Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2011)
J. Bender, K. Erleben, and E. Galin (Editors)

Interactive High-Resolution Boundary Surfaces for Deformable Bodies with Changing Topology

Jun Wu[†1,2], Christian Dick[‡1], and Rüdiger Westermann[§1]

[1] Computer Graphics & Visualization Group, Technische Universität München, Germany
[2] State Key Lab of Virtual Reality Technology and Systems, Beihang University, China

Abstract

Recent work has demonstrated that composite finite-elements provide an effective means for physically based modeling of deformable bodies. In this paper we present a number of highly effective improvements of previous work to allow for a high-performance and high-quality simulation of boundary surfaces of deformable bodies with changing topology, for instance, due to cuts and incisions. Starting at a coarse resolution simulation grid, along a cut we perform an adaptive octree refinement of this grid down to a desired resolution and iteratively pull the fine level finite-element equations to the coarse level. In this way, the fine level dynamics can be approximated with a small number of degrees of freedom at the coarse level. By embedding the hierarchical adaptive composite finite-element scheme into a geometric multigrid solver, and by exploiting the fact that during cutting only a small number of cells are modified in each time step, high update rates can be achieved for high resolution surfaces at very good approximation quality. To construct a high quality surface that is accurately aligned with a cut, we employ the dual-contouring approach on the fine resolution level, and we instantly bind the constructed triangle mesh to the coarse grid via geometric constrains.

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

1. Introduction

In numerical simulations using composite finite-elements the basis functions of the finite-elements on a coarse grid are assembled via linear combinations of basis functions on a finer grid. Composite finite-elements allow approximating the dynamics of a set of fine level elements by one single coarse element that reflects the portion of material that is covered by the fine elements. Such elements have originally been invented to resolve complicated material micro-structures with only few degrees of freedom (DOFs) and to enable multigrid methods to effectively represent complicated object boundaries at ever coarser scales [HS97, SW06, PRS07, LPR*09]. To respect physically disconnected parts of the simulation domain on the coarse grids, the connectivity between the embedded elements can be considered to generate one coarse grid element for every part [ABA00, MBF04].

In computer graphics, composite finite elements have been used as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation, i.e., by Nesme et al. [MKJF09], and just recently to model material discontinuities that are caused by cuts and incisions, for instance, by Jeřábková et al. [JBB*10] and Dick et al. [DGW11]. While Dick et al. take advantage of composite finite-elements to improve the convergence of a multigrid solver in situations where material discontinuities are covered by the coarse grid cells, Jeřábková and co-workers employ the underlying principle to allow simulating complicated topologies at a much coarser resolution than the finest structures.

In this paper we propose combining the strengths of

† wujun@me.buaa.edu.cn
‡ dick@tum.de
§ westermann@tum.de

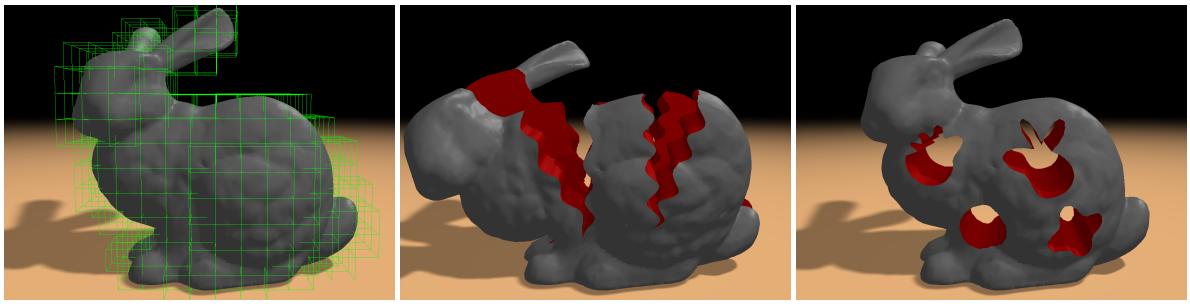

Figure 1: *Left: The coarse hexahedral simulation grid and the high-resolution surface that is constructed from the embedded fine grid. Middle and Right: The deformation of the body after complex sin wave and stamp-like cutting have been performed. Cutting, surface construction, and deformation simulation are performed at 12 simulation frames per second.*

previous composite finite-element methods for cutting deformable bodies. In particular, we present a combination of the adaptive octree refinement with iterative composite element hierarchy to enable simulating high-resolution cuts with a small number of DOFs. To enable sharp creases when cutting into the material, and to generate a high-quality boundary surface along a cut, we propose using the dual-contouring approach [JLSW02]. Figure 1 demonstrates the quality and speed that can be achieved by using a low-resolution composite finite-element simulation grid (top) in combination with an embedded adaptive grid at which a high-resolution surface is constructed.

The specific contributions of our paper are:

- A combination of adaptive top-down refinement and iterative fine-to-coarse compositing finite-elements to enable high resolution cuts and efficient computation of the coarse grid equations.
- Acceleration techniques to further improve the computational efficiency of equation composition, i.e., usage of an iterative coarsening approach to exploit the fact that many interpolations weights are zero and thus to reduce the number of arithmetic operations, and employment of a look-up-table for the first coarsening step.
- A new approach for constructing a high-resolution, high-quality surface that accurately aligns with a cut and reduces the number of surface elements.
- An incremental update scheme for both the coarse grid finite-elements and the fine grid render surface to achieve high simulation rates.

