CGI2013 manuscript No.
(will be inserted by the editor)

Efficient Collision Detection for Composite Finite Element Simulation of Cuts in Deformable Bodies

Jun Wu · Christian Dick · Rüdiger Westermann

Abstract Composite finite elements (CFEs) based on a hexahedral discretization of the simulation domain have recently shown their effectiveness in physically based simulation of deformable bodies with changing topology. In this paper we present an efficient collision detection method for CFE simulation of cuts. Our method exploits the specific characteristics of CFEs, i.e., the fact that the number of simulation degrees of freedom is significantly reduced. We show that this feature not only leads to a faster deformation simulation, but also enables a faster collision detection. To address the non-conforming properties of geometric composition and hexahedral discretization, we propose a topology-aware interpolation approach for the computation of penetration depth. We show that this approach leads to accurate collision detection on complex boundaries. Our results demonstrate that by using our method cutting on high resolution deformable bodies including collision detection and response can be performed at interactive rates.

Keywords Cutting · Deformable bodies · Collision detection · Composite finite elements

1 Introduction

The physically-based simulation of cuts in deformable bodies can significantly improve the realism in surgery simulators and computer games. To efficiently compute the deformation, composite finite elements (CFEs) [1] based on a uniform or adaptive hexahedral (octree) discretization of the simulation domain have recently been adopted in the computer graphics community [2,3]. The idea is to approximate the finite element discretization of the governing system of partial differential equations on a high-resolution grid by subsuming blocks of finite elements into coarser elements, thereby reducing the numbers of simulation degrees of freedom (DOFs) and thus trading performance for a moderate decrease of accuracy. Additionally exploiting the fact that during cutting only a small number of elements are modified in each simulation frame, [4] and [5] have further demonstrated that interactive update rates—12 fps including mesh cutting, surface reconstruction, and deformation computation—can be achieved for simulating progressive cuts in elastic bodies with an effective resolution of 100^3 hexahedral elements.

Besides the pure deformation simulation, efficient collision detection is another essential component in interactive virtual environments. For deformable bodies with changing topology, collision detection is particularly time consuming, since new geometric primitives are created on-the-fly. As a consequence of cutting, an object may be split into several separated objects. It is therefore necessary to consistently detect both inter- and intra-collisions. Moreover, a quantitative measure of the penetration is desired for robust collision response.

Considering the promising results of CFEs, it is highly interesting to investigate collision detection in this specific context. On the one hand, CFEs have the potential to simplify collision detection, thanks to their nature of reducing the number of finite elements. On the other hand, they present several challenges with respect to the underlying non-conforming geometric structures, i.e., the coarse composite elements (which might be duplicated to represent separated material parts on each side of a cut) versus the underlying hexahedral elements, and the hexahedral discretization versus a trian-

