A Flexible and Interactive Approach for Cutting Deformable Objects

Christian Dick, Joachim Georgii, and Rüdiger Westermann
Computer Graphics and Visualization Group, Technische Universität München, Germany

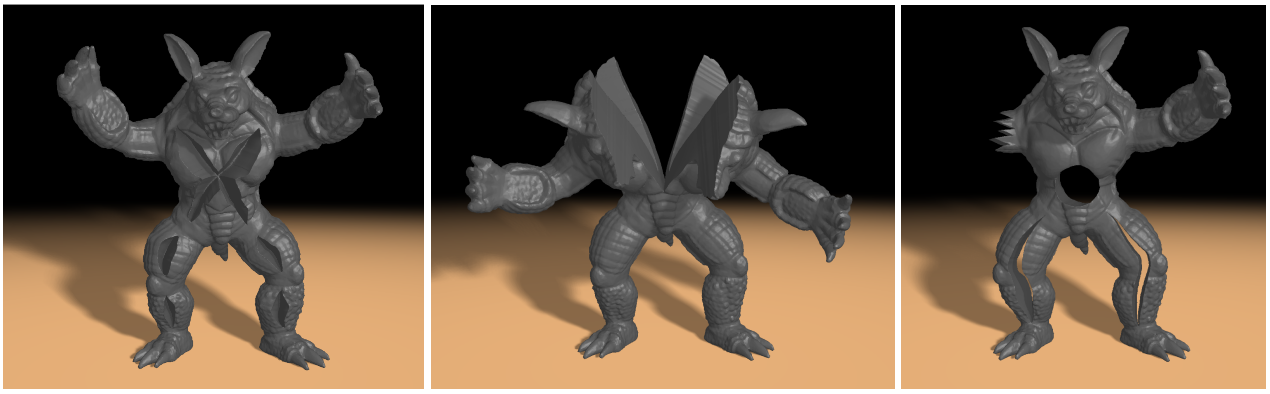

Figure 1: *Arbitrary cuts in the Stanford Armadillo model. An octree finite hexahedra model consisting of 307,000 simulation elements is used. From left to right, 46,000, 90,000, and 92,000 additional elements were induced by the cuts. Cutting and simulation of this model is performed at four to five seconds per time step.*

Abstract

We present a hierarchical finite element method for interactively simulating cuts in linear elastic deformable bodies. Our method draws upon recent work on octree and multigrid schemes for the efficient numerical solution of partial differential equations. We propose a novel approach for incorporating topological changes in octree grids into multigrid schemes, including algorithms for updating the system equations on successive multigrid levels. A hexahedral grid is adaptively refined at the surface of a cutting tool until a finest resolution level, and the cut is simulated by separating elements along the cell faces at this level. Since all elements have the same shape, only a scaling of a pre-computed matrix is required to construct the stiffness matrices of new elements. To build a multigrid hierarchy that reflects the induced discontinuities, elements at coarse-grid levels are duplicated to represent the resulting connectivity components. An extension of the splitting cubes algorithm is used to construct a surface that accurately aligns with the simulated cuts. A comparison to finite element simulations on tetrahedral grids in which cuts have been modeled explicitly shows very high accuracy of the proposed technique. By using our approach, interactive cuts at an effective object resolution of 256^3 can be achieved.

Keywords: Interactive cutting, deformable objects, finite elements, multigrid, octree meshes

1 Introduction

Interactive, yet realistic simulations of incisions and cuts in deformable bodies place strong requirements on the used algorithms and technologies. Simulation tools supporting this kind of operability must be able to handle topological changes of the underlying simulation domain and to accurately predict the movement of the resulting deformable body parts due to internal and external loads. The efficient construction of an accurate boundary surface of the separated parts is crucial for using such tools in virtual environments.

Starting with the seminal work of Terzopoulos and co-workers [1987; 1988], physics based methods for simulating deformable models have been researched extensively in computer graphics for the last two decades. In numerous publications the many aspects of deformable body simulation have been addressed, resulting in a multitude of strategies to realistically simulate the movement of such objects. Among these, especially finite element methods have been shown to be able to effectively model the relevant physical and mechanical principles underlying deformable body simulation.

To efficiently simulate cuts in deformable bodies, the cutting algorithm has to be designed on par with the simulation technique used to predict the resulting body movements. Finite element methods on tetrahedral grids often make use of tetrahedral subdivision to cut the mesh [Bielser et al. 1999; Bielser and Gross 2000; Mor and Kanade 2000]. This enables efficient element refinements and makes element integration straightforward, but it can create severely ill-conditioned simulation elements. Moreover, it is not clear how efficient solvers like geometric multigrid methods can be used on such refining grids. Since the resulting meshes are unstructured, in general, the construction of a mesh hierarchy at rates that are suitable for interactive applications is very difficult to achieve.

From a simulation point of view, tetrahedral subdivision requires adding new element equations to the system matrix and updating the equations of elements affected by a cut. To assemble these equations, the stiffness matrices of new elements have to be computed via element-wise integration. Since tetrahedral subdivision creates a large number of additional elements, the size of the system of equations as well as the computational cost for updating this system grow rapidly if many cuts are performed.

Polyhedral subdivision [Wicke et al. 2007] creates significantly less elements than tetrahedral subdivision, but it requires a numerically more involved integration. The virtual node algorithm [Molino et al. 2004] avoids element splitting by duplicating elements that are cut and distributing the resulting material components to the duplicates. It thus avoids ill-shaped elements and only moderately increases the element count. On the other hand, both approaches restrict a cut to the resolution of the simulation grid, which makes it difficult to partially cut an element or to perform a non-planar cut