The remainder of our paper is organized as follows: First, we will acknowledge work that is related to ours. Next, we will briefly describe the hierarchical octree refinement of a hexahedral grid along a cut, and the reconstruction of a high-resolution render surface from the fine grid. In Section 4, the composite finite-element approach in combination with an efficient assembly of the coarse grid equations from the fine grid equations is outlined. In Section 5, we introduce the multigrid solver for the linear system of equations of composite finite element. Finally, detailed timing and memory statistics are provided, and the paper is concluded with some ideas for further work.

2. Related Work

In computer graphics a number of approaches for cutting deformable models have been proposed over the past years. Early work has focused on the adaptive refinement of tetrahedral elements along a cut surface [BMG99, MK00]. This work has then be extended towards the avoidance of ill-shaped elements via mesh alignment [NS01, SHS01, SOG06] and element deletion [CDA00, FDA02], and the reduction of the number of newly created elements by using pre-computed refinement patterns [BG00, BGTG03, GCMS00]. The virtual nodes algorithm [MBF04] also performs tetrahedral subdivision, but it uses a coarse grid in which the cells that are cut are duplicated to perform the deformable body simulation. An extension that allows for arbitrarily shaped and high resolution cut surfaces was presented in [SDF07].

A cutting approach using polyhedral subdivision was proposed in [WBG07, KMBG08], which also performs the deformable body simulation on a polyhedral model representation. The use of extended finite elements [BB99], which are enriched by additional basis functions to capture material discontinuities, was proposed in [AH08, JK09, KMB*09].

Recently, cutting of deformable bodies using hexahedral elements was proposed in Jeřábková et al. [JBB*10], Dick et al. [DGW11], and Seiler et al. [SSSH11]. The first and third approach build upon composite finite-elements to reduce the number of DOFs to be solved for, and the second approach approach uses composite elements to improve the convergence of a multigrid finite-element solver.

Jeřábková et al. model a cut via element removal on the finest resolution level. The boundary surface along the cut is reconstructed via the Marching Cubes algorithm. This approach, however, limits the resolution at which a cut can be performed and, in general, makes it difficult to construct a

surface that accurately aligns with a cut. Dick et al., on the other hand, perform an adaptive octree subdivision along a cut, restricting the high resolution grid to those regions where it is required. To accurately align the surface with a cut and to allow sharp creases when cutting into stiff material, the splitting cubes algorithm [PGCS09] is employed. Seiler et al. propose a method that allows decoupling the resolution of a material surface from the resolution of the simulation grid, but since the method is restricted to non-progressive cuts using a volumetric blade, the construction of a boundary surface is extremely simplified and stamping rather than cutting is simulated.

3. Geometry and Topology Representation

Our method for cutting deformable objects is particularly designed to allow for interactive simulation update rates and, at the same time, to allow for a detailed modeling of fine cuts, including the adaptation of a high-resolution render surface for a high-quality rendering. To achieve this, we use a high-resolution model to represent the geometry and topology of the deformable body, which is coupled with a lower-resolution finite element model for the simulation of this body. The geometry and topology representation consists of a volume representation and a surface representation, and will be explained in this section. The coupling of the two models will be explained in the Section 4.

3.1. Volume Representation

To model cuts in the deformable body, we use a linked volume representation as it was initially proposed by Frisken-Gibson [FG99]. The basic idea of the linked volume representation is to decompose the object into a set of hexahedral cells, using a uniform hexahedral grid. Face-adjacent cells are connected via links, with six links emanating from each cell. Cuts are modeled by marking links as disconnected when they are intersected by the virtual cutting blade (see Figure 2). Cuts are thus represented at the resolution of the hexahedral grid.

Since the resolution of a uniform grid is in practice limited by memory requirements, we use an adaptive octree grid as proposed by Dick et al. [DGW11], 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 cells 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 cells (possibly only one octree cell, when both endpoints of the link are lying within the same cell) 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. Cells 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 cells sharing a vertex, an edge, or a face is at most one (restricted octree). Cuts are modeled analogously to the modeling of the object surface, i.e., cells are adaptively refined along a cut down to the finest level, where links are marked as disconnected (see Figure 4 for an example). Material properties such as Young's modulus and density are assigned on a per-cell basis. To model inhomogeneous materials, the octree can further be refined. For a more detailed description, we would like to refer the reader to [DGW11].

3.2. Surface Representation

To render the boundary surface of the deformable object—including the additional surface parts that are generated by a cut—a surface mesh is reconstructed from the volume representation. Since the object boundary is represented by the cells at the finest level, the surface reconstruction is performed at this level. The meshing procedure has to consider that sharp surface features are generated by a cut and should be reproduced (see Figure 1 for an example). Furthermore, the procedure should be able to efficiently establish a correspondence between the surface vertices and the vertices of the simulation grid (see Section 4). Given this correspondence, the surface vertices can be bound to the simulation vertices and move according to the object deformations, eliminating the need to reconstruct the surface in every frame from the deformed volume configuration.

3.2.1. Surface Meshing

In our implementation the surface is reconstructed via the dual contouring approach [JLSW02]. The dual grid we use is the adaptive grid that is formed by the links between the cells at the finest level. For each link that is cut by the blade,

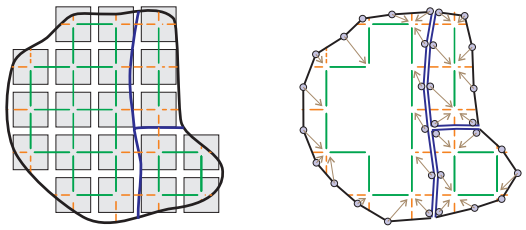

Figure 2: *2D Illustration of surface reconstruction and binding of the resulting surface vertices. Left: Linked volume representation of an object, consisting a set of simulation nodes (gray) which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary (black) and the newly generated surface due to cutting (blue). Right: Surface reconstruction from the grid that is formed by links. The binding of resulting surface vertices (blue dot) to the simulation nodes is indicated by the brown arrows.*

the distances between the intersection point to the link's endpoints as well as the normal of the blade at the intersection point are stored. Compared to the splitting cubes algorithm [PGCS09], which was used in [DGW11], the dual contouring approach reduces the number of triangles by a factor of four. The reason is that the vertices on the links are not used in the triangulation of the boundary surface.