J. Wu (✉) · C. Dick · R. Westermann
Technische Universität München, Germany
E-mail: jun.wu@tum.de

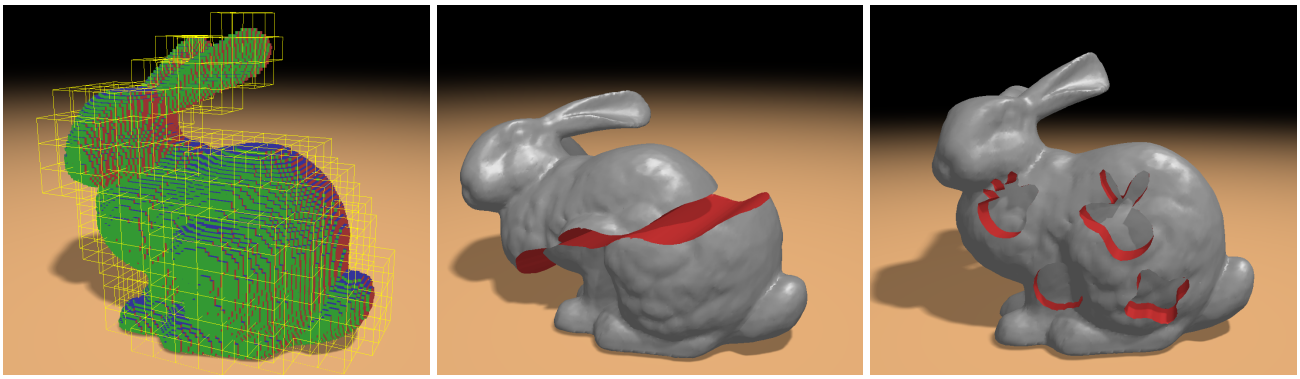

Fig. 1 Cutting of the Stanford Bunny model. Left: High resolution hexahedral elements and coarse composite elements (yellow grid). Middle: The deformable bunny is cut into two parts, resulting in the upper part sliding down. Right: After four stamp-like cuts, the resulting cylinders slide out under tight constraints. Collision detection and finite element simulation are performed at 41 fps. Interactive cutting of the deformable body takes additional 37 ms per simulation frame for updating the surface mesh, the stiffness matrices, and the distance field.

gulated surface mesh representation. The non-conformity is particularly severe in the case of cuts: The gap of a cut has an initial width of zero. In contrast, a naive collision detection approach purely based on hexahedral elements can only achieve an accuracy in the order of the hexahedral grid spacing.

In this paper we present an efficient collision detection method for interactive CFE simulation of cuts applied to high-resolution deformable bodies. Our method exploits the composition structure of CFEs to speed up the collision query, and addresses the non-conformity of the underlying geometric structures by analyzing the topology of the hexahedral element grid in order to improve collision accuracy. Our method detects both intra-object and inter-object collisions, and also supports application scenarios where one of the objects is not closed (e.g., a thin scalpel).

The specific contributions of our paper are:

- A collision detection algorithm which exploits the specific characteristics of CFEs. The complexity of our broad phase collision detection depends on the number of simulation DOFs, instead of the number of geometrical primitives. This leads to a speedup factor which scales exponentially with respect to the level of composition.
- A topology-aware interpolation scheme to determine the penetration depth on the non-conforming hexahedral element grid. By flipping the sign of distance values associated with hexahedral elements based on analyzing their topological connectivity, the accuracy of penetration depth interpolation around the boundary is improved.

The remainder of this paper is organized as follows: In the next section, we summarize work that is related to ours. In Section 3, we briefly review CFE simulation of cuts in deformable bodies. In Section 4, we then present our collision detection approach. In Section 5, we describe the distance field computation and updating. Detailed timing statistics and evaluations are provided in Section 6, and the paper is concluded in Section 7.

2 Related Work

Composite finite elements approximate a high resolution finite element discretization of a partial differential equation by means of a small set of coarser elements. Such elements have originally been invented to resolve complicated simulation domains with only a few degrees of freedom, and are also used in the context of geometric multigrid methods to effectively represent complicated object boundaries at ever coarser scales [1, 6–8].

In computer graphics, the idea of CFEs has been used as a special kind of homogenization for resolving complicated topologies and material properties in deformable body simulation [2], and just recently to model material discontinuities that are caused by cuts and incisions, for instance, to improve the convergence of a geometric multigrid solver [3] and to reduce the number of simulation DOFs in order to increase simulation performance [4, 5].

A high quality surface mesh reconstructed after cutting can serve as a good basis for collision handling. Jeřábková et al. [4] model a cut by removing elements on the finest resolution level, and reconstruct the boundary surface by means of the Marching Cubes algorithm. The element removal approach, in general, makes it difficult to construct a surface that accurately aligns with a cut. Seiler et al. [9] propose a method that decouples the resolution of the material surface from the resolution of the simulation grid, but the method is restricted to volumetric non-progressive cuts. Dick et al. [3] model

progressive cuts by disconnecting links between hexahedral elements, and reconstruct the surface by a splitting cubes algorithm [10], taking into account the exact intersections of the cutting blade and the links. Based on this work, Wu et al. [5] use CFEs to reduce the number of simulation DOFs, and propose a dual contouring approach [11] to construct a high quality surface which is accurately aligned with a cut.

Collision detection for general deformable bodies has been widely studied and an excellent survey is given by Teschner et al. [12].

To simulate deformable bodies with dynamically created geometric primitives, we are interested in collision detection methods that do not rely on heavy precomputation. Spatial hashing [13] requires no preprocessing of surface meshes, and detects both self-collisions and collisions between different bodies. Layered depth images (LDIs) [14–16] hold these properties as well, but are limited to closed manifold objects. Bounding volume hierarchies (BVHs) are optimized to handle dynamic topologies [17, 18], and can be combined with locally updated distance fields to simulate brittle fractures [19].

