accurate approximations to the exact solution of a system of equations, and thus are an effective means to balance speed and accuracy by choosing a fixed number of iterations or by specifying a stopping criterion in terms of a threshold for the error reduction. When rating the efficiency of an iterative solver, the major criterion is the achieved convergence rate, i.e., error reduction per computing time.

Nienhuys and van der Stappen [NFvdS00, NFvdS01] used a conjugate gradient (CG) solver [She94], which requires large number of iterations to obtain stable and visually realistic deformations [CJA*10, CAR*09]. On the other hand, since CG solvers involve matrix-vector and vector-vector products they 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 employ a hierarchy of successively coarser grids, such that successively lower frequency error components can be effectively relaxed on successively coarser grids. Multigrid 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.

In multigrid methods, In addition to the relaxation scheme and the coarse grid hierarchy, transfer operators are required to transfer quantities between the grids. Consider a two-level geometric hierarchy where the fine and coarse level are denoted by superscripts h and $2h$, respectively. The linear system on the fine grid has the form $A^h x^h = b^h$. On the coarse grid, the system matrix is approximated by $A^{2h} = R_h^{2h} A^h I_{2h}^h$, where I_{2h}^h is the interpolation operator and $R_h^{2h} = (I_{2h}^h)^{\mathrm{T}}$ the restriction operator. In each iteration, the solver performs the following steps ($\tilde{x}^h$ denotes the current approximate solution):

1. Relax $A^h \tilde{x}^h \approx b^h$,
2. Compute residual $r^h = b^h - A^h \tilde{x}^h$,
3. Restrict residual to the coarse grid: $r^{2h} = R_h^{2h} r^h$,
4. Solve residual equation on the coarse grid: $A^{2h} e^{2h} = r^{2h}$,
5. Interpolate error to the fine grid: $\tilde{e}^h = I_{2h}^h e^{2h}$,
6. Apply correction to the solution: $\tilde{x}^h = \tilde{x}^h + \tilde{e}^h$,
7. Relax $A^h \tilde{x}^h \approx b^h$.

Pre- and post-smoothing (steps 1 and 7) usually involves one or two relaxation iterations. Applying the two-grid method recursively for step 4 leads to a multigrid V-cycle: The relaxation is performed on ever coarser grids $2h$, $4h$, $8h$, ... On the coarsest grid, for step 4 a conjugate gradient solver or a direct solver can be used. In order to adequately represent cuts on the coarser levels, the grid hierarchy can be constructed in a way similar to the hierarchical construction of composite finite elements (see Figure 10).

6. Summary 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

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
Zhong et al. [ZWP05]	Tet., deletion	FEM	Static (direct solver)	Interactive	Static FEM with a direct solver
Wu & Heng [WH05]	Tet., refinement	FEM	Static (CG + direct solver)	Interactive	CG + direct solver
Steinemann et al. [SHGS06]	Tet., refinement + snapping	Mass-spring	Explicit	Interactive (Fig. 16 a)	Refinement + snapping
Chentanez et al. [CAR*09]	Tet., refinement	FEM	Implicit (CG solver)	Interactive (Fig. 16 d)	Needle insertion
Lee et al. [LPO10]	Tet., deletion	FEM	Static (direct solver)	Interactive	Direct solver + condensation
Courtecuisse et al. [CJA*10, CAK*14]	Tet., deletion/refinement	FEM	Implicit (CG solver)	Interactive (Fig. 16 c,e)	Surgery applications
Li et al. [LZW*14]	Tet., refinement	FEM	Explicit	Interactive	Volumetric images
Molino et al. [MBF04]	Tet., duplication	FEM	Mixed explicit/implicit	Offline	Basic virtual node algorithm
Sifakis et al. [SDF07]	Tet., duplication	FEM		Offline (Fig. 15 a)	Arbitrary cutting
Wang et al. [WJST14]	Tet., duplication	FEM		Offline	Redevelopment
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. 15 c)	Enrichment textures
Frisken-Gibson [FG99]	Hex., deletion	ChainMail	Local relaxation	Interactive	Linked volume
Jeřábková et al. [JBB*10]	Hex., deletion	CFEM	Implicit	Interactive	CFEM
Dick et al. [DGW11a]	Hex., refinement	FEM	Implicit (multigrid)	Offline/Interactive (Fig. 15 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. 16 b, f)	Residual stress, collision
Wicke et al. [WBG07]	Poly., splitting	PFEM	Implicit	Offline (Fig. 15 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, graph-based diffraction	Meshfree		Offline/Interactive (Fig. 15 e)	Splitting fronts propagation
Pietroni et al. [PGCS09]	Particles, extended visibility	Meshfree		Interactive	Splitting cubes algorithm
Jung & Lee [JL12]	Particles, connectivity graph	Meshfree	Semi-implicit	Interactive	Dynamic BVHs

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

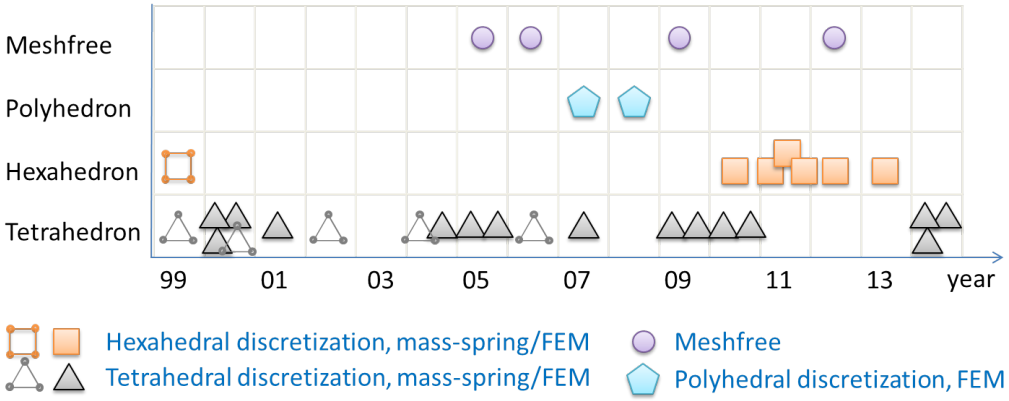

Figure 12: Plot of publications of techniques summarized in Table 1 in chronological order.

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.

Note that considering the specified discretization, in the approaches using CFEs, duplication of elements is used on the composite element level, but initially the topological discontinuity is represented by element deletion [JBB*10] or link disconnection after element refinement [WDW11] on the finest level.

Figure 12 depicts the chronological appearance of the discussed techniques with respect to the underlying discretization and physical model. From this it can be observed that the majority of publications in the field address mesh-based approaches, and that there seems to be a clear trend towards physically accurate simulations using finite elements. Whereas approaches based on a tetrahedral discretization are constantly used from the beginning, approaches using a hexahedral discretization have emerged more recently.

We show representative simulation results for each group in Figures 15 and 16. Figure 15 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 octree grid, and (e) the meshfree method. Figure 16 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 (including a realistic texturing of the cutting surfaces [BZH*05]), (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.