In dual contouring, for each cell that contains a "feature" a representative vertex is positioned at this feature. In our application a feature is indicated by at least one disconnected link in the dual cell. The position of the representative vertex is where the quadratic function

$$E = \sum_i (n_i \cdot (x - p_i))^2 \quad (1)$$

has a minimum. Here, x denotes the vertex position, and p_i and n_i are the positions and normals at the intersections of the boundary surface with the links. As described before, the positions of these intersections on a link as well as the normals at these points are stored whenever a link is cut by the blade. Thus, the quadratic function can be evaluated in turn. As proposed in the original dual contouring algorithm, we then connect representative vertices in adjacent cells to construct the surface mesh.

Another advantage of the dual contouring algorithm is the quality of the resulting surface mesh. In the splitting cubes algorithm, the interior vertex is determined for the first cut by averaging face vertices or minimizing a quadratic function, depending on the angle formed by cutting planes. For successive cuts, the interior vertex is projected to a new cut plane. Thus it is processing order dependent. Our implementation results (see Figure 3) show that the dual contouring approach produces better surface meshes than the splitting cubes algorithm. Note that to handle singularity situations where cut planes are almost parallel, in our implementation of splitting cubes, the position of the interior vertices is explicitly clipped to ensure it is inside the cell, whereas in dual contouring, the singularity is handled by clipping singular values during solving the minimization of the quadratic function [JLSW02].

So far, the meshing procedure generates one surface part for every cell that is cut. However, a cut divides the material into multiple disconnected parts, for each of which a boundary surface has to be computed. To achieve this, we duplicate the representative vertices in the interior of each cell that is cut, as many times as there are disconnected material parts in this cell. As illustrated in Figure 2, each duplicated vertex—associated to a particular disconnected part—is bound to the nearest vertex of the respective part in the dual cell containing this vertex.

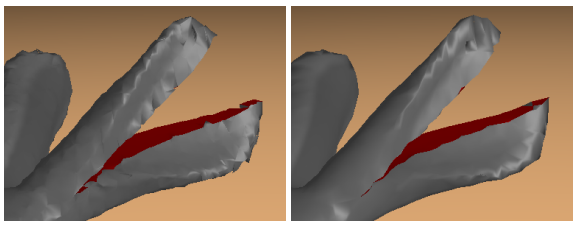

Figure 3: *A comparison of mesh quality of the splitting cubes algorithm (left) and the dual contouring approach (right).*

To accelerate the meshing procedure, the number of duplicated copies and the connectivity of an interior vertex to a cell's vertices are precomputed and stored in a look-up-table. This table has 2^{12} entries, each of which corresponds to one cutting pattern of the cell. At runtime, for a given representative vertex, a number of disconnected material parts, and a classification of the cell vertices to the parts the binding of the center vertices can be determined efficiently.

3.2.2. Mesh Deformation

To let the reconstructed surface move according to the object deformations, we use the binding that has been established between the surface vertices and the dual grid vertices. Since every dual grid vertex corresponds to exactly one primal grid cell, i.e., the fine grid cells, every surface vertex is bind to exactly one primal cell. Thus, the deformation of these cells—which is computed from the deformations of the simulation grid as described later on—is carried over to the surface vertices via tri-linear interpolation of the primal cell vertices.

4. Finite Element Simulation

A major design goal of the presented approach is to achieve interactive simulation update rates. Starting from the adaptive octree grid described in the previous section, one way to create a finite element model would be to create a hexahedral element with tri-linear shape functions for each octree cell, with three degrees of freedom (DOFs) located at each non-hanging vertex. However, since we employ a high-resolution octree grid to accurately represent complicated cuts and the boundary of the object, this approach would lead to a very high number of DOFs, far exceeding the number of DOFs that can be simulated at interactive update rates.

In this paper, we use composite finite elements [HS97, SW06] to reduce the number of DOFs and thus to achieve interactive update rates. A composite finite element is obtained by combining multiple 'standard' elements into a single element. In particular, the shape functions of the composite finite element are assembled from the shape functions of the individual elements.

We use a hexahedral composite finite element model, which is assembled from a 'standard' finite element model with one hexahedral element per octree cell. In the following we will use the term 'composite element' and 'hexahedral element' to refer to the elements of the respective model. Similar to the work by Preusser et al. [PRS07, LPR*09], we use

an inverse compositing scheme in that we start with tri-linear shape functions on the composite finite elements, and define the tri-linear shape functions of the hexahedral elements via restriction of the composite element shape functions to the domains of the individual hexahedral elements. Note that the DOFs of the hexahedral elements remain located at the vertices of the composite elements. Since it is explicitly allowed that a composite element is only partially filled with hexahedral elements, objects with complicated boundaries can be effectively discretized using a small number of elements.