For simulating reduced deformations governed by only a few DOFs, James and Pai [20] have demonstrated that precomputed BVHs can be updated at a cost proportional to the number of simulation DOFs. Built on intensive precomputation of collision-free certificates, reduced structures have also been exploited for self-collision processing [21, 22]. These algorithms are designed for reduced deformable bodies with consistent topology. CFEs can be considered as a special kind of a locally reduced model.

To facilitate dynamically created geometric primitives and to detect both inter- and intra-collisions, for CFEs we propose to utilize spatial hashing in the broad phase of collision detection. The exact penetration information at complicated boundaries is then evaluated with a novel topology-aware interpolation scheme.

To compute the penetration depth for collision response, a practical workflow for interactive applications [13, 23–25] is to first find potentially overlapping pairs of a vertex and a volumetric element. The vertex is then transformed, with respect to this volumetric element, from the deformed configuration to the reference configuration, where the penetration depth and direction of the vertex is evaluated. Finally, the penetration information is transformed back to the deformed configuration and utilized to compute the collision force. High update rates can be achieved by using a precomputed distance field in the reference configuration.

3 Composite Finite Element Simulation of Cuts

Our collision detection method is particularly designed for CFE simulation, which has recently gained popularity to simulate cuts in medical applications [4, 5, 9]. In the following, we briefly summarize the main principles of CFE simulation of cuts in deformable bodies. For a detailed discussion, we refer the reader to [5].

For CFE simulation, three coupled geometric representations are employed to describe a deformable body (see Figure 2 for an illustration).

Hexahedral elements. The deformable body is discretized by means of trilinear hexahedral elements that are aligned on a restricted octree grid. This grid is adaptively refined along the surface (including cutting surfaces) of the object. To simplify the discussion, a uniform Cartesian grid is assumed in the following. Physical properties (i.e., Young's modulus and Poisson's ratio) are assigned on a per-element basis. The topology is represented by links between face-adjacent elements. Cuts are modeled by marking links as disconnected. For each link that is cut the intersection point with the cutting blade and the blade's normal at that point is stored. This linked hexahedral element representation is the basis for constructing the following two representations, which are used for efficient deformation computation and high quality visual rendering, respectively.

Composite elements. From the linked hexahedral elements, composite elements are constructed by considering $2^k \times 2^k \times 2^k$ blocks of hexahedral elements. In each block, we analyze the connectivity among the elements, and create one composite finite element for each connectivity component. In this way we create one composite finite element for the distinct material parts separated by cuts, and thus accurately represent cuts in the composite finite element discretization. As a consequence, multiple composite finite elements might exist at the same location. An example for duplicated composite elements is shown Figure 2, indicated by a darker cell color. From an implementation point of view, the composition process is performed in k iterations by successively considering $2 \times 2 \times 2$ blocks. While Figure 2 illustrates an example of one-level composition ($k = 1$), the composite elements can be several levels coarser than the hexahedral elements, thereby trading speed for a moderate decrease of simulation accuracy. Note that when the underlying hexahedral element grid is an adaptive octree grid, the composite element grid is also an adaptive octree grid.

The deformation computation is performed on the composite elements. From a mathematical point of view, this is achieved by starting with a finite element discretization on the fine hexahedral elements, and substi-

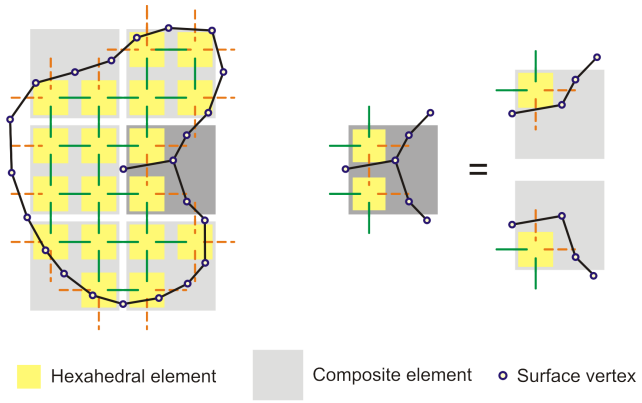

Fig. 2 A deformable body is represented by linked hexahedral elements (yellow), from which composite elements (gray) and a surface mesh (black) are constructed. Note that the gaps between elements are purely illustrative, i.e., in the finite element model, the elements are directly attached to each other. The dark gray color indiciates that more than one (here: two) composite elements exits at the same location, corresponding to the distinct material parts separated by the cuts. Links are shown in green and orange color if marked as connected and disconnected, respectively.

tuting the DOFs of the hexahedral elements by means of trilinear interpolation from the DOFs of the composite elements.