7. Collision Handling and Haptic Rendering

Besides the simulation of deformable bodies, collision handling as well as the haptic rendering of cutting are two important issues in a virtual cutting system. In the following, we briefly cover these topics with the intention to expose challenges of cutting-related research problems. Collision response between separated bodies is analogous to standard deformable body simulation without cutting, i.e., collisions can be resolved by penalty-based methods (i.e., by introducing repulsion forces that are scaled according to penetration depth or penetration volume), or by constraint-based methods (i.e., by enforcing non-penetration by solving a linear complementarity problem).

7.1. Collision Detection

Collision detection for general deformable bodies has been widely studied and an excellent survey is given by Teschner et al. [TKH*05]. These techniques are primarily designed for objects with fixed topology, and most of these techniques need an intensive pre-process to build acceleration data structures. Although, in principle, these methods can also be applied in the context of virtual cutting, the re-initialization of the acceleration structures when topological changes are applied strongly limits their usability. In this section, we discuss the special requirements on collision detection approaches in virtual cutting scenarios, and discuss methods specially designed to meet these requirements.

In virtual cutting simulation, new volumetric elements are created on-the-fly, and new surfaces are exposed. To handle these dynamically created geometric primitives, it is necessary to rebuild or update acceleration data structures such as boundary volume hierarchies. Moreover, as a result of cutting, an object may be incrementally split into several separated objects. It is therefore necessary to consistently detect both inter- and intra-collisions. Consequently, ideal solutions for collision detection for virtual cutting are methods that do not rely on heavy pre-computation, detect both self-collisions and collisions between different bodies, and provide a quantitative measure of the penetration for robust collision response.

In several simulators using the SOFA framework [FDD*12] (e.g., [JBB*10, CAK*14]), collision detection is performed by using layered depth images (LDIs) [HZLM01, HTG04, FBAF08]. LDIs do not require preprocessing of surface meshes, and can be efficiently generated in each simulation step by parallel rasterization on the GPU. LDIs sample a closed manifold object by casting a set of parallel rays and enumerating the intersections between each ray and the object. Along each ray, the line segment from an odd intersection (entering the object) to an even intersection (leaving the object) is considered as part of the object. By comparing the LDIs of two objects, the intersection volume and its gradient can be computed and employed in collision response. For detecting self-collisions, the intersections are classified as entering and leaving based on the angle between the forward ray and the surface normal at the intersection point. One open question of LDIs is the representation of thin objects, such as surgical scalpels. Since LDIs do not support non-closed manifold meshes, it would be necessary to use rasterization at extremely high-resolution in order to sufficiently represent thin features.

Wu et al. [WDW13] proposed an efficient collision detection algorithm particularly tailored to composite finite element simulation of cuts. In the broad phase, potentially colliding pairs of a deformed volumetric element and a displaced surface vertex are identified by using a spatial hashing approach [THM*03]. All surface vertices in the simulation environment, including vertices of a thin scalpel, are treated in a uniform way. For each potentially colliding element/vertex pair, the surface vertex is back-transformed to its position in the reference configuration using the interpolation weights of the vertex with respect to the volumetric element. The penetration at this position is evaluated from a distance field in the reference configuration (which is locally updated during cutting), and forward-transformed to the deformed configuration for collision response. It was shown that by checking the coarse composite elements, rather than

the underlying fine hexahedral elements, a significant performance gain can be obtained. The results also demonstrate that smooth collision response can be achieved for deformable bodies using a hexahedral discretization, despite of the presence of staircase boundaries.

Bounding volume hierarchies (BVHs) are an efficient data structure to accelerate collision detection. However, the tightness of bounding volumes and the culling efficiency degrade significantly in case of large deformations and topological changes. Especially in the context of fracture simulation, several methods have been proposed to optimize the reconstruction/updating of BVHs. To efficiently insert/delete geometric primitives, Otaduy et al. presented a method to dynamically reconstruct BVHs, as opposed to reconstructing them from scratch [OCSG07]. The hierarchies are balanced by simple local operations for progressive fractures. BVHs for large scale fracture simulation was studied in [HSK*10]. Recently, Glondu et al. [GSM*12] demonstrated real-time brittle fracture animations based on a combination of locally updated distance fields and sphere trees for adaptive collision detection, together with an approximation of modal analysis for fracture simulation [GMD13].

Jung et al. proposed a method to reconstruct BVHs for meshfree simulation of cuts, where an undirected graph is maintained to encode the connectivity of simulation nodes [JL12]. The BVH reconstruction is triggered by the event that an object is completely excised into two pieces. This event is detected by examining the node connectivity: In a breadth-first traversal starting from an arbitrary node, if at least one node of the object is not visited during the traversal, the excision event is reported.

7.2. Haptic Rendering of Cutting

Haptics provides an intuitive interface to interact with the simulated deformable bodies and conveys rich information about the dynamics directly to the user. This is especially useful in medical simulations [CMJ11]. While haptics has been mentioned in many cutting simulators, few provide a sufficient description on the implementation and evaluation. Realistic haptic rendering of cutting is a challenging task due to the required high update rates of haptic rendering and the complex physical interaction between the cutting tool and soft tissues.

First, haptic rendering requires update rates of 1 kHz or higher, in order to achieve (perceived) smooth force feedback [SCB04]. It was reported that users can perceive differences at update rates between 500 Hz and 1 kHz [BOYBJK90]. More fundamentally, the update rate is closely related to the stability of the haptic system, i.e., a human-in-the-loop system consisting of the user, the (digitally controlled) haptic device, and the (time-discrete) virtual environment. The latency inherent in time-discrete systems can lead to unstable behaviors (e.g., vibrations). We refer to the haptics literature [CGSS93, CB94] for a detailed explanation. The sufficient condition of passivity and thus stability of the haptic device for a viscoelastic virtual wall (the simplest virtual environment) is given in [CS97] as

$$\Delta T < \frac{2(b-B)}{K}, \tag{18}$$

where ΔT is the sampling period, b is the inherent damping of the device, and K and B are the stiffness and damping of the virtual wall, respectively. This condition implies that a high update rate is very necessary to simulate the interaction with stiff virtual objects.

Equation 18 states the stability for haptic interaction where the cutting tool is simply represented by a single point. For a general cutting tool which may simultaneously contact the deformable body at several locations, it is necessary to modulate the overall contact stiffness, considering the limited impedance offered by haptic devices. To this end, rather than directly mapping the position of the virtual tool from the position signal read from the haptic device, it is customary to separate these two positions, and virtually couple them by a spring (and a damper). This virtual coupling approach results in the simulation-based haptic rendering: The movement of the virtual tool is driven by the coupling force, which tries to align the virtual tool with the haptic stylus, and the interaction force (e.g., the cutting force) between the virtual tool and deformable bodies. The coupling force, rather than the interaction force is sent to the haptic device. In this way the stability of the device can be easily ensured by tuning the coupler [MPT99]. It also enables that the haptic simulation and the deformation/cutting simulation run at different update rates. We refer to a recent survey on haptic rendering [OGL13] for a detailed explanation of different rendering paradigms. The virtual coupling scheme is successfully employed in bone drilling [WWWZ10] and in soft tissue cutting [WWD14] to compute feedback forces which can be rendered stably.