In addition to the handling of complicated boundaries, another challenge is to handle complicated topologies in a coarse simulation model. In the original works by Preusser et al., the composite finite elements are assembled from the hexahedral elements by only considering their spatial location, disregarding the connectivity between the elements. For our application of interactive cutting, this would mean that material parts separated by a cut would possibly be merged into the same composite element, preventing the opening of the cut in the simulation.

We therefore employ a strategy to build a composite finite element model that precisely represents the topology of the object. Our strategy is based on analyzing the connectivity between elements to possibly create more than one composite element at the same location, each representing a separated material part. A similar approach has been used by Nesme et al. [MKJF09] and Jeřábková et al. [JBB*10] to model separated material parts in coarse finite element models, and by Aftosmis et al. [ABA00] and Dick et al. [DGW11] to represent separated material parts in the multigrid hierarchy of a geometric multigrid solver.

Our simulation is based on the linear theory of elasticity. To accurately simulate deformations with large rotations, we use the corotational formulation of strain.

4.1. Construction of the Simulation Model

The composite finite element model is based on a grid which is significantly coarser than the adaptive octree grid used to represent the volume of the deformable object. We build this coarser grid from the adaptive octree grid by successively removing leaf nodes from the associated octree. Note that the current set of leaf nodes corresponds to the current set of grid cells of the adaptive grid. In our implementation, we remove all nodes from octree levels $0,\dots,\ell-1$, where level number 0 denotes the finest level of the octree. In this way we obtain a coarsened adaptive octree grid for the composite finite element model, where the cells on the finest level coincide with blocks of $(2^\ell)^3$ cells on the finest level of the initial grid. It is worth noting that instead of using this uniform coarsening strategy, it would also be possible to use an adaptive strategy for removing octree nodes, for example to use a higher simulation accuracy in certain regions of interest.

The basic idea underlying the construction of the composite finite element model is to analyze the connectivity of the material within each coarse grid cell, and to create an individual composite element for each connectivity component. In this way, separated material parts are modeled by different elements, enabling the opening of a cut in the simulation.

The topologies of the composite finite element model and the hexahedral finite element model are each represented by an undirected graph, where the nodes corresponds to the elements, and the edges specify the connectivity between elements. For the hexahedral finite element model, the graph is directly obtained from the linked volume representation (see Figure 4).

The construction of the composite finite element model is performed in a two-step process. In the first step, we create the composite finite elements by considering the subgraph of the hexahedral finite element model induced by each coarse grid cell. We determine the connectivity components of this subgraph by using a depth-first search, and for each connectivity component, we create one composite element, which exactly subsumes the hexahedral elements corresponding to the nodes of this connectivity component. In the second step, the connectivity between the composite finite elements is determined. Two composite elements are connected, if there exists two connected hexahedral elements such that one hexahedral element is merged into the first, and the other into the second composite element. Finally, a shared vertex representation is computed for the composite finite element model. Two connected elements share a common face, and in particular the vertices incident to this face.

Instead of creating the target resolution of the composite finite element model directly from the hexahedral finite element model, we use an iterative coarsening approach, in that we iteratively apply the described scheme by only removing the leaf nodes of the finest octree level in each step (i.e., $\ell = 1$) (see Figure 4).

4.2. Computation of Element Matrices

We assemble the stiffness and mass element matrices for the composite finite elements from the stiffness and mass element matrices of the underlying hexahedral finite elements. The deformation behavior of the deformable body is described by the physical principal of total potential energy minimization, applied to each single point in time. For a standard hexahedral finite element discretization, the total potential energy E is

$$E(u) = \frac{1}{2}u^{\mathrm{T}}Ku - (f - M\ddot{u})^{\mathrm{T}}u, \tag{2}$$

where u is the linearization of the displacement vectors at non-hanging vertices of the hexahedral finite element grid, f is the linearization of the external force vectors applied at these vertices, and K and M denote the global stiffness and mass matrix, respectively. Minimizing this energy via

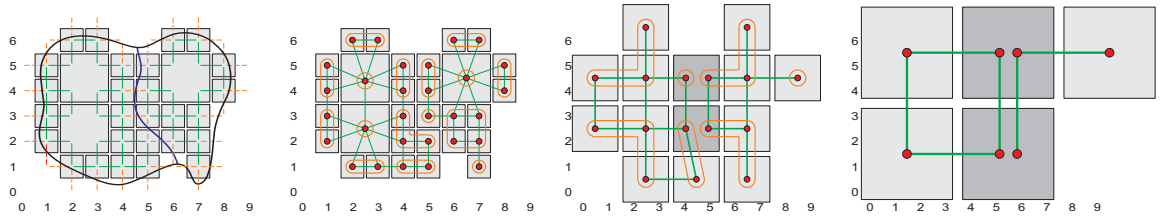

Figure 4: *Hierarchical construction of the composite finite element model. Left: 2D illustration of the adaptive linked volume representation, consisting of a set of rendering nodes (gray) which are connected via links (green). These links are disconnected (dashed, orange) at the object's original boundary (black) and along the newly generated surface due to cutting (blue). Middle left to right: Iterative coarsening of the finite element model. The underlying graph representation is indicated by red vertices and green edges. For each block of 2^3 cells on the respectively finest level the connected components (orange) are determined, and the elements of each connected component are replaced by a separate composite finite element.*

$\frac{\partial}{\partial u}E(u) = Ku - (f - M\ddot{u}) = 0$ leads to the well-known spatially discretized equation of motion.

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 tri-linear interpolation from the vertices of the composite finite elements. This is described by the equation $u = I\tilde{u}$, where $\tilde{u}$ denotes the linearization of the displacements at the vertices of the composite finite element grid, and the interpolation matrix I expresses the tri-linear interpolation from these vertices.