Surface mesh. By means of a dual contouring approach, a smooth surface triangle mesh is constructed on the dual grid consisting of the links between hexahedral elements by utilizing the intersection points and the normals of the cutting blade stored at these links [5]. The surface is bound to the hexahedral element model by assigning each vertex to its topologically connected, closest hexahedral element. In this way, the computed deformation of the finite element model can be transferred to the surface mesh.

The three representations are geometrically non-conforming, i.e., a composite finite element may be only partially filled with hexahedral elements, and the surface mesh arbitrarily intersects the hexahedral element grid.

In each simulation time step, the computed per-vertex displacements of the composite elements therefore are propagated to the hexahedral elements and then to the surface mesh vertices, both by means of trilinear interpolation. It is worth noting that a) parts of the triangle mesh associated with the same composite element cannot self-collide, since the coordinates of the surface mesh vertices are trilinearly interpolated from those of the same composite element, and that b) for a given vertex position, inferring between the trilinear interpolation weights with respect to the composite element and the trilinear interpolation weights with respect to an underlying hexahedral element is straightforward.

4 Collision Detection for CFE Simulation of Cuts

From an abstract point of view, we consider collision detection as querying whether a given vertex penetrates into a solid object, and if yes, reporting the penetration depth and direction to be used for computing the collision response. Since the querying vertex can be from the same object or another one, this abstraction handles both inter- and intra-collisions. Note that the vertex can also be from a non-closed object, e.g., a thin blade in surgery simulations.

Our collision detection approach consists of a broad and a narrow phase. In the broad phase (Section 4.1), potential overlaps between surface vertices and composite elements are detected. This phase is performed in the deformed configuration. Testing coarse composite elements instead of fine hexahedral elements leads to a significant speedup.

In the narrow phase (Section 4.2), we determine the penetration depth for each potentially colliding vertex by transforming the composite element/vertex pair determined in the broad phase into the reference configuration. Due to the trilinear interpolation between composite element DOFs and hexahedral element DOFs, we can directly determine the position of the vertex with respect to the hexahedral element grid. The penetration depth is interpolated from a signed distance field. Since distance field voxels on both sides of a cutting surface are classified as inside, i.e., are assigned to a negative distance value, a simple interpolation of the penetration depth for a vertex position close to a cutting surface is not accurate (e.g., the distance value of a vertex directly on the cutting surface should be zero, but since all surrounding voxels carry a negative distance value, a non-zero value would be returned). We address this problem by specifically flipping the sign of distance values based on considering the connectivity between hexahedral elements.

4.1 Broad Phase Collision Detection

To identify for each vertex the potentially overlapping composite elements, we employ the spatial hashing approach proposed by Teschner et al. [13]. The basic idea is to subdivide the 3D space using a uniform Cartesian grid. For each geometric primitive (vertices and composite elements), we determine the set of grid cells that are intersected by the primitive. If the sets of grid cells for two primitives are not disjunct, the pair of primitives is classified as potentially colliding and is further examined in the narrow phase. To efficiently represent the 3D grid in main memory, a hash table is employed.

Our broad phase collision detection consists of two passes. In the first pass, we traverse all composite elements. For each composite element, we construct an axis aligned bounding box in the deformed configuration. Since the surface mesh vertices are not strictly placed in the interior of the composite element, e.g., the mesh constructed using the sharp feature preserving dual contouring approach [11], we compute a bounding box which covers the composite element's eight vertices as well as all surface mesh vertices associated with this composite element (i.e., the vertices of the part of the surface mesh that belongs to the material part described by the composite element). The composite element's ID is stored in each grid cell that is intersected by the bounding box.

In the second pass, we traverse all surface vertices. For each vertex, we determine the grid cell that contains the vertex. The grid cell's list of composite elements ID generated in the first pass directly yields the composite elements that are potentially colliding with the considered vertex. Note that the composite element which the surface mesh vertex is associated with can be excluded from the list, since self-collisions of mesh parts associated with the same composite element cannot occur.