Second, the physical cutting mechanism (i.e., the fracturing of soft tissues driven by cutting tools) is largely unknown, especially considering the complex material properties and various cutting tools such as needles, blades, scissors, or punchers. 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 cutting approaches so far are purely geometry-based: The tissues are cut by geometric intersection tests, as discussed in Section 2.1. 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,

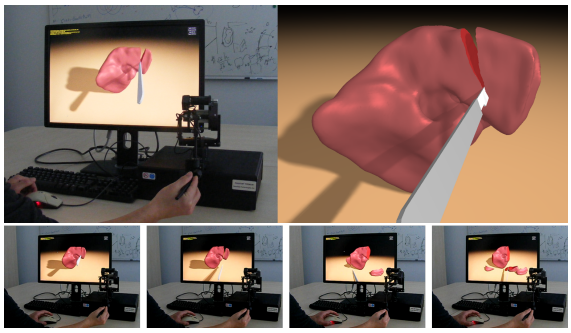

Figure 13: Left: Experiment setup. To cut a liver model the user manipulates a haptic device that is mapped to a scalpel. Right: High quality surface rendering. Bottom: A sequence of images from a live recording, available at http://wwwcg.in.tum.de/research/research/projects/real-time-haptic-cutting.html.

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].

An approach to obtain an intuitive feedback force is to employ a velocity-proportional force model [WWD14]. Intuitively, if the user moves the scalpel with a high speed against the deformable object, (s)he feels a large resistant force. The force direction is opposite to the direction of movement, and the force magnitude is proportional to both the speed of movement and the contact volume between the scalpel and the deformable object. Another possible solution is the data-driven approach [HKSH09].

8. 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 13 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. 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].

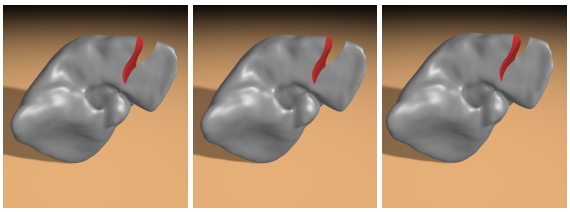

Figure 14: From left to right: Simulation of cuts using finite elements on a uniform hexahedral grid, finite elements on an adaptive octree grid, and composite finite elements on an adaptive octree grid.

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. 14 (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

	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.

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 14 (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. 14 (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.

9. 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 and patient-specific material is becoming of ever increasing importance. Going beyond the model of linear elasticity and homogeneous material, soft tissues exhibiting non-linear, 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.

Interactive simulation frame rates can also be achieved by model reduction techniques, as having been demonstrated in a number of engineering [NACC08] and also graphics applications [BJ05]. The idea is to carefully approximate a large system of equations with a much smaller system (i.e., to reduce the number of DOFs), without significantly sacrificing accuracy. A major difficulty of applying these techniques in the context of interactive cutting scenarios is the fact that determining the reduced system is typically very compute-intensive, so that in practice at least the cutting zone is non-reducible and must be tackled fully, without reduction. A possible direction thus is to couple model reduction for reducible zones and full simulation for non-reducible zones [KGRB13, NAG*12].

Another direction of research is the development of approaches for modeling the physical interaction between a scalpel and soft tissues accurately [MH01, CDL07, MRO08]. This can greatly contribute to the realistic haptic rendering of cutting forces. 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.

The research on interactive virtual cutting is not limited to the computer graphics community. In computational mechanics, where research on cutting and fracturing was initially more concerned with simulation accuracy (e.g. accurate material models, accurate boundary conditions [MBB*11], accurate cutting force models [MMSE11], goal-oriented error estimates [GENR*14]), there is a recent trend towards interactive surgery simulations [NAG*12, JJMW13]. It will be interesting to investigate how such models can be integrated into efficient simulators. We envisage that cross-fertilization with computational mechanics will further advance virtual cutting simulation towards its application in pre-operative planning and surgery training.

Acknowledgments

Jun Wu was supported by a scholarship from the Erasmus Mundus TANDEM, an European Commission funded program. This work was additionally supported by the European Union under the ERC Advanced Grant 291372 SaferVis – Uncertainty Visualization for Reliable Data Discovery.

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. 19

[Bat96] Bathe K.-J.: *Finite Element Procedures*. Prentice Hall, 1996. 24

[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), IEEE Computer Society Press, pp. 116–442. 4, 5, 14

[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. 4, 5, 14

[BJ05] Barbič J., James D. L.: Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. 24*, 3 (July 2005), 982–990. 19

[BLG94] Belytschko T., Lu Y. Y., Gu L.: Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering 37*, 2 (1994), 229–256. 11

[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, 5, 14

[BN98] Bro-Nielsen M.: Finite element modeling in surgery simulation. *Proceedings of the IEEE 86*, 3 (1998), 490–503. 24

[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. 13

[BOYBJK90] Brooks Jr. F. P., Ouh-Young M., Batter J. J., Jerome Kilpatrick P.: Project gropehaptic displays for scientific visualization. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 177–185. 16

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

[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, 14

[BZH*05] Bachofen D., Zátonyi J., Harders M., Székely G., Fruh P., Thaler M.: Enhancing the visual realism of hysteroscopy simulation. *Studies in health technology and informatics 119* (2005), 31. 15

[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. 14, 15, 27

[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, 13, 14, 27

[CB94] Colgate J. E., Brown J. M.: Factors affecting the z-width of a haptic display. In *IEEE International Conference on Robotics and Automation* (1994), IEEE, pp. 3205–3210. 16

[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, 4, 14

[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. 16, 19

[CGSS93] Colgate J. E., Grafing P. E., Stanley M. C., Schenkel G.: Implementation of stiff virtual walls in force-reflecting interfaces. In *IEEE Virtual Reality Annual International Symposium* (1993), IEEE, pp. 202–208. 16

[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. 13, 14, 19, 27

[CMJ11] Coles T., Meglan D., John N.: The role of haptics in medical training simulators: A survey of the state of the art. *IEEE Transactions on Haptics 4*, 1 (Jan 2011), 51–66. 16

[CS97] Colgate J. E., Schenkel G. G.: Passivity of a class of sampled-data systems: Application to haptic interfaces. *Journal of robotic systems 14*, 1 (1997), 37–47. 16

[DG95] Desbrun M., Gascuel M.-P.: Animating soft substances with implicit surfaces. In *Proceedings of SIGGRAPH* (1995), ACM, 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), Eurographics Association, 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, 3, 6, 7, 9, 13, 14, 17, 27

[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. 19

[ED08] Eisemann E., Décoret X.: Single-pass gpu solid voxelization for real-time applications. In *Proceedings of Graphics Interface 2008* (Toronto, Ont., Canada, Canada, 2008), GI '08, Canadian Information Processing Society, pp. 73–80. 6

[FBAF08] Faure F., Barbier S., Allard J., Falipou F.: Image-based collision detection and response between arbitrary volume objects. In *SCA '08: Symposium on Computer Animation* (2008), Eurographics Association, pp. 155–162. 15

[FDD*12] Faure F., Duriez C., Delingette H., Allard J., Gilles B., Marchesseau S., Talbot H., Courtecuisse H., Bousquet G., Peterlik I., et al.: SOFA: A multi-model framework for interactive physical simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*. Springer, 2012, pp. 283–321. 15