For a composite finite element discretization, the total potential energy E thus is

$$E(I\tilde{u}) = \frac{1}{2}\tilde{u}^{\mathrm{T}} I^{\mathrm{T}} K I \tilde{u} - (f - MI\ddot{\tilde{u}})^{\mathrm{T}} I\tilde{u}. \tag{3}$$

This energy is minimized via

$$\frac{\partial}{\partial \tilde{u}}E(I\tilde{u}) = \underbrace{I^{\mathrm{T}}KI}_{\tilde{K}}\tilde{u} - (\underbrace{I^{\mathrm{T}}f}_{\tilde{f}} - \underbrace{I^{\mathrm{T}}MI}_{\tilde{M}}\ddot{\tilde{u}}) = 0. \tag{4}$$

Thus, the global stiffness and mass matrix for the composite finite element discretization are obtained via $\tilde{K} = I^{\mathrm{T}}KI$ and $\tilde{M} = I^{\mathrm{T}}MI$, respectively, and the external forces which are specified on the underlying hexahedral finite element discretization are propagated to the composite finite element discretization via $\tilde{f} = I^{\mathrm{T}}f$.

Applying these equations to a single composite finite element c, its element matrices $\tilde{K}^c$ and $\tilde{M}^c$ can be assembled from the element matrices $\tilde{K}^e$ and $\tilde{M}^e$ of the underlying hexahedral elements e that are merged into the considered composite element via

$$\tilde{K}^c_{mn} = \sum_{e \text{ in } c} \sum_{i=1}^{8}\sum_{j=1}^{8} w^{c\to e}_{m\to i} w^{c\to e}_{n\to j} K^e_{ij}, \quad m,n = 1,\dots 8 \tag{5}$$

($\tilde{M}^c$ is computed analogously). 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, the notation K^e_{ij} denotes a 3×3 block of scalar entries.

The tri-linear 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

$$w^{c\to e}_{m\to i} = (1 - \frac{|x^c_m - x^e_j|}{s^c})(1 - \frac{|y^c_m - y^e_j|}{s^c})(1 - \frac{|z^c_m - z^e_j|}{s^c}), \tag{6}$$

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.

Instead of creating the target resolution of the composite finite element model directly from the hexahedral finite element model, it is also possible to iteratively apply the described scheme by only removing the leaf nodes of the finest octree level in each step (i.e., $\ell = 1$). In this case, many of the interpolation weights $w^{c\to e}_{m\to i}$ are zero, which effectively reduces the number of arithmetic operations needed for assembling the composite finite element matrices, and leads to a speed-up of a factor of about 1.5 compared to the direct approach.

Considering the element matrices of the underlying hexahedral finite elements, since all elements have the same shape, these elements can be generated from a single element stiffness and mass matrix. Let K^e and M^e denote the element stiffness and mass matrix for a generic element of side length 1, then the element stiffness and mass matrix for an element of side length s is sK^e and s^3M^e, respectively. Furthermore, the element stiffness and mass matrix scales linearly with the Young's modulus and the material density, respectively.

The fixation of the object is also specified on the underlying hexahedral finite element discretization. We propagate this fixation to the composite finite element discretization by fixing exactly those vertices on the composite element grid which are related to a fixed vertex on the underlying hexahedral element grid via a non-zero interpolation weight.

In case of simulating an object consisting of homogeneous material, it is possible to accelerate the assembly of the composite element matrices by employing a look-up table for coarsening of the finest level of the adaptive octree grid. Consider a coarse grid cell with a domain of 2^3 fine grid cells, we pre-compute the composite element matrices for all possible patterns of fine grid cells being empty or filled with material. Therefore, this look-up table has a size of $2^{(2^3)} = 256$ entries, and a size of 1.1 MB. By using the look-up table, the assembly of the composite finite element matrices is accelerated by a factor of about 2.

In our implementation, the composite finite element matrices are reused between successive simulation time steps. Since a progressive cut in a single time step only affects a very small portion of the model, in each time step only a small number of composite element matrices have to be (re-)assembled, which greatly reduces the computational costs per time step.

4.3. Corotational Strain Formulation

To accurately simulate deformations with large rotations using the linear theory of elasticity, we employ the corotational formulation of strain [RB86, HS04]. Since in the linear theory a linear strain tensor is employed, which interprets rotations as strains, the solution significantly diverges from the correct solution with increasing rotations within the deformation. The basic idea underlying the corotational formulation of strain is to remove the rotations before the linear strain is computed, and later re-add these rotations to the resulting stresses. The corotational formulation is based on the finite element discretization by computing a single rotation for each finite element.

In our approach, we employ the corotational strain formulation on the composite finite element discretization. To determine the rotation of a composite finite element c, we first compute the average deformation gradient $\overline{\nabla\varphi}^c$ via

$$\overline{\nabla\varphi}^c = \Big(\sum_{e \text{ in } c} \int_{\Omega^e} \nabla\varphi \, dx\Big) \,/\, \Big(\sum_{e \text{ in } c} \int_{\Omega^e} 1 \, dx\Big), \qquad (7)$$

where $\varphi(x) = x + u(x)$ denotes the deformation, and Ω^e denotes the domain of element e. The rotation matrix is then obtained by polar decomposition of the average deformation gradient [Hig86]. Note that our formulation of deformation gradient considers that a composite finite element might be only partially filled with hexahedral elements. Otherwise, an incorrect rotation of the composite finite element may lead to numerical instability.