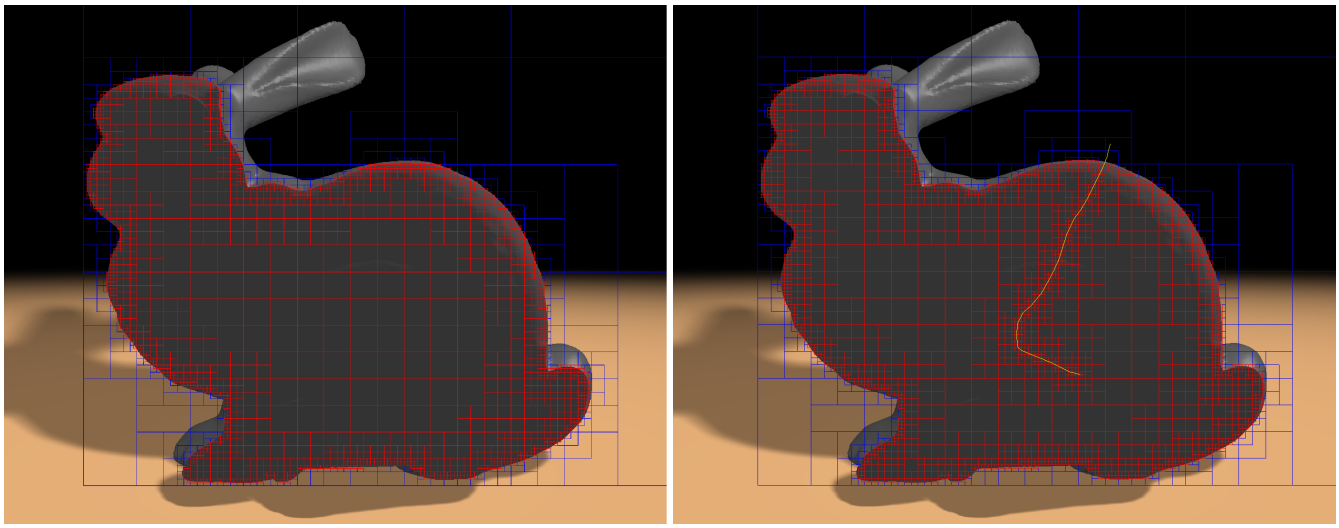

Figure 2: *A cross-section through an adaptive octree grid without (left) and with (right) a cut.*

through an element.

Extended finite elements (XFEM) [Abdelaziz and Hamouine 2008; Jeřábková and Kuhlen 2009] can overcome this limitation by using additional step functions to represent material discontinuities that accurately align with a cut. Enrichment fields for XFEM as proposed in [Kaufmann et al. 2009] for simulating cuts in 2D structures also show this potential. However, since XFEM come at the expense of significantly increasing computational cost of element integration they may not be well suited for interactive 3D applications. [Sifakis et al. 2007a] combine the virtual node algorithm with a method to accurately compute polyhedral material components in the cut elements. This allows keeping the number of simulation elements low, but it requires an exhaustive geometric clipping procedure to determine the components on either side of a cut.

Our contribution. We present a fast and accurate approach for simulating cuts in deformable objects. This approach alleviates many of the limitations of previous approaches. For example, it is independent of the initial grid resolution and creates well-conditioned simulation elements. We achieve this by introducing a multigrid finite hexahedron method for physics based simulation on adaptive grids. Because the method only requires scaled instances of one local stiffness matrix to accommodate whatever shape is needed, it has small storage requirements and only needs to consider a few cases in the construction and update of the multigrid hierarchy. Figure 1 shows some cuts that have been performed using our approach.

Instead of remeshing the simulation grid along a cut as in tetrahedral subdivision, an adaptive finite hexahedron approximation of the cut object is built from the actual precise location of the cut. Simulation elements that are touched by the cutting tool are recursively subdivided using a regular octree refinement, so that the refinement levels of adjacent cells do not differ more than one. An example is given in Figure 2. The refinement is performed until a sufficient approximation is reached, and on this subdivision level cutting is performed along the element faces. To render a smooth polygon surface that aligns with the cut, we extend the splitting cubes algorithm [Pietroni et al. 2009] to compute a watertight boundary surface from the 3D octree grid.

Since the refinement process results in an increasing element number, a scalable method is required to enable interactive, yet realistic simulation of the deforming body parts. To achieve this goal, we first discretize the linear elastic finite element model on the octree that is generated by the cutting algorithm. The corotated strain formulation [Müller et al. 2002; Müller and Gross 2004; Hauth and Straßer 2004; Georgii and Westermann 2008] is extended to hexahedral elements to enable large deformations. Our method employs previous results on octree-based finite difference and finite element discretizations [Popinet 2003; Losasso et al. 2004; Haber and Heldmann 2007; Sampath and Biros 2009] for building the element equations, but we extend these approaches to handle topological changes of the simulation grid.

We then introduce a novel multigrid solver for the octree discretization to achieve linear time complexity. The hexahedral simulation grid accommodates fast updates of the multigrid hierarchy to reflect the adaptive refinements induced by a cut. In contrast to tetrahedral grids, the regular structure of the octree allows building a nested geometric hierarchy as well as the corresponding interpolation and restriction operators straightforwardly.

Since none of the previous multigrid approaches considers topological changes of the simulation grid, we propose a novel algorithm that provides this functionality. Incorporating topological changes into a multigrid scheme requires transferring these changes to the coarser grids, involving special treatment of the coarse-grid finite elements that cover topological splits. We present a new approach to efficiently handle these cases by duplicating respective elements and distributing the system equations to the duplicates according to the resulting material components.

2 Related Work

In computer graphics, deformable models were first introduced by Terzopolous et al. [1987; 1988]. A good overview of the multitude of methods for realistically simulating deformable bodies can be found in [Nealen et al. 2005]. For example, boundary element models [James and Pai 1999], adaptive and multiresolution approaches [Debunne et al. 2001; Capell et al. 2002; Grinspun et al. 2002], gridless techniques [Müller et al. 2005; Sifakis et al. 2007b], and finite element methods [Bro-Nielsen and Cotin 1996; Wu et al. 2001]

have been proposed. The simulation of brittle fracture based on finite elements was described by O'Brien and Hodgins [1999] and later extended to ductile fracture [O'Brien et al. 2002]. Nesme et al. [2009] proposed a composite element formulation that considers varying material properties within a coarse element.

Tetrahedral subdivision methods for cutting deformable objects were introduced in [Bielser et al. 1999; Mor and Kanade 2000]. To reduce the number of ill-shaped elements, Nienhuys and van der Stappen [2000] proposed cutting along the element faces. Cotin et al. [2000] and Forest et al. [2002] deleted elements that were cut. Smooth cuts that also reduce the number of ill-shaped elements were achieved in [Nienhuys and Stappen 2001; Serby et al. 2001; Steinemann et al. 2006] by adaptively aligning mesh edges and faces with the cutting surface. By restricting subdivisions to a few refinement patterns [Bielser and Gross 2000; Bielser et al. 2003] the number of additional simulation elements caused by a cut can be reduced. A multi-resolution approach for this method was presented in [Ganovelli et al. 2000]. The virtual node algorithm [Molino et al. 2004] avoids ill-shaped elements by duplicating simulation elements and re-assigning material components on both sides of a cut.