[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, 14

[FM03] Fries T.-P., Matthies H. G.: Classification and overview of meshfree methods. *Technical University of Braunschweig* (2003). 11, 25

[FSHH12] Fierz B., Spillmann J., Hoyos I., Harders M.: Maintaining large time steps in explicit finite element simulations using shape matching. *Visualization and Computer Graphics, IEEE Transactions on 18*, 5 (May 2012), 717–728. 8

[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. 4, 5

[GENR*14] González-Estrada O., Nadal E., Ródenas J., Kerfriden P., Bordas S., Fuenmayor F.: Mesh adaptivity driven by goal-oriented locally equilibrated superconvergent patch recovery. *Computational Mechanics 53*, 5 (2014), 957–976. 19

[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

[GMD13] Glondu L., Marchal M., Dumont G.: Real-time simulation of brittle fracture using modal analysis. *IEEE Transactions on Visualization and Computer Graphics 19*, 2 (Feb. 2013), 201–209. 16

[GSM*12] Glondu L., Schvartzman S., Marchal M., Dumont G., Otaduy M.: Efficient collision detection for brittle fracture. In *SCA '12: Symposium on Computer Animation* (2012), Eurographics Association, pp. 285–294. 16

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

[GW08] Georgii J., Westermann R.: Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation* (2008), EG, pp. 11–19. 25

[Hag89] Hager W. W.: Updating the inverse of a matrix. *SIAM Review 31*, 2 (1989), 221–239. 13

[HJST13] Hegemann J., Jiang C., Schroeder C., Teran J. M.: A level set method for ductile fracture. In *SCA '13: Symposium on Computer Animation* (2013), ACM, pp. 193–201. 7

[HKSH09] Hover R., Kosa G., Szekely G., Harders M.: Data-driven haptic rendering: From viscous fluids to visco-elastic solids. *Haptics, IEEE Transactions on 2*, 1 (Jan 2009), 15–27. 17

[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. 13

[HS97] Hackbusch W., Sauter S.: Composite finite elements for the approximation of PDEs on domains with complicated micro-structures. *Numerische Mathematik 75*, 4 (1997), 447–472. 9

[HSK*10] Heo J.-P., Seong J.-K., Kim D., Otaduy M. A., Hong J.-M., Tang M., Yoon S.-E.: Fastcd: fracturing-aware stable collision detection. In *SCA '10: Symposium on Computer Animation* (2010), Eurographics Association, pp. 149–158. 16

[HTG04] Heidelberger B., Teschner M., Gross M.: Detection of collisions and self-collisions using image-space techniques. *Journal of WSCG 12*, 3 (2004), 145–152. 15

[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. 19

[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. 19

[HZLM01] Hoff III K. E., Zaferakis A., Lin M., Manocha D.: Fast and simple 2d geometric proximity queries using graphics hardware. In *I3D '01: Symposium on Interactive 3D graphics* (2001), ACM, pp. 145–148. 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, 6, 9, 14, 15, 17

[JJMW13] Jin X., Joldes G., Miller K., Wittek A.: 3D algorithm for simulation of soft tissue cutting. In *Computational Biomechanics for Medicine*, Wittek A., Miller K., Nielsen P. M., (Eds.). 2013, pp. 49–62. 19

[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, 9, 14, 18

[JL12] Jung H., Lee D. Y.: Real-time cutting simulation of meshless deformable object using dynamic bounding volume hierarchy. *Computer Animation and Virtual Worlds 23*, 5 (2012), 489–501. 12, 14, 16

[JLSW02] Ju T., Losasso F., Schaefer S., Warren J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (2002), 339–346. 7

[KGRB13] Kerfriden P., Goury O., Rabczuk T., Bordas S.: A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. *Computer Methods in Applied Mechanics and Engineering 256*, 0 (2013), 169–188. 19

[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, 9, 14, 18, 27

[KMBG08] Kaufmann P., Martin S., Botsch M., Gross M.: Flexible simulation of deformable models using discontinuous galerkin FEM. In *SCA '08: Symposium on Computer Animation* (2008), Eurographics Association, 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. 18

[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. 4

[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. 13, 14

[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. 9

[LS81] Lancaster P., Salkauskas K.: Surfaces generated by moving least squares methods. *Mathematics of Computation 37*, 155 (1981), 141–158. 11, 25

[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), ACM, pp. 421–429. 4, 8, 13

[LZW*14] Li S., Zhao Q., Wang S., Hao A., Qin H.: Interactive deformation and cutting simulation directly using patient-specific volumetric images. *Computer Animation and Virtual Worlds 25*, 2 (2014), 155–169. 4, 14

[MBB*11] Moumnassi M., Belouettar S., Béchet É., Bordas S. P., Quoirin D., Potier-Ferry M.: Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering 200*, 5–8 (2011), 774 – 796. 19

[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, 4, 6, 11, 14

[MBP14] Muguercia L., Bosch C., Patow G.: Fracture modeling in computer graphics. *Computers & Graphics* (2014). 2

[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

[MDM*02] Müller M., Dorsey J., McMillan L., Jagnow R., Cutler B.: Stable real-time deformations. In *SCA '02: Symposium on Computer Animation* (2002), ACM, pp. 49–54. 25

[MG04] Müller M., Gross M.: Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), Canadian Human-Computer Communications Society, pp. 239–246. 4

[MH01] Mahvash M., Hayward V.: Haptic rendering of cutting: A fracture mechanics approach. *Haptics-e 2*, 3 (2001), 1–12. 19

[MK00] Mor A. B., Kanade T.: Modifying soft tissue models: Progressive cutting with minimal new element creation. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention* (2000), Springer-Verlag, pp. 598–607. 2, 4, 5, 14

[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, 14

[MKN*04] Müller M., Keiser R., Nealen A., Pauly M., Gross M., Alexa M.: Point based animation of elastic, plastic and melting objects. In *SCA '04: Symposium on Computer animation* (2004), Eurographics Association, pp. 141–151. 2, 11, 25, 26

[MMSE11] Moore J. Z., Malukhin K., Shih A. J., Ehmann K. F.: Hollow needle tissue insertion force model. *{CIRP} Annals - Manufacturing Technology 60*, 1 (2011), 157 – 160. 19

[MPT99] McNeely W. A., Puterbaugh K. D., Troy J. J.: Six degree-of-freedom haptic rendering using voxel sampling. In *Proceedings of SIGGRAPH* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 401–408. 16

[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. 19

[NACC08] Niroomandi S., Alfaro I., Cueto E., Chinesta F.: Real-time deformable models of non-linear tissues by model reduction techniques. *Computer Methods and Programs in Biomedicine 91*, 3 (2008), 223–231. 19

[NAG*12] Niroomandi S., Alfaro I., González D., Cueto E., Chinesta F.: Real-time simulation of surgery by reduced-order modeling and X-FEM techniques. *International Journal for Numerical Methods in Biomedical Engineering 28*, 5 (2012), 574–588. 19

[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, 13, 14

[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. 4, 13, 14

[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. 8

[NRBD08] Nguyen V. P., Rabczuk T., Bordas S., Duflot M.: Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation 79*, 3 (2008), 763 – 813. 11

[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. *ACM Trans. Graph. 21*, 3 (July 2002), 291–294. 2