5. Geometric Multigrid Solver

Applying the Newmark time integration scheme to the spatially discretized equation of motion leads to a linear system of equations in every simulation time step. We solve this system using the geometric multigrid approach proposed by Dick et al. [DGW11].

For the considered application of cutting of deformable objects, the challenge is to build a multigrid hierarchy that represents the arising complicated mesh topologies on the coarser levels. In particular, ignoring the topology and simply merging cells based on their spatial location would lead to a significant reduction of the convergence rate compared to the case without cuts. The approach of Dick et al. [DGW11] uses a strategy for generating a multigrid hierarchy that reflects the cuts on the coarser levels which is similar to the strategy proposed in this paper for generating the composite finite element grid from the underlying hexahedral element grid. For details we would like to refer the reader to the original work.

In contrast to the original work, where the multigrid solver is applied to an adaptive octree finite element discretization, in this paper the solver is applied to a composite finite element discretization. We use 3 V-cycles per time step, with 1 pre- and 1 post-smoothing Gauss-Seidel steps, using an overrelaxation parameter of 1.7. On the coarsest level, a Cholesky solver is employed. For the model sizes presented in this paper, we use 2 or 3 grid hierarchy.

6. Results

In the following we analyze the potential of our proposed cutting approach for interactive applications like virtual surgery simulations. The major focus in this analysis is on the performance analysis of the composite finite-element approach and the quality analysis of the generated surfaces. For a thorough analysis of the convergence behavior of the multigrid finite-element approach, including a detailed comparison with alternative solvers, let us refer to [DGW11]. We present a number of experiments, all of which were run on a standard desktop PC with an Intel Xeon X5560 processor running at 2.80 GHz (a single core is used), 8 GB of RAM, and an NVIDA GeForce GTX 480 graphics card.

Table 1 shows a detailed performance statistics for interactive cutting of the Stanford bunny model (see Figure 1), the Armadillo model (see Figures 5 and 6), and the Filigree model (see Figure 7). The second group of columns indicates the size of the composite element and the finest hexahedral element. The size is unified for simplicity. The third group gives information of the models. The first column in this group gives the number of the DOFs we are solving before and after a cut has been performed. The increasing number of DOFs is caused by the duplication of simulation elements that are cut. The next column shows the hexahedral grid resolution to which the adaptive refinement corresponds and from which the surface mesh is extracted. The last two columns give and the number of triangles that are reconstructed, and the memory requirements of our approach.

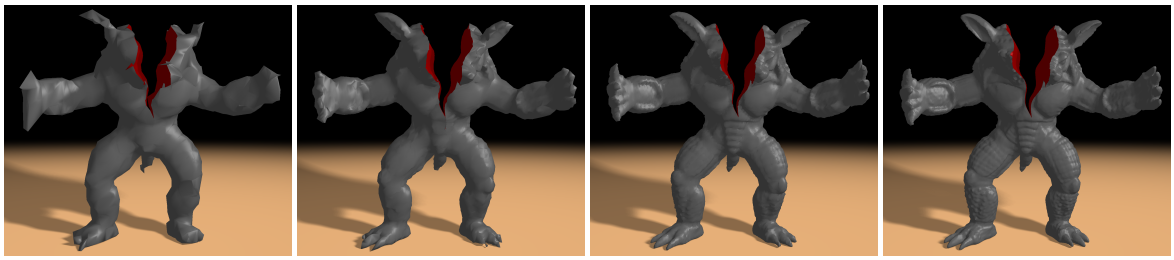

Figure 5: *Composite finite-element embedding of the armadillo model. In all images the resolution of the simulation grid is the same, but the hexahedral finite-element representation from which the stiffness matrices are assembled and the surface is reconstructed is adaptively refined along the cuts using 1, 2, 3, and 4 refinement levels, respectively, from left to right.*

Figure 6: *Left: The coarse hexahedral simulation grid and the high-resolution surface that is constructed from the embedded fine grid. Middle: Simulation using composite elements obtained from coarsening the hexahedral finite element model by three levels (10,563 DOFs). Each simulation time step takes 176.37 ms. Right: Simulation of cuts in the Armadillo model using hexahedral finite elements (569,748 DOFs). Each simulation time step takes 2829.87 ms.*

Model	Comp. size	Hex. size	# DOFs of simulation	Contour grid resolution	# Tris × 1k	Memory [MB]	Time [ms]						Speedup over full
							Cut	Contour	Compose	Solve	Intpl.	Total	
Armadillo	1	1/2	1,794 / 2,052	22×20×26	3	2.45	1.29	0.08	0.48	5.65	0.14	7.64	3.2×
Armadillo	1	1/4	2,205 / 2,439	43×39×51	12	13.48	3.56	0.33	2.55	6.63	1.07	14.14	6.1×
Armadillo	1	1/8	2,256 / 2,514	85×77×101	49	67.02	14.11	1.30	14.94	6.86	6.41	43.62	10.3×
Armadillo	1	1/16	2,289 / 2,547	169×153×201	197	311.68	44.86	5.23	71.23	7.07	28.53	156.92	17.8×
Armadillo	1/2	1/16	8,895 / 10,563	169×153×201	252	385.80	43.01	5.48	69.10	29.37	29.41	176.37	16.0×
Bunny	1	1/16	735 / 1,161	101×78×100	69	112.61	18.41	2.88	43.42	3.37	14.14	82.22	17.2×
Bunny	1/2	1/16	3,051 / 4,485	101×78×100	69	112.61	17.34	2.79	40.27	11.18	14.17	85.75	16.1×
Bunny	1	1/8	735 / 1,149	51×39×50	17	24.49	4.85	1.08	14.04	3.76	3.14	26.87	10.1×
Filigree	1	1/8	9,399 / 9,606	250×38×251	256	341.68	13.04	0.19	35.03	22.07	31.99	102.32	25.2×