[Wicke et al. 2007; Kaufmann et al. 2008] introduced polyhedral subdivision, which splits initial tetrahedra into polyhedra and then subdivides these elements further. Extended finite element methods [Belytschko and Black 1999] enrich a finite element model with specific basis functions to capture discontinuities in the simulation elements. The use of XFEM for virtual surgery simulation and cut simulation in 2D thin shells was demonstrated in [Abdelaziz and Hamouine 2008; Jeřábková and Kuhlen 2009] and [Kaufmann et al. 2009], respectively. [Sifakis et al. 2007a] clipped a high-resolution material boundary surface mesh against a coarse simulation mesh to consider fine material components in a coarse elasticity simulation.

Octree-based physical simulation of fluids and gases was shown in [Popinet 2003; Shi and Yu 2004; Losasso et al. 2004]. Both restricted and unrestricted octrees were used. To achieve high resolution of small scale details, one focus was on deriving adaptive finite difference discretizations of the governing equations. Finite element discretizations for the numerical solution of partial differential equations on restricted octrees were introduced in [Haber and Heldmann 2007; Sampath and Biros 2009].

Multigrid approaches [Briggs et al. 2000] for the solution of large linear systems have recently gained much attention in the computer graphics community due to their linear time complexity. Applications range from fluid simulation [Bolz et al. 2003] over deformable body simulation [Wu and Tendick 2004; Georgii and Westermann 2006; Shi et al. 2006] to image processing [Kazhdan and Hoppe 2008].

3 Cutting Algorithm

To enable interactive cuts, we propose an efficient algorithm that cuts along the element faces in a hexahedral grid. For the sake of simplicity, we will first describe the cutting procedure in a uniform grid before we introduce the extension towards non-uniform octree grids.

3.1 Cuts - Uniform Grid

Therefore, let us assume that the object to be cut has been discretized into a hexahedral grid with sufficient resolution to capture all relevant details. Discretization means building a binary representation, where every hexahedron is classified as inside or outside depending on whether it's center is in the interior of the object or not. We call a hexahedron that belongs to the object a voxel. Voxels represent the finite elements that are used in the physics based simulation.

We leverage a linked volume representation [Frisken-Gibson 1999] that connects the centers of face adjacent voxels via links. Initially, each 3D voxel has six links. Cutting the FE-model is performed by disconnecting the links that are cut by the cutting blade (see Figure 3 for an illustration). In our implementation the blade is realized as a triangle mesh, which is built from consecutive cut-lines that are induced via a cutting tool. To find the links that are cut, all links in the vicinity of the cutting front, i.e., the surface spanned by the current and the last cut-line, are tested for an intersection with the triangles representing this front. Note that in the deformed state the displaced voxel centers have to be considered to find the intersections. To prune as many links as possible in the intersection test, we compute a bounding box hierarchy for the deformed state based on the octree representation introduced below.

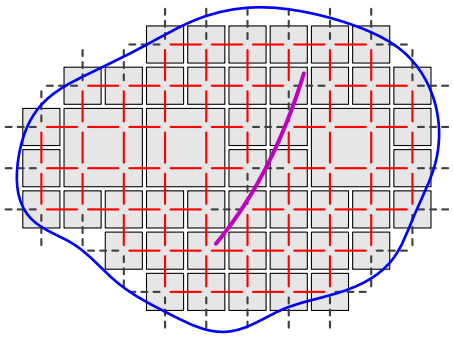

Figure 3: *Illustration of the linked volume representation of the object, consisting of a set of voxels (gray) which are connected via links (red) (for simplicity in 2D). These links are released (dashed, gray) when cut by a cutting surface (violet). An adaptive octree representation is built by successively merging blocks of 2^3 voxels with full connectivity into larger voxels.*

3.2 Cuts - Octree Grid

An adaptive object discretization is built from the uniform hexahedral discretization in a bottom-up process. Contiguous blocks of 2^3 voxels that do not have been cut are merged into larger hexahedra. This process is recursively repeated up to a coarsest level. The octree is restricted to not have level differences between adjacent cells of more than one. This is later required for the construction of a multigrid hierarchy, where equations of fine grid vertices are distributed to corresponding coarse grid vertices. If an unrestricted octree would be used, the corresponding vertices might be hanging as illustrated in Figure 4, and therefore the equations cannot be distributed to these vertices [Wang 2000; Sampath and Biros 2009].

Voxels on the coarser octree levels do not establish links as on the finest level (level zero), but instead the links associated to the voxels on the finest level are pulled up to the respective coarse grid voxel (see Figure 3). For example, on the first level a voxel contains $8 \cdot 6$ links, on the second level it contains $8 \cdot 8 \cdot 6$ links, and so on. The reason therefore is that in our implementation cutting is always performed by disconnecting links on the finest level. Even if a cut is through a voxel on one of the coarser levels, we test for intersections between the blade and the finest level links. Fortunately, because every voxel on a particular coarse level carries full connectivity, the respective links do not have to be stored explicitly.

If a coarse grid voxel is cut, all links in this voxel that are intersected by the blade are disconnected, and the voxel is adaptively

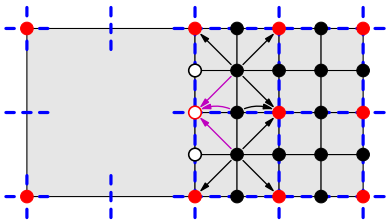

Figure 4: *Equations of fine grid vertices (black) are distributed to corresponding coarse grid vertices (red). In case of an unrestricted octree, some coarse grid vertices might be hanging (white with red border).*

refined down to the finest level along the disconnected links. This procedure creates additional elements, which number increases proportional to the surface increase of the object that is cut.