[OCSG07] Otaduy M., Chassot O., Steinemann D., Gross M.: Balanced hierarchies for collision detection between fracturing objects. In *VR '07: IEEE Virtual Reality Conference* (march 2007), IEEE, pp. 83 –90. 16

[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

[OGL13] Otaduy M., Garre C., Lin M.: Representations and algorithms for force-feedback display. *Proceedings of the IEEE 101*, 9 (Sept 2013), 2068–2080. 16

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

[PCOS10] Pietroni N., Cignoni P., Otaduy M., Scopigno R.: Solid-texture synthesis: A survey. *Computer Graphics and Applications, IEEE 30*, 4 (July 2010), 74–89. 3

[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, 7, 11, 12, 14

[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, 14, 26

[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), University of Ulm, pp. 52–66. 9

[RB86] Rankin C., Brogan F.: An element independent corotational procedure for the treatment of large rotations. *Journal of pressure vessel technology 108*, 2 (1986), 165–174. 25

[RBZ10] Rabczuk T., Bordas S., Zi G.: On three-dimensional modelling of crack growth using partition of unity methods. *Computers & Structures 88*, 23-24 (2010), 1391–1411. 11

[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

[SCB04] Salisbury K., Conti F., Barbagli F.: Haptic rendering: introductory concepts. *IEEE Computer Graphics and Applications 24*, 2 (2004), 24–32. 16

[SDF07] Sifakis E., Der K. G., Fedkiw R.: Arbitrary cutting of deformable tetrahedralized objects. In *SCA '07: Symposium on Computer animation* (2007), Eurographics Association, pp. 73–80. 3, 4, 6, 14, 27

[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. 19

[She94] Shewchuk J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep., Carnegie Mellon University, 1994. 13

[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. 4

[SHGS06] Steinemann D., Harders M., Gross M., Szekely G.: Hybrid cutting of deformable solids. In *VR '06: IEEE Virtual Reality Conference* (2006), IEEE, pp. 35–42. 4, 5, 14, 27

[Si06] Si H.: *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*, 2006. http://tetgen.org. 4

[Sla02] Slaughter W. S.: *The linearized theory of elasticity*. Springer, 2002. 24

[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 *SCA '06: Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 63–72. 2, 11, 12, 14, 27

[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, 7, 14

[SW06] Sauter S., Warnke R.: Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing 77*, 1 (2006), 29–55. 9

[THM*03] Teschner M., Heidelberger B., Müller M., Pomerantes D., Gross M. H.: Optimized spatial hashing for collision detection of deformable objects. In *VMV '03: Proceedings of the Vision, Modeling, and Visualization Conference* (2003), Aka GmbH, pp. 47–54. 15

[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. 13, 14

[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, 15

[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. 4

[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, 14, 27

[WBWD12] Wu J., Bürger K., Westermann R., Dick C.: Interactive residual stress modeling for soft tissue simulation. In *VCBM '12: Proceedings of Eurographics Workshop on Visual Computing for Biology and Medicine* (2012), Eurographics Association, pp. 81–89. 3, 27

[WDW11] Wu J., Dick C., Westermann R.: Interactive high-resolution boundary surfaces for deformable bodies with changing topology. In *VRIPHYS '11: Proceedings of 8th Workshop on Virtual Reality Interaction and Physical Simulation* (2011), Eurographics Association, pp. 29–38. 2, 3, 6, 7, 9, 14, 18

[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. 3, 14, 15, 17

[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. 13, 14

[WJST14] Wang Y., Jiang C., Schroeder C., Teran J.: An adaptive virtual node algorithm with robust mesh cutting. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2014), The Eurographics Association, pp. 77–85. 6, 14

[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. 16, 17, 27

[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. 16, 17

[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. 4, 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. 12, 14

Appendix A: Physically-based Simulation of Deformable Bodies

In this appendix we briefly summarize the basics of the linear theory of elasticity, the finite element method, meshfree methods, and time integration schemes underlying virtual cutting approaches.

A.1 Elasticity Theory

The theory of elasticity studies how elastic materials deform under external forces. The deformation behavior is mathematically described by a system of partial differential equations.

Deformation Given an elastic object in the undeformed *reference configuration* $\Omega \subset \mathbb{R}^3$, the deformation is modeled by a *displacement vector field* $u : \Omega \to \mathbb{R}^3$, $x \mapsto u(x)$, where x denotes the Euclidean coordinates of a material point in the object's reference configuration, yielding the *deformed configuration* $\{x + u(x) \,|\, x \in \Omega\}$.

A *strain* tensor describes the local (differential) change of shape of the deformable object. In particular, the *Green-St. Venant strain tensor* is defined as

$$\varepsilon_G = \frac{1}{2}\left(\nabla u + (\nabla u)^{\mathrm{T}} + (\nabla u)^{\mathrm{T}} \nabla u\right). \tag{19}$$

ε_G is a symmetric second order tensor, and is non-linear due to its quadratic term $(\nabla u)^{\mathrm{T}} \nabla u$.

Material model A *stress* tensor describes the internal forces acting in the deformable body. For elastic materials, the stress is a function of the strain, referred to as *material model*.

Linear elasticity The *linear* theory of elasticity is based on the assumption of small displacements and a linear relationship between stress and strain. This finally leads to a linear system of equations, which can be solved efficiently. By using a corotational strain formulation which removes the per-element rigid body rotation part from the deformation before

the strain is computed, the linear theory can also be used to accurately simulate deformations exhibiting large rotations. Linear elasticity is therefore widely employed in interactive graphics applications.

By assuming that the displacements are small (i.e., $\|\nabla u\| \ll 1$), the quadratic term in the Green-St. Venant strain tensor ε_G can be neglected, leading to a linear strain tensor, known as the *infinitesimal strain tensor*

$$\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^{\mathrm{T}}\right). \tag{20}$$

The linear relationship between stress and strain is described by

$$\sigma = C : \varepsilon, \tag{21}$$

where σ is the symmetric second-order stress tensor, and C is the fourth-order *elasticity tensor*, representing the material properties. For an *isotropic* material, the deformation is independent of the material's spatial orientation, reducing the stress-strain relationship to the form

$$\sigma = 2\mu\varepsilon + \lambda \operatorname{tr}(\varepsilon) I, \tag{22}$$

where the *Lamé constants* λ and μ are related to the *Young's modulus* E and the *Poisson's ratio* ν by $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$ (I is a 3×3 identity matrix). Rearranging the entries of the symmetric tensors as vectors, the linear material model can be written as matrix-vector product, e.g., for isotropic materials as

$$\underbrace{\begin{pmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{12}\\ \sigma_{13}\\ \sigma_{23}\end{pmatrix}}_{\overline{\sigma}} = \underbrace{\begin{pmatrix}2\mu+\lambda & \lambda & \lambda & & & \\ \lambda & 2\mu+\lambda & \lambda & & & \\ \lambda & \lambda & 2\mu+\lambda & & & \\ & & & \mu & & \\ & & & & \mu & \\ & & & & & \mu\end{pmatrix}}_{\overline{C}} \underbrace{\begin{pmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ 2\varepsilon_{12}\\ 2\varepsilon_{13}\\ 2\varepsilon_{23}\end{pmatrix}}_{\overline{\varepsilon}}. \tag{23}$$

Equations of equilibrium The static elasticity problem consists of determining a displacement vector field $u : \Omega \to \mathbb{R}^3$ such that at each material point the surplus of the internal body forces—expressed by the divergence of the stress tensor field—balances the external body forces f_b according to

$$-\operatorname{div}\sigma(\varepsilon(u)) = f_b \qquad \text{in} \quad \Omega \setminus \partial\Omega, \tag{24}$$

and that the boundary conditions

$$u = u^0 \qquad \text{on} \quad \Gamma_D, \tag{25}$$

$$\sigma(\epsilon(u))\, n = f_s \qquad \text{on} \quad \Gamma_N, \tag{26}$$

on the boundary $\partial\Omega = \Gamma_D \dot{\cup} \Gamma_N$ are satisfied, where n is the unit outer normal on $\partial\Omega$. The boundary conditions consist of Dirichlet boundary conditions (Eq. 25), which prescribe the displacement u^0 on Γ_D, and Neumann boundary conditions (Eq. 26), which prescribe surface tractions f_s on Γ_N.