Table 1: *Performance statistics for different models and different resolutions of the fine grid finite-elements.*

During cutting a new surface has to be reconstructed from the refined volume representation, and the resulting changes in the fine level elements, i.e., their stiffness matrices, have to be propagated to the coarse simulation level to rebuild the stiffness matrices at this level. Since cutting is always performed incrementally, meaning that in every frame the blade is moved only a small distance through the material, the aforementioned update operations have to be performed only locally around the cutting region. The timings required for incrementally performing the cut, reconstructing the surface, and updating the composite finite-elements are listed in the first part of group Time. In our examples, the cuts were realized such that in average the number of newly created fine level elements was about 3% of the total number of elements. From our experiences this seems to quite realistically simulate the speed at which a cut is performed in reality (see accompanied video). It is clear, on the other hand, that a faster cut can significantly increase the number of affected

elements per frame, resulting in vastly increasing simulation times. The second part of this group lists the times required for solving the system of equations on the coarse simulation level and interpolating the computed displacement at the surface vertices. The last column in this group gives the total simulation time per time step.

We compare in the last column the performance of our composite finite elements approach with the full simulation on the finest level as presented in [DGW11]. In this comparison, the incremental updating of surface and stiffness matrices is also applied to the full simulation. The speedup of overall performance depends on the refinement level. With three levels composition, the speedup of our method is about $10\times$ for Armadillo and Bunny models, and $25\times$ for the Filigree model.

Our statistics indicate that the proposed cutting approach can be used very effectively in interactive scenarios requiring high update rates and high-resolution boundary surfaces, at reasonable approximation quality. Even the highest resolution Armadillo model can be cut, simulated, and rendered at 5-6 fps on a single core of our target architecture. Especially the possibility to reproduce sharp material features induced by a cut at high quality distinguishes our approach from previous approaches.

To demonstrate the approximation quality of the composite finite-element approach, we have performed one additional test using the Armadillo model. The result of this test is shown in Figure 6. In the middle, the model is simulated using composite elements obtained from coarsening the hexahedral finite element model by three levels (10,563 DOFs), while on the right it is simulated using hexahedral finite elements (569,748 DOFs). The simulation time step takes 2829.87 ms for the full simulation and 180.99 ms for simulating the composite elements.

It can be seen that the composite finite-element approach yields a slightly different deformation than the full resolution approach, because a much lower number of DOFs is solved. On the other hand, even for the rather large disconnected parts that are generated by the cut, which undergo large displacements due to their high masses, the differences are not too severe. Especially in virtual surgery simulation, where typically only small cuts and incisions are performed by a doctor, these difference become much less significant.

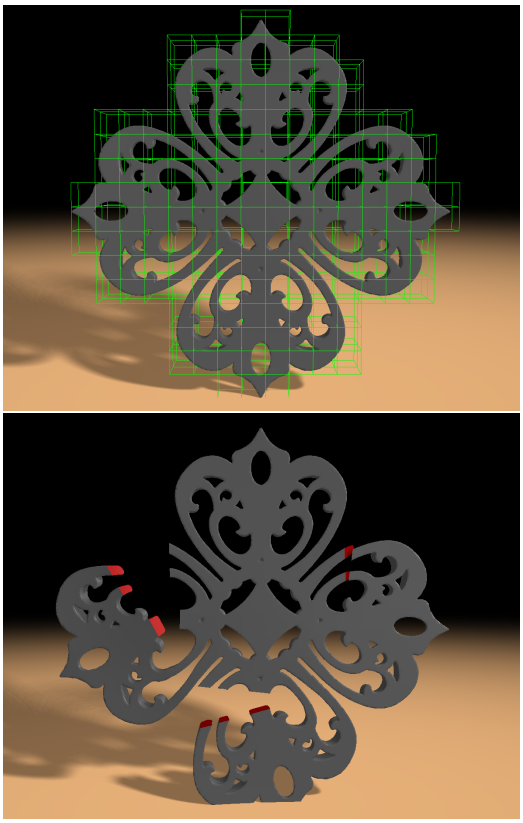

Figure 7: *Cutting on a Filigree model with complex topology. Top: The coarse hexahedral simulation grid. Bottom: The deformation of the body after cutting has been performed. Cutting, surface construction, and deformation simulation are performed at 10 simulation frames per second.*

7. Conclusion

We have presented several highly effective improvements to existing cutting approaches using composite finite-elements, with the goal to efficiently generate a high-resolution boundary surface from a much coarser simulation grid. By using composite finite-elements, in combination with an adaptive refinement of an embedded finite-element grid along the cuts, a very high resolution surface representation is achieved. Since we use the dual contouring approach for reconstructing this surface, sharp features are reproduced, and compared to previous approaches a significantly smaller number of triangles is generated. Even though the simulation accuracy depends on the number of DOFs we solve on the coarse simulation grid, by assembling the stiffness matrices on this grid from the matrices on the adaptively refined volume representation, topological changes can be simulated at very good approximation quality.

In the future we plan to integrate self-collision detection into our approach, using the high-resolution surface representation and propagating the resulting penalty forces to the simulation vertices. Furthermore, we will investigate the parallelization of the proposed approach on current multi-core desktop systems to achieve real-time performance even for very high resolution volume representations. Finally, we are actually working on the integration of a haptic feedback device into our simulation to evaluate the full potential of the proposed cutting approach in virtual surgery simulation.