4 Multigrid Solver Supporting Cuts

Once the adaptive discretization of the cut object has been generated, the governing equations describing the dynamic behavior of this object under the influence of external forces have to be discretized accordingly. We use a finite element approach with implicit second-order Newmark time integration to discretize the Lagrangian equations of motion for a linear elastic body on the hexahedral grid [Bathe 2002]. The discretization results in a large system of equations $Ku = f$ that needs to be solved in every time integration step. Here, u is the linearized displacement vector and f contains the forces applied to the finite element vertices. The global stiffness matrix K assembles all element stiffness matrices, which are constructed using tri-linear shape functions. We extended the corotated Cauchy strain formulation [Müller et al. 2002; Müller and Gross 2004; Hauth and Straßer 2004; Georgii and Westermann 2008] to hexahedral finite elements, which allows to simulate large deformations. Specifically, the polar decomposition is used to determine a hexahedron's rotation from its average deformation gradient.

In general, the system of equations can be solved with many different solvers like a conjugate gradient solver or Cholesky factorization. However, our particular scenario requires a solver that can efficiently handle a system that changes in every time step: the system has to respect topological changes due to cuts and it has to be updated with respect to the current element rotations to incorporate the corotated strain formulation.

Geometric multigrid solvers on hexahedral grids meet these requirements, since (a) the construction of the multigrid hierarchy on such grids can be performed very efficiently due to the straightforward realization of the restriction and prolongation operators, and (b) the linear-time complexity of multigrid solvers makes them particular attractive in scenarios like ours where scalability in the number of elements is required. It is worth noting that algebraic multigrid methods, which construct the transfer operators from the system matrix, do not conform to the first requirement in general.

It now also becomes apparent why we favor an adaptive hexahedral discretization over, for instance, a tetrahedral discretization. Since all simulation elements have the same shape and only differ in size, we need only one instance of an element stiffness matrix. If we denote by K^e this instance for an uniform hexahedron with side length 1, then an uniform hexahedron with side length s has the element stiffness matrix sK^e [Ježábková et al. 2004].

The challenge in developing a multigrid approach that supports topological changes of the simulation grid lies in constructing the coarse grid hierarchy. Since cuts also have to be represented on the coarser levels, voxels cannot simply be merged based on their spatial relationship. This would correspond to merging physically and mechanically disconnected parts, and it would result in low approximation quality and very slow convergence rate or even divergence of the solver. For the sake of clarity, we first describe the construction of a multigrid hierarchy for a uniform hexahedral grid with cuts.

4.1 Hierarchy Construction - Uniform Grid

Without cuts, building the multigrid hierarchy simply involves merging blocks of 2^3 cells into one coarse grid cell. The corresponding coarse grid equations are formed by regularly distributing the equations of the fine grid vertices to the respective coarse grid vertices [Briggs et al. 2000]. This requires only 4 different cases to be considered in 3D (see Figure 5 for an illustration of the respective cases).

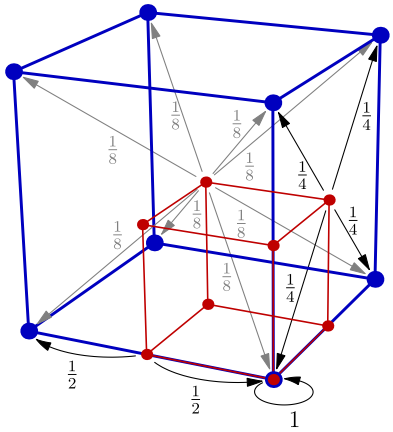

Figure 5: *Weights used to distribute the per-vertex equations of a fine grid cell (red) to its corresponding coarse grid cell (blue). For the interpolation of the unknowns from the coarse grid, the same weights are used at the fine grid (Galerkin condition). For simplicity reasons, the distribution is only illustrated for selected vertices.*

To respect cuts in the simulation domain, a similar merging strategy is performed but the connectivity between merged voxels is considered to eventually generate more than one coarse grid cell at the same position. Starting on the initial voxel grid, where links have been established as described in Section 3, the following steps are performed subsequently at each level to build the coarse grid hierarchy (see also the 2D example in Figure 6):

Step 1: Analogously to the case without cuts, the blocks of double fine grid cell size are processed independently, and for each of these blocks all fine grid cells are considered that lie inside the block. However, these cells are interpreted as nodes of an undirected graph, in which the edges correspond to the links between the cells. Links to cells which are not in the same block are excluded. On this graph, we determine the connected components using a depth-first search. For each component, one coarse grid cell that merges the cells belonging to this component is created. Similar to the virtual node algorithm [Molino et al. 2004], at the same geometric position multiple cells might be created in this way. In contrast to the virtual node algorithm, however, these cells do not have any physical or mechanical relevance in our approach, since they are only used to accelerate the solution process of the mechanical equations given by the voxels. Hence, our algorithm avoids

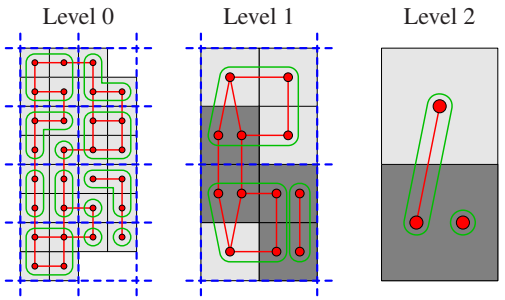

Figure 6: *Construction of the coarse multigrid levels based on the undirected graph representation (red) in a uniform grid. The blocks of double cell size, in which connected components (green) are searched, are marked in blue. Blocks with duplicated cells are dark gray.*

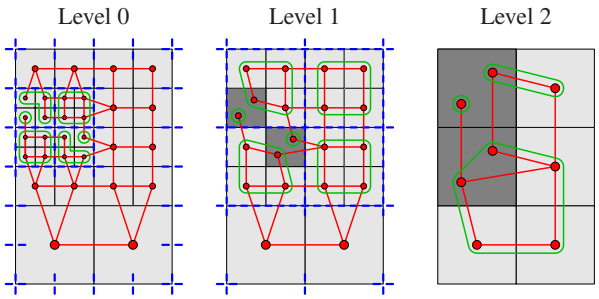

Figure 7: *Construction of the coarse multigrid levels based on the undirected graph representation in an octree grid. The notation is analogously to Figure 6.*

duplicating elements that have a mechanical relevance, and therefore respects the underlying physical model more accurately.