For the simulation of dynamic motion, inertial forces are incorporated into the equilibrium equation according to

$$\rho\,\ddot{u} - \operatorname{div}\sigma(\varepsilon(u)) = f_b, \tag{27}$$

where ρ is the material density, and $\ddot{u}$ denotes the acceleration. The boundary conditions are analogous to the static case. In addition, initial conditions prescribing the displacement and the velocity of the deformable body at the initial time are required [Sla02].

A.2 The Finite Element Method

The finite element method (FEM) is one of the most popular approaches to solve the systems of partial differential equations arising from elasticity theory. A detailed explanation of finite element procedures for mechanics can be found in textbooks (e.g., [Bat96]), and a concise introduction of FEM in medical simulation is given in [BN98].

Weak formulation of the elasticity problem Multiplying the dynamic equilibrium equation (Eq. 27) with an arbitrary test function v and integrating over the simulation domain Ω leads to the variational formulation

$$\int_\Omega \rho v\cdot\ddot{u}\,\mathrm{d}x + \int_\Omega \epsilon(v) : \sigma(\epsilon(u))\,\mathrm{d}x - \int_\Omega v\cdot f_b\,\mathrm{d}x = 0 \quad \forall v \tag{28}$$

(here, $f_s \equiv 0$ is assumed for simplicity). The formulation for the static elasticity problem can be obtained by omitting the first term corresponding to the inertial forces.

Finite element discretization In the finite element method, the simulation domain Ω is decomposed into a finite set of elements. Typically, triangles or quadrilaterals are employed for discretizing a 2D domain, while tetrahedra or hexahedra are employed in the 3D case.

Building on such a spatial discretization, FEM approximates the continuous displacement field by means of interpolation of the displacements at the vertices of the finite element grid. Specifically, the *element shape functions* $\phi_i^e(x)$ interpolate the displacement field within an element Ω^e from the element's vertices according to

$$u|_{\Omega^e}(x) = \sum_i^{n_v} \phi_i^e(x) u_i^e = \Phi^e(x) u^e. \tag{29}$$

where n_v is the number of simulation nodes of this element, $\Phi^e(x)$ the element shape matrix

$$\Phi^e(x) = \begin{pmatrix}\phi_1^e(x) & & & \phi_{n_v}^e(x) & & \\ & \phi_1^e(x) & & \cdots & \phi_{n_v}^e(x) & \\ & & \phi_1^e(x) & & & \phi_{n_v}^e(x)\end{pmatrix}, \tag{30}$$

and $u^e = (u_1^e, \ldots, u_{n_v}^e)^{\mathrm{T}}$ the linearization of the displacement vectors at the element's vertices.

Linear and trilinear interpolation are good candidates as shape functions for the tetrahedral and hexahedral discretization, respectively. They fulfill the requirements imposed by the finite element method: a) Partition of unity: $\Sigma_i^{n_v}\phi_i^e(x) = 1$;

b) Linear field reproduction: $\Sigma_i^{n_v} \phi_i^e(x) x_i = x$; and c) Kronecker delta function property:

$$\phi_i^e(x_j) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

Condition a) is compulsory, while condition b) and c) are preferable to simplify the treatment of boundaries.

Using the shape functions, and expressing the strain by $\overline{\epsilon}(u)|_{\Omega^e}(x) = B^e(x)u^e$, where B^e is the *element strain matrix*

$$B^e(x) = \begin{pmatrix} \frac{\partial \phi_1^e}{\partial x_1} & & & \cdots & \frac{\partial \phi_{n_v}^e}{\partial x_1} & & \\ & \frac{\partial \phi_1^e}{\partial x_2} & & & & \frac{\partial \phi_{n_v}^e}{\partial x_2} & \\ & & \frac{\partial \phi_1^e}{\partial x_3} & & & & \frac{\partial \phi_{n_v}^e}{\partial x_3} \\ \frac{\partial \phi_1^e(x)}{\partial x_2} & \frac{\partial \phi_1^e(x)}{\partial x_1} & & & \frac{\partial \phi_{n_v}^e(x)}{\partial x_2} & \frac{\partial \phi_{n_v}^e(x)}{\partial x_1} & \\ \frac{\partial \phi_1^e(x)}{\partial x_3} & & \frac{\partial \phi_1^e(x)}{\partial x_1} & & \frac{\partial \phi_{n_v}^e(x)}{\partial x_3} & & \frac{\partial \phi_{n_v}^e(x)}{\partial x_1} \\ & \frac{\partial \phi_1^e(x)}{\partial x_3} & \frac{\partial \phi_1^e(x)}{\partial x_2} & & & \frac{\partial \phi_{n_v}^e(x)}{\partial x_3} & \frac{\partial \phi_{n_v}^e(x)}{\partial x_2} \end{pmatrix}, \tag{32}$$

it can be derived that the weak formulation Eq. 28 leads to a system of ordinary differential equations,

$$\underbrace{\int_{\Omega^e} \rho(\Phi^e)^{\mathrm{T}} \Phi^e \mathrm{d}x}_{M^e} \ddot{u}^e + \underbrace{\int_{\Omega^e} (B^e)^{\mathrm{T}} \overline{C} B^e \mathrm{d}x}_{K^e} u^e = \underbrace{\int_{\Omega^e} (\Phi^e)^{\mathrm{T}} f_b \mathrm{d}x}_{f^e}, \tag{33}$$

(*formally* written for a single element). M^e, K^e, and f^e are called the element mass matrix, the element stiffness matrix, and the element load vector, respectively.

Assembling the per-element equations with respect to the global indices of the shared vertices among the elements yields a system of ordinary differential equations for the entire object, i.e., $M\ddot{u} + Ku = f$. It is common to apply a velocity-dependent Rayleigh damping term which leads to

$$M\ddot{u} + D\dot{u} + Ku = f, \tag{34}$$

where D is calculated as $D = \alpha M + \beta K$. $\alpha, \beta \geq 0$ are the mass and the stiffness proportional damping coefficient. Note that Neumann boundary conditions (prescribing tractions) are naturally incorporated into the weak formulation and do not need further treatment. In contrast, Dirichlet boundary conditions (prescribing displacements for specific vertices) must be explicitly enforced by removing these vertices as DOFs from the finite element formulation. To ensure a consistent treatment of free and fixed vertices, these vertices can be later incorporated into the finally resulting linear system of equations by adding the equation $Iu_i = u_i^0$ for each fixed vertex i.