8. Acknowledgements

Jun Wu is supported by the Erasmus Mundus TANDEM, an European Commission funded program.

References

[ABA00] AFTOSMIS M., BERGER M., ADOMAVICIUS G.: A parallel multigrid method for adaptively refined cartesian grids with embedded boundaries, aiaa 2000-0808. In *38th AIAA Aerospace Sciences Meeting and Exhibit* (2000). 1, 5

[AH08] ABDELAZIZ Y., HAMOUINE A.: Review: A survey of the extended finite element. *Comput. Struct. 86*, 11-12 (2008), 1141–1151. 2

[BB99] BELYTSCHKO T., BLACK T.: Elastic crack growth in finite elements with minimal remeshing. *Int. J. Numer. Methods Eng.*, 5 (1999), 601–620. 2

[BG00] BIELSER D., GROSS M.: Interactive simulation of surgical cuts. In *Pacific Graphics* (2000), pp. 116–125. 2

[BGTG03] BIELSER D., GLARDON P., TESCHNER M., GROSS M. H.: A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graphics* (2003), pp. 377–386. 2

[BMG99] BIELSER D., MAIWALD V. A., GROSS M. H.: Interactive cuts through 3-dimensional soft tissue. *Comput. Graph. Forum 18*, 3 (1999), 31–38. 2

[CDA00] COTIN S., DELINGETTE H., AYACHE N.: A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training. *The Visual Computer 16*, 7 (2000), 437–452. 2

[DGW11] 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. 1, 2, 3, 4, 5, 7, 9

[FDA02] FOREST C., DELINGETTE H., AYACHE N.: Removing tetrahedra from a manifold mesh. In *CA '02: Proceedings of the Computer Animation* (2002), p. 225. 2

[FG99] FRISKEN-GIBSON S. F.: Using linked volumes to model object collisions, deformation, cutting, carving, and joining. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 333–348. 3

[GCMS00] GANOVELLI F., CIGNONI P., MONTANI C., SCOPIGNO R.: A multiresolution model for soft objects supporting interactive cuts and lacerations. *Comput. Graph. Forum 19*, 3 (2000), 271–282. 2

[Hig86] HIGHAM N. J.: Computing the polar decomposition—with applications. *SIAM Journal on Scientific and Statistical Computing 7*, 4 (1986), 1160–1174. 7

[HS97] HACKBUSCH W., SAUTER S.: Composite finite elements for the approximation of pdes on domains with complicated micro-structures. *Numerische Mathematik 75* (1997), 447–472. 1, 4

[HS04] HAUTH M., STRASSER W.: Corotational simulation of deformable solids. In *Proceedings of WSCG* (2004), pp. 137–145. 7

[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. 1, 2, 5

[JK09] JEŘÁBKOVÁ L., KUHLEN T.: Stable cutting of deformable objects in virtual environments using xfem. *IEEE Comput. Graph. Appl. 29*, 2 (2009), 61–71. 2

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21* (2002), 339–346. 2, 3, 4

[KMB*09] KAUFMANN P., MARTIN S., BOTSCH M., GRINSPUN E., GROSS M.: Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. 28*, 3 (2009), 1–10. 2

[KMBG08] KAUFMANN P., MARTIN S., BOTSCH M., GROSS M.: Polyhedral finite elements using harmonic basis functions. *Comput. Graph. Forum 27*, 5 (2008), 1521–1529. 2

[LPR*09] LIEHR F., PREUSSER T., RUMPF M., SAUTER S., SCHWEN L. O.: Composite finite elements for 3d image based computing. *Comput. Vis. Sci. 12*, 4 (2009), 171–188. 1, 4

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3 (2004), 385–392. 1, 2

[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), pp. 598–607. 2

[MKJF09] MATTHIEU N., KRY P. G., JEŘÁBKOVÁ L., FAURE F.: Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28* (July 2009), 52:1–52:9. 1, 5

[NS01] NIENHUYS H.-W., STAPPEN A. F. V. D.: 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* (2001), pp. 145–152. 2

[PGCS09] PIETRONI N., GANOVELLI F., CIGNONI P., SCOPIGNO R.: Splitting cubes: a fast and robust technique for virtual cutting. *Vis. Comput. 25*, 3 (2009), 227–239. 3, 4

[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. 1, 4

[RB86] RANKIN C., BROGAN F.: An element-independent corotational procedure for the treatment of large rotations. *ASME J. Pressure Vessel Tchn. 108* (1986), 165–174. 7

[SDF07] SIFAKIS E., DER K. G., FEDKIW R.: Arbitrary cutting of deformable tetrahedralized objects. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), pp. 73–80. 2

[SHS01] SERBY D., HARDERS M., SZÉKELY G.: 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* (2001), pp. 425–433. 2

[SOG06] STEINEMANN D., OTADUY M. A., GROSS M.: Fast arbitrary splitting of deforming objects. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 63–72. 2

[SSSH11] SEILER M., STEINEMANN D., SPILLMANN J., HARDERS M.: Robust interactive cutting based on an adaptive octree simulation mesh. *The Visual Computer* (2011), 1–11. 2

[SW06] SAUTER S. A., WARNKE R.: Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing 77* (2006). 1, 4

[WBG07] WICKE M., BOTSCH M., GROSS M.: A finite element method on convex polyhedra. In *Proceedings of Eurographics* (2007). 2