Step 2: The connectivity between cells on the coarse grid level is established. Therefore we consider exactly those links which had been excluded in the first step. If a link between any two cells in two adjacent coarse grid cells exists, a link between these two coarse grid cells is established and stored. Cells which are connected via links share their vertices to allow equations to be built at these vertices.

The hierarchy that is constructed in this way has the property that small cuts disappear at coarser resolution levels. This is in-line with the multigrid idea that the coarse grid levels describe the macroscopic behavior of the body. Small cuts do not strongly affect this behavior and can therefore be neglected at one of the coarser levels.

4.2 Hierarchy Construction - Octree Grid

The multigrid scheme on the adaptive octree grid has to consider that the grid has hanging vertices which require special treatment. We use the special finite elements introduced by Wang [2000] and extended to 3D by Sampath and Biros [2009], in which hanging vertices are replaced by linear (along edges) and bilinear (on faces) interpolation of adjacent vertices. For each voxel configuration in which a hanging vertex occurs we determine an element striffness matrix by adapting our pre-computed element stiffness matrix K^e accordingly. The resulting 18 instances [Sampath and Biros 2009] are cached in a small lookup table. In this way, hanging vertices are removed from the finite element model.

Building the hierarchy proceeds similar to the uniform case, with the exception that on each level it must be considered that cells with different size exist. At the transition from level l to level $l+1$, only cells of size 2^l are merged. All other cell are passed to the next coarser level. Allowing also larger cells on each level to be merged would require a rebalancing to ensure the restrictedness of the octree. This is avoided by our approach, which only slightly increases the number of coarse grid cells. In Figure 7 a 2D example of the construction process is given.

Finally, once the cell hierarchy has been constructed the coarse grid equations have to be built correspondingly to use the multigrid solver. Therefore, the fine grid equations are propagated to the respective coarse grid vertices as shown in Figure 5. Since the corotated strain formulation is used, the equations change in every time step to account for the element rotations.

The equations on each hierarchy level are computed and stored per *cell*. This results in constant memory requirements per cell to store the equations and in constant memory access patterns during equation construction. In contrast, storing and constructing the equations per *vertex* would yield varying memory requirements and access patterns, since due to the cutting vertices can have a varying number of adjacent vertices. Note that there is no need to store the equations on the finest level, since they can be built on-the-fly from the generic element stiffness matrix and the element rotations. In our implementation this gives a performance gain of a factor of 2.5 due to increased cache coherence.

We now explain the construction of the per-cell equations on the coarse grid levels. The equations for each element c are stored as a 8×8 matrix K^c, with each coefficient being a 3×3 matrix. The matrix K^C for a coarse grid cell C is computed from the matrices K^c of all cells c on the finer level that are merged into cell C as follows:

$$K^C_{mn} = \sum_{c \text{ in } C} \sum_{i=1}^{8} \sum_{j=1}^{8} w^{c \to C}_{i \to m} \, w^{C \to c}_{n \to j} \, K^c_{ij} \;, \quad m,n = 1 \ldots 8. \quad (1)$$

In this equation, i, j, m, n are element-local vertex indices, and $w^{c \to C}_{i \to m}$ and $w^{C \to c}_{n \to j}$ are the corresponding weights for restriction and prolongation between the fine grid cells c and the coarse grid cell C, respectively (see Figure 5).

To finally solve the system of equations, in the relaxation steps of the multigrid V-cycle, one has to consider the equations per vertex. The per-vertex equations are assembled on-the-fly during the relaxation step from the per-element equations as follows: Considering a vertex v, we accumulate the contributions of the corresponding element vertices $V \subset \mathcal{C} \times \{1 \ldots 8\}$, where $c \in \mathcal{C}$ denotes the element and $i, j \in \{1 \ldots 8\}$ denote local vertex indices. The equation at vertex v then is

$$\sum_{(c,i) \in V} \sum_{j=1}^{8} K^c_{ij} u^c_j = f^v, \quad (2)$$

where u^c_j are the displacement vectors at the vertices of element c, and f^v is the force acting on the vertex v.

5 Surface Construction

To reconstruct a smooth surface that is aligned with the cut object one can simply store the respective parts of the cut surface at each surface hexahedron. These parts can then be attached to the deforming vertices via tri-linear interpolation to move accordingly. However, this implies that a left and a right side of the cut surface

has to be generated, of which only one lies within the simulation elements while the other one is outside. Although this problem can be solved by introducing new cells that cover the respective other part of the surface, this approach results in quite complex cases to be considered.

For this reason, we provide a simpler and more elegant approach based on a dual grid representation. The dual grid is built from the links between the voxel centers. These links define the connectivity between the simulation elements, and exactly these links are cut by the cutting tool. At each link that is cut we store two distances that describe the exact cut points on this link, thereby allowing a link to be cut multiple times. In this way, the cubic cells of the dual grid are cut at their edges, which results in a representation that allows us to directly apply the splitting cubes algorithm [Pietroni et al. 2009] at these cells. The splitting cubes algorithm constructs the cut surface in the interior of a cell by distinguishing the different cases from the patterns of the cut edges. The algorithm introduces interior points in the cell by considering the normals of the cut surface at the intersection points, such that a smooth surface is constructed in the interior of a cell as shown in Figure 8.

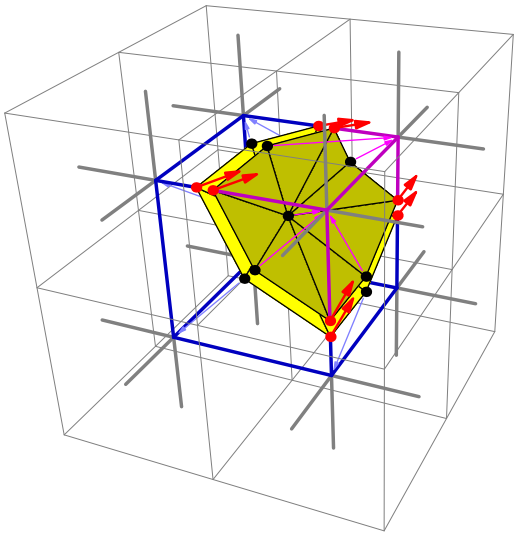

Figure 8: *Dual grid (bold lines) consisting of the links between the voxels (thin gray lines). Using the splitting cubes algorithm, for each cell of the dual grid a local rendering surface is built. The surface is spanned by the intersection points (red) of the links with the cutting surface as well as the cutting surface's normals at these points (red arrows). For the deformation, vertices are bound to the nearest voxel of the respective connected component (thin blue and magenta arrows to the respective voxel centers).*

A question that remains to be answered is how the surface along the cut is bound to the simulation grid, such that it deforms accordingly. For vertices on the links that are cut, the corresponding simulation element is found by following the links to the left or to the right, respectively. To handle interior vertices we utilize the fact that each interior vertex is dedicated to a unique surface patch in the splitting cubes approach. Therefore, the respective hexahedra can be found by following the links of the edges that are cut by the respective surface patch. From the determined set of hexahedra we choose the one that has the shortest distance to the surface vertex. The interior vertex is then bound to this hexahedron by means of trilinear interpolation or extrapolation.