Corotational strain formulation The infinitesimal strain tensor works under the assumption small displacements. It is not rotation invariant, and interprets rotations as strains, leading to the introduction of artificial stresses. To realistically simulate deformations with large rotations—as they are typically occurring in virtual cutting applications—without resorting to the non-linear Green-St. Venant strain tensor, a rotationally invariant infinitesimal stain formulation [RB86], known as corotational stain formulation, is widely applied in graphics applications [MDM*02]. The idea is to rotate the deformed finite elements back to align with their initial orientation in the reference configuration before the strain is computed. This leads to the equation

$$[R^e]K^e([R^e]^{\mathrm{T}}(x^e + u^e) - x^e) = f^e \tag{35}$$

(for the static case), where $[R^e]$ denotes a $n_v \times n_v$ block diagonal matrix, and each diagonal entry is the element rotation matrix R^e, determined from the current element deformation. The rotation matrix can be computed by minimizing the distance between the rotated deformed and the undeformed configuration [GW08]. Rearranging this formulation for u^e leads to $[R^e]K^e[R^e]^{\mathrm{T}}u^e = f^e + [R^e]K^e(x^e - [R^e]^{\mathrm{T}}x^e)$, which can be incorporated into Eq.34. It should be noted that the corotational strain formulation does not change the structure of the system matrix, however, it is required to update the entries at each simulation time step since the element rotations are determined per time step.

A.3 Meshfree Methods

In contrast to FEMs which operate on a finite element mesh with an explicit connectivity among the nodes, meshfree methods maintain node adjacency by defining an influence region for each node, described by a weight kernel. The weight kernel for the node i located at the position x_i can be defined as, for example [MKN*04],

$$\omega(x, x_i, r_i) = \begin{cases} \frac{315}{64\pi r_i^9}(r_i^2 - \|x_i - x\|^2)^3 & \text{if } \|x_i - x\| < r_i, \\ 0 & \text{otherwise,} \end{cases} \tag{36}$$

where r_i is the influence radius. The value of the weight kernel 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.

Meshfree methods model objects as a set of interacting nodes which carry properties, e.g., mass, volume, and density. The mass m_i carried by a node is initialized as $m_i = s\, r_i^3\, \rho$, where ρ is the material density, and s is a scaling constant for all nodes, chosen such that the estimated density $\rho_i = \Sigma_j m_j \omega(x_j, x_i, r_i)$ is close to ρ. The volume covered by a node is calculated by $v_i = m_i/\rho_i$.

In meshfree methods, the displacement field is approximated as $u(x) = \Sigma_i \phi_i(x) u_i$ from the linearized displacement vectors u_i at a set of (nearby) nodes $\{x_i\}$ and the shape functions $\phi_i(x)$ [FM03]. The meshfree shape functions can be approximated by using, for example, the moving least squares (MLS) scheme [LS81]. MLS is a method to reconstruct continuous functions from a set of sample points based

on their distribution, i.e., the connectivity of samples are not required. Given the weight kernel ω, MLS yields the following shape functions [PKA*05]

$$\phi_i(x) = \omega(x, x_i, r_i) p^{\mathrm{T}}(x) [H(x)]^{-1} p(x_i), \tag{37}$$

where p denotes a complete polynomial basis $p(x) = [1\ x\ ...\ x^n]^{\mathrm{T}}$, and $[H(x)]^{-1}$ is the inverse of the moment matrix defined as

$$H(x) = \Sigma_i \omega(x, x_i, r_i) p(x_i) p^{\mathrm{T}}(x_i). \tag{38}$$

Since the inversion of a matrix is involved in the shape functions, a direct evaluation of their derivative is non-trivial. An alternative is to approximate the gradient ∇u at nodes using a MLS formulation with a linear basis [MKN*04]. Representing the neighbors of the node x_i as $\{x_j\}$, the displacement of neighbor nodes can be approximated by

$$\tilde{u}_j = u_i + \nabla u|_{x_i} x_{ij}, \tag{39}$$

where $x_{ij} = x_j - x_i$, and $\nabla u|_{x_i}$ is the displacement gradient at the node x_i. The sum of the squared differences between the approximated values $\tilde{u}_j$ and the known values u_j is

$$e = \Sigma_j (\tilde{u}_j - u_j)^2 \omega(x_j, x_i, r_i). \tag{40}$$

Minimizing this error function by assigning a zero value to its derivative with respect to u_i, it can be derived that

$$\nabla u|_{x_i} = A^{-1} \left(\Sigma_j (u_j - u_i) x_{ij} \omega(x_j, x_i, r_i) \right), \tag{41}$$

where the matrix $A_{3\times 3} = \Sigma_j x_{ij} x_{ij}^{\mathrm{T}} \omega(x_j, x_i, r_i)$.

With the gradient of the displacement field computed, the strain tensor can be evaluated at the node x_i according to the Green-St. Venant (Eq. 19) or the infinitesimal (Eq. 20) strain formulation. The stress tensor for linear elastic materials can be computed using Eq. 21.

Internal forces can be derived as the negative gradient of the strain energy density with respect to the displacement field,

$$f^{int} = -\nabla_u (\frac{1}{2} \epsilon : \sigma). \tag{42}$$

By integrating this function over the volume v_i covered by the node x_i, it can be derived that this yields the following form of internal forces [MKN*04],

$$f_i^{int} = 2 v_i \left(I + (\nabla u|_{x_i})^{\mathrm{T}} \right) \sigma_i A^{-1} \left(\Sigma_j x_{ij} \omega(x_j, x_i, r_i) \right). \tag{43}$$

The dynamic simulation problem can then be formulated as

$$f_i^{int} + f_i^{ext} - m_i \ddot{u} = 0, \tag{44}$$

where f_i^{ext} denotes the external forces applied to the node x_i.

To animate the boundary surface which is for example represented by a surface mesh, the displacement u of a surface vertex can be computed based on the displacements u_i of its nearby simulation nodes as

$$u(x) = \frac{1}{\Sigma_i \omega(x, x_i, r_i)} \Sigma_i \omega(x, x_i, r_i) \left(u_i + \nabla u|_{x_i} (x - x_i) \right). \tag{45}$$

A.4 Time Integration

There exist implicit and explicit time integration schemes to numerically integrate the time-dependent system of ordinary differential equations arising in dynamic deformation simulations, i.e., Eq. 34. Here, we take the implicit Euler and the explicit Euler time integration scheme as examples.