6 Results and Analysis

Performance. All of our experiments were run on a desktop PC, equipped with an Intel Xeon X5560 2.8 GHz processor, 8 GB of RAM, and an NVIDIA GeForce GTX 280 graphics card. Table 1 provides information on the simulation models we have used and the performance of our approach for cutting and simulation. The first column gives the initial number of simulation elements we started with and the maximal number of elements that were generated due to cuts. The next column shows the time spent on adapting the octree, rebuilding the multigrid hierarchy by determining the connected components, and constructing the splitting cubes surface. Note that these steps are only required while the cutting tool is moved through the object. Finally, the time it takes to simulate one time step (we use a fix time step of 0.05 seconds) is given, which includes recomputing the coarse grid equations (as shown in Equation 1) and performing two multigrid V-cycles. Note that this time is measured for the maximal number of hexahedra. As can be seen, for grids of moderate size, interactive rates for both cutting and physics-based simulation can be achieved.

	#Hexahedra		Time [sec]	
Model	Initial	Max	Cutting	Simulate
Liver	16,426	17,511	0.042	0.175
Bunny	117,650	177437	0.371	1.851
Armadillo	307,354	397,171	1.057	4.154
Cube	133,232	440,168	1.198	4.537

Table 1: *Timing statistics for various models using our adaptive octree approach for cutting and simulation.*

For the high-resolution Armadillo model shown in Figure 1, the octree approach results in a performance gain (cutting and simulation) of a factor of 4.5 over a uniform simulation grid on the finest resolution level. At the same time, in the octree approach the number of simulation elements is only $1/6$ of the number of elements in the uniform setting.

Validation. To validate the accuracy of our approach, we compare the octree-based simulation of a cut object to a simulation using a finite tetrahedra model where the same cut is modeled explicitly. For the tetrahedral model a geometric multigrid solver was used on a non-nested mesh hierarchy. This hierarchy was generated in a pre-process using successive coarsening, which alone took several minutes; it is not clear how a tetrahedral mesh hierarchy can be efficiently constructed to make tetrahedral models amenable to multigrid solvers in scenarios where cuts are induced. In contrast, our approach avoids time-consuming mesh computations. Our adaptive finite hexahedra approach was performed on 140,757 elements,

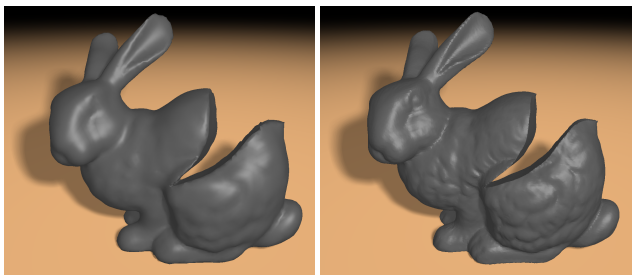

Figure 9: *Simulation validation. Left: Reference simulation on a tetrahedral grid (166,000 elements). Right: Our octree approach using 141,000 hexahedral simulation elements.*

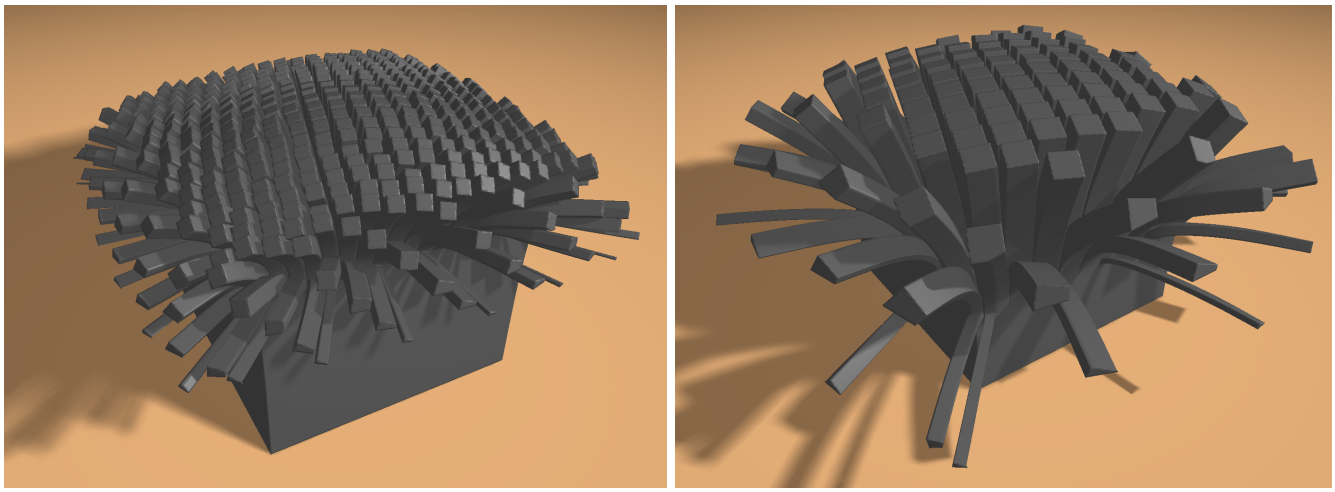

Figure 10: *A cube model is simultaneously cut at many positions using a grid-like cutting tool consisting of square blades, resulting in 355,000 (left) and 440,000 (right) hexahedra. A radial force field is applied to fan out the resulting rods. Note that rods with different size bend differently. Cutting and simulation takes less than 4.8 seconds (left) and 5.7 seconds (right) per time step.*

while the reference simulation used a tetrahedral grid with 165,806 elements. In both cases, the same cut was carried out, and the same external forces were applied to open the cut. As can be seen in Figure 9, the two approaches do not show any noticeable differences, besides the geometric differences in the two boundary surfaces.

Multigrid Analysis. Since we carry out a simulation of the dynamics of a deformable body, the solution from the last time step already gives a good initial guess to start with in the current time step. Due to this reason we observed high accuracy using 2 V-cyles per time step, each with 1 pre- and 1 post-smoothing step. Specifically, the residual was then already reduced to $1/4$ in our examples, yielding maximal errors in the computed displacement field below the voxel size[1]. Since a V-cycle is rather cheap compared to the update of the multigrid equations, one can apply more V-cycles per time step without substantially affecting the overall performance.

It is clear, on the other hand, that the convergence rates we achieve are not as good as, for instance, reported in [Kazhdan and Hoppe 2008] for solving a Laplacian system. The elastic problem we consider in this work is numerically far more involved, and it has to deal with a complex boundary setting that is known for it's tendency to decrease the convergence rate. Regardless these issues, however, especially due to its efficient update mechanism for incorporating topological changes and its capability to effectively scale in the number of simulation elements, the multigrid scheme on finite hexahedra showed extremely well suited for the particular application. A conjugate gradient solver, for instance, which is widely used in numerical simulation techniques, showed a significantly slower convergence in our application due to the stiffness of the equations. Direct solvers like the Cholesky factorization, on the other hand, turned out to be impractical in this application. This is due to the change of the system of equations in every time step, which requires performing the time-consuming factorization in every time step, too.

Examples. In Figure 10 we applied multiple simultaneous cuts in a cube using a uniform grid of blades consisting of many squared cutting surfaces. The finite element model consists of 133,232 hexahedra (up to 440,168 hexahedra after cutting). Even such a complex topology modification can still be performed at roughly 5 seconds per time step, including cutting and simulation. The different bending and twisting of the resulting rods indicates high quality in resembling the underlying physical model. Figure 11 shows arbitrary cuts through the Stanford bunny model. Note that in contrast to many previous approaches, multiple cuts through the same object point as demonstrated in the middle image can be handled accurately by our method. Therefore, our approach alleviates the restriction to a limited number of cuts, which is inherent to many other approaches.

[1]Please note that the voxel size is the approximation made to represent the mechanic properties at the cuts, and a more accurate solution of the displacement vectors does not necessarily provide a better solution.

7 Conclusion and Future Work

We presented a flexible and interactive approach for cutting deformable objects. The method uses an adaptive octree grid to represent cuts at an arbitrary scale. Due to a novel multigrid approach, topological changes in the octree grid can be handled efficiently, thereby allowing a large amount of finite elements to be cut and deformed interactively. An extension of the splitting cubes approach has been proposed, which allows to construct a smooth surface from the cut simulation grid, which can be either used to render the object or to detect collisions. The performance and flexibility of our approach makes it amenable for a large range of applications, including virtual surgery environments as demonstrated in Figure 12.

The method opens also interesting areas for future research. So far we assumed homogeneous objects to be simulated. For heterogeneous objects, however, the way the simulation octree is generated has to be modified, such that the octree also adapts to material jumps in the object. In future we will investigate how our multigrid approach can be optimized for this scenario. Furthermore, we will also integrate collision handling into our approach. We plan to use the deforming splitting cubes surface to detect self-collisions [Teschner et al. 2005], while the collision response is then propagated to the respective simulation voxels.

References

ABDELAZIZ, Y., AND HAMOUINE, A. 2008. Review: A survey of the extended finite element. *Comput. Struct. 86*, 11-12, 1141–

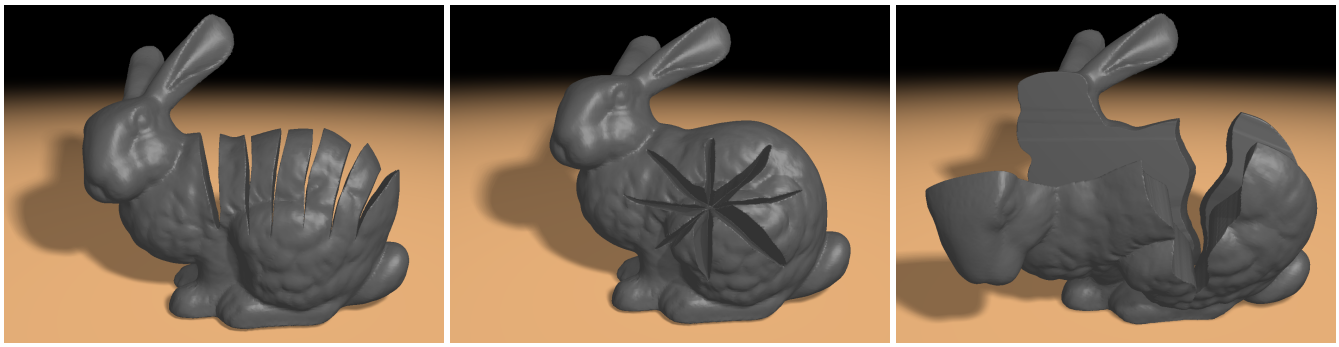

Figure 11: *A potpourri of different cuts is applied to the Stanford bunny model to demonstrate the flexibility of our approach. From left to right, the cut models consist of 196,000, 166,000, and 177,000 hexahedral elements. Cutting and simulation are performed at approximately 2.4, 2.0, and 2.1 seconds per time step, respectively.*

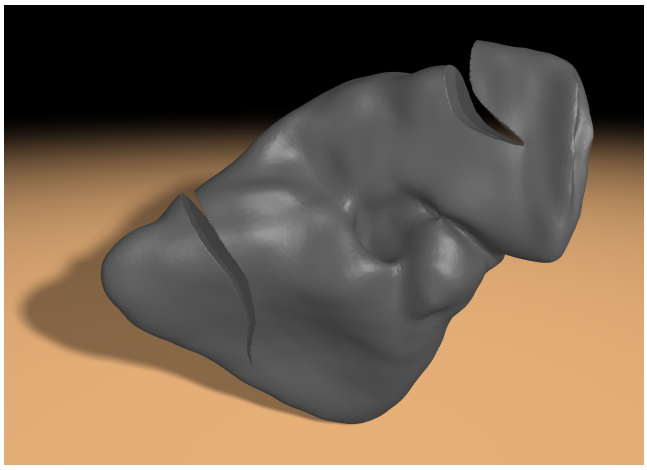

Figure 12: *Interactive cuts in a model of a human liver consisting of 17,511 hexahedra demonstrate the potential of our approach for virtual surgery simulation. Cutting and simulation are carried out at 5 time steps per second.*

1151.