Implicit time integration allows for using a reasonably large time step size $\mathrm{d}t$ without introducing numerical problems. Using a finite difference discretization of the time derivatives, the implicit Euler time integration leads to

$$M \frac{u^{t+\mathrm{d}t} - 2u^t + u^{t-\mathrm{d}t}}{\mathrm{d}t^2} + D \frac{u^{t+\mathrm{d}t} - u^{t-\mathrm{d}t}}{2\mathrm{d}t} + K u^{t+\mathrm{d}t} = f^{t+\mathrm{d}t}. \tag{46}$$

This equation can be rewritten as

$$\tilde{K} u^{t+\mathrm{d}t} = \tilde{f}^{t+\mathrm{d}t}, \tag{47}$$

with

$$\begin{aligned} \tilde{K} &= K + \tfrac{M}{\mathrm{d}t^2} + \tfrac{D}{2\mathrm{d}t}, \\ \tilde{f}^{t+\mathrm{d}t} &= f^{t+\mathrm{d}t} + M \tfrac{2u^t - u^{t-\mathrm{d}t}}{\mathrm{d}t^2} + D \tfrac{u^{t-\mathrm{d}t}}{2\mathrm{d}t}. \end{aligned} \tag{48}$$

This linear system of equations can be solved using numerical solvers as presented in Section 5.

The explicit (or forward) Euler time integration scheme approximates the elastic force Ku based on the displacement of the previous time step, and thus avoids solving a system of equations. In explicit time integration, the equations of motion are decoupled and each DOF is evaluated independently. It is written as

$$u^{t+\mathrm{d}t} = \left(\frac{M}{\mathrm{d}t^2} + \frac{D}{2\mathrm{d}t} \right)^{-1} \left(f^{t+\mathrm{d}t} - K u^t + M \frac{2u^t - u^{t-\mathrm{d}t}}{\mathrm{d}t^2} + D \frac{u^{t-\mathrm{d}t}}{2\mathrm{d}t} \right). \tag{49}$$

However, the stability of explicit schemes is only guaranteed for sufficiently small time steps, expressed by the *Courant condition*, which gives an upper limit for the time step size according to

$$\mathrm{d}t < h \sqrt{\frac{\rho}{\lambda + 2\mu}}. \tag{50}$$

Here h denotes the smallest distance of two vertices in the reference configuration, ρ is the material density, and λ, μ are the Lamé constants. In general, the stiffer the simulated materials are, the smaller must be the time step size. This implies that a large number of simulation steps are required to advance an interactive simulation. The problem aggravates in the situation of a fine discretization (i.e., a small h). Therefore, from a stability point of view, explicit integration schemes are not well suited for interactive applications which require large time steps (e.g., $10 \sim 100$ ms).

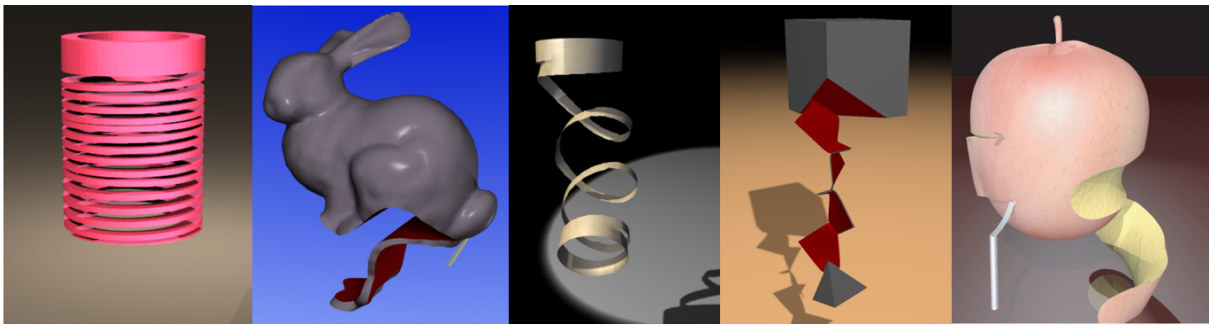

Figure 15: Offline progressive cutting scenarios simulated by (a) the virtual node algorithm on a tetrahedral mesh (image courtesy of Sifakis et al. [SDF07] ©2007 ACM), (b) the polyhedral finite element method (image courtesy of Wicke et al. [WBG07] ©2007 WILEY), (c) the extended finite element method 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.

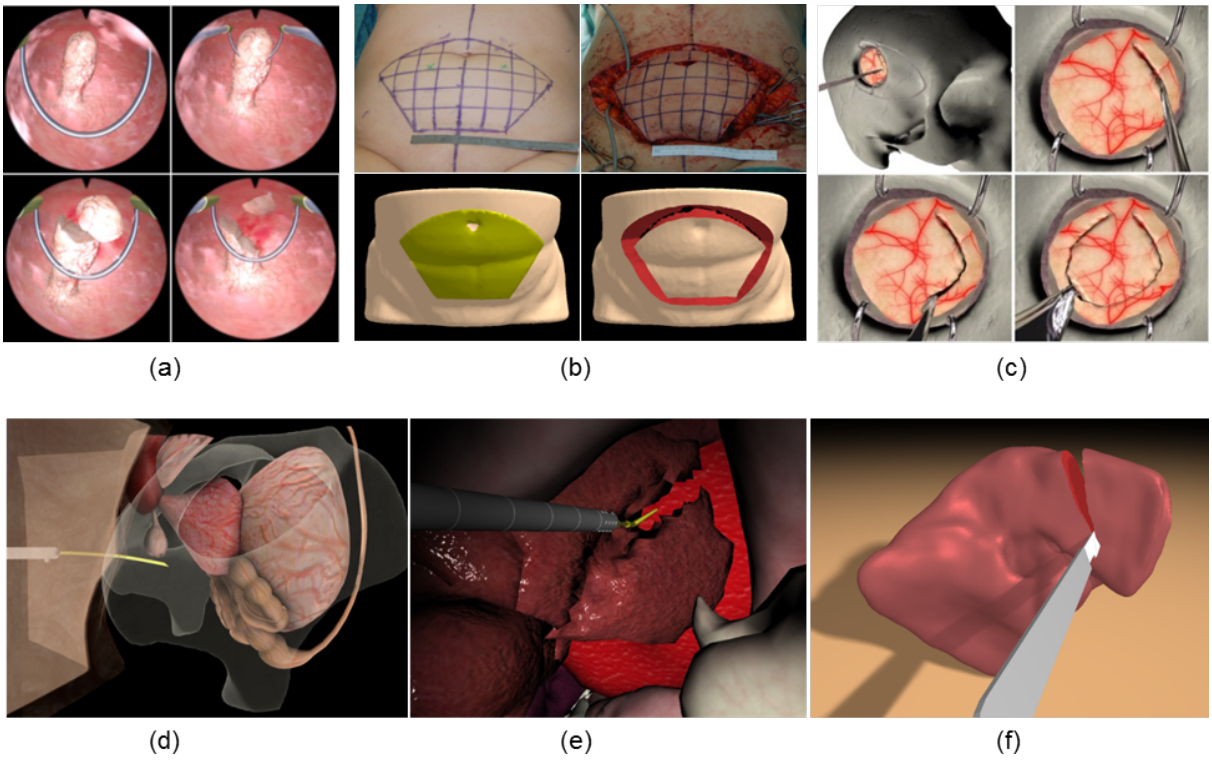

Figure 16: Interactive simulation in medical contexts. (a) Ablating a polyp in a hysteroscopy simulator (image courtesy of Steinemann et al. [SHGS06] ©2006 IEEE). (b) Virtual soft tissue cutting and shrinkage simulation [WBWD12] (abdomen photographs courtesy of Dr. med. Laszlo Kovacs). (c) Real-time 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.