BATHE, K.-J. 2002. *Finite Element Procedures*. Prentice Hall.

BELYTSCHKO, T., AND BLACK, T. 1999. Elastic crack growth in finite elements with minimal remeshing. *Int. J. Numer. Methods Eng.*, 5, 601620.

BIELSER, D., AND GROSS, M. 2000. Interactive simulation of surgical cuts. In *Pacific Graphics*, 116–125.

BIELSER, D., MAIWALD, V. A., AND GROSS, M. H. 1999. Interactive cuts through 3-dimensional soft tissue. *Comput. Graph. Forum 18*, 3, 31–38.

BIELSER, D., GLARDON, P., TESCHNER, M., AND GROSS, M. H. 2003. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Conference on Computer Graphics and Applications*, 377–386.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. 2003. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In *Proceedings of SIGGRAPH*, 917–924.

BRIGGS, W. L., HENSON, V. E., AND MCCORMICK, S. F. 2000. *A Multigrid Tutorial, Second Edition*. SIAM.

BRO-NIELSEN, M., AND COTIN, S. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics*, 57–66.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 41–47.

COTIN, S., DELINGETTE, H., AND AYACHE, N. 2000. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training. *The Visual Computer 16*, 7, 437–452.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of SIGGRAPH*, 31–36.

FOREST, C., DELINGETTE, H., AND AYACHE, N. 2002. Removing tetrahedra from a manifold mesh. In *CA '02: Proceedings of the Computer Animation*, IEEE Computer Society, Washington, DC, USA, 225.

FRISKEN-GIBSON, S. F. 1999. Using linked volumes to model object collisions, deformation, cutting, carving, and joining. *IEEE Transactions on Visualization and Computer Graphics 5*, 4, 333–348.

GANOVELLI, F., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 2000. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Comput. Graph. Forum 19*, 3, 271282.

GEORGII, J., AND WESTERMANN, R. 2006. A Multigrid Framework for Real-Time Simulation of Deformable Bodies. *Computer & Graphics 30*, 408–415.

GEORGII, J., AND WESTERMANN, R. 2008. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*, 11–19.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: a simple framework for adaptive simulation. In *Proceeding of SIGGRAPH*, 281–290.

HABER, E., AND HELDMANN, S. 2007. An octree multigrid method for quasi-static maxwell's equations with highly discontinuous coefficients. *J. Comput. Phys. 223*, 2, 783–796.

HAUTH, M., AND STRASSER, W. 2004. Corotational simulation of deformable solids. In *Proceedings of WSCG*, 137–145.

JAMES, D. L., AND PAI, D. K. 1999. ArtDefo: accurate real time deformable objects. In *Proceedings of SIGGRAPH*, 65–72.

JEŘÁBKOVÁ, L., AND KUHLEN, T. 2009. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Comput. Graph. Appl. 29*, 2, 61–71.

JEŘÁBKOVÁ, L., KUHLEN, T., WOLTER, T. P., AND PALLUA, N. 2004. A voxel based multiresolution technique for soft tissue deformation. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA, 158–161.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2008. Polyhedral finite elements using harmonic basis functions. *Comput. Graph. Forum 27*, 5, 1521–1529.

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. 28*, 3, 1–10.

KAZHDAN, M., AND HOPPE, H. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph. 27*, 3, 1–10.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 457–462.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3, 385–392.

MOR, A. B., AND KANADE, T. 2000. 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*, Springer-Verlag, London, UK, 598–607.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface*, 239–246.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 49–54.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3, 471–478.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Proceedings of Eurographics*, 71–94.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, ACM, New York, NY, USA, 1–9.

NIENHUYS, H.-W., AND STAPPEN, A. F. V. D. 2001. A surgery simulation supporting cuts and finite element deformation. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer-Verlag, London, UK, 145–152.

NIENHUYS, H.-W., AND VAN DER STAPPEN, A. F. 2000. Combining finite element deformation with cutting for surgery simulations. In *Proceedings of Eurographics Short Presentations '00*, 43–52.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH*, 137–146.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH*, 291–294.

PIETRONI, N., GANOVELLI, F., CIGNONI, P., AND SCOPIGNO, R. 2009. Splitting cubes: a fast and robust technique for virtual cutting. *Vis. Comput. 25*, 3, 227–239.

POPINET, S. 2003. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys. 190*, 2, 572–600.

SAMPATH, R., AND BIROS, G., 2009. A parallel geometric multigrid method for finite elements on octree meshes. In review, available online at http://www.cc.gatech.edu/grads/r/rahulss/.

SERBY, D., HARDERS, M., AND SZÉKELY, G. 2001. A new approach to cutting into finite element models. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer-Verlag, London, UK, 425–433.

SHI, L., AND YU, Y. 2004. Visual smoke simulation with adaptive octree refinement. *Computer Graphics and Imaging*, 1319.

SHI, L., YU, Y., BELL, N., AND FENG, W.-W. 2006. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. 25*, 3, 1108–1117.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 73–80.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 81–90.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 63–72.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proceedings of SIGGRAPH*, 269–278.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proceedings of SIGGRAPH*, 205–214.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., AND VOLINO, P. 2005. Collision detection for deformable objects. *Computer Graphics Forum 24*, 1, 61–81.

WANG, W. 2000. Special bilinear quadrilateral elements for locally refined finite element grids. *SIAM J. Sci. Comput. 22*, 6, 2029–2050.

WICKE, M., BOTSCH, M., AND GROSS, M. 2007. A finite element method on convex polyhedra. In *Proceedings of Eurographics*.

WU, X., AND TENDICK, F. 2004. Multigrid integration for interactive deformable body simulation. In *Proceedings of International Symposium on Medical Simulation*, 92–104.

WU, X., DOWNES, M. S., GOKTEKIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Proceedings of Eurographics*, 349–358.