The parameters of spatial hashing significantly influence the performance of collision detection. We follow the optimized spatial hashing approach [13] to assign these parameters. The spacing of the uniform Cartesian grid is chosen equal to the size of the composite elements in the reference configuration. To reduce hash collisions, we employ the XOR hash function to obtain a hash value $H(x, y, z)$ from a grid cell's index (x, y, z):

$$H(x, y, z) = (x\, p_1 \oplus y\, p_2 \oplus z\, p_3) \mod n, \tag{1}$$

where $\oplus$ is the XOR operator, and the prime numbers p_1, p_2, p_3 are 73856093, 19349663, 83492791. The size n of the hash table is chosen to be eight times the number of composite elements.

4.2 Narrow Phase Collision Detection

To further examine potentially colliding vertex/composite element pairs, we transform the vertex back into the reference configuration. The penetration depth and direction for this vertex are then computed using a topology-aware interpolation scheme based on a signed distance field, and are later utilized for collision response.

The transformation of the vertex is based on its trilinear interpolation weights with respect to the deformed composite element. These can be computed by solving the interpolation equation system using Newton iteration [26]. From the trilinear interpolation weights, we can immediately determine the location of the vertex in the reference configuration. All further processing in the narrow phase is performed in the reference configuration.

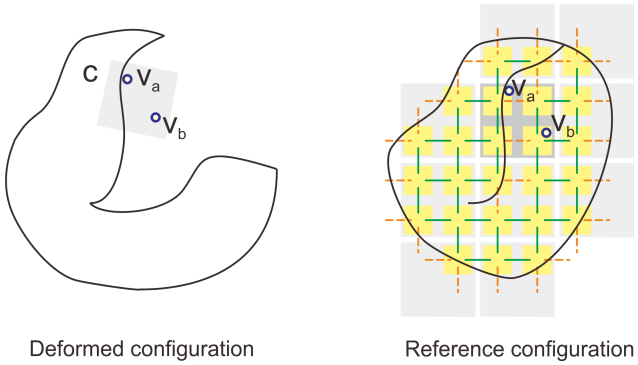

Fig. 3 Collision detection artifacts at cutting surfaces. In the deformed configuration (left), the vertices v_a and v_b are outside the object. Transformed back into the reference configuration (right), the vertices lie inside the object, although in a different material part with respect to the original query.

To determine the penetration depth and direction, we employ a signed distance field of the surface of the deformable body in the reference configuration. Each sample of the distance field consists of the signed scalar distance between the sample point and the nearest surface point, as well as the vector pointing from the sample to the nearest surface point (vector distance). The distance field is sampled at the grid cell centers of a uniform Cartesian grid, which is aligned with the hexahedral element grid (finest octree level). A negative (positive) distance value classifies the respective voxel as inside (outside). The spatial extent of the distance field must be chosen such that there is at least one layer of outside voxels. If in the following a more distant outside voxel is queried, any positive distance value can be returned (e.g., 1.0).

To determine whether the considered vertex penetrates the object, and at the same time obtain the penetration depth and direction for this vertex, a simple approach would be to sample the distance field by means of trilinear interpolation of the adjacent (surrounding) eight voxels, and to consider the vertex as penetrating, iff the obtained distance value is negative. However, this simple interpolation produces artifacts at cutting surfaces, where separated material parts are directly adjacent in the reference configuration. This situation is illustrated in Figure 3. Considering vertex v_a or v_b, all adjacent voxels have negative distance values, resulting in a negative interpolated distance value and thus in the (incorrect) classification of the vertex as penetrating. Note that the error can easily be larger than the size of a voxel/hexahedral element, as is the case for

vertex v_b. The approach can be extended by determining the hexahedral element where the considered vertex is lying in. If this hexahedral element topologically does not belong to the composite element, the vertex is classified as non-penetrating. Otherwise, the distance field is trilinearly interpolated and the classification is performed based on the sign of the resulting distance value as before. This approach limits the error to half the size of a voxel/hexahedral element, but due to the clamping of the 'inside' region at boundaries of hexahedral elements, it leads to a jagged object surface as seen by the collision handling algorithm (see Figure 5). Further results obtained by the described algorithm—in the following referred to as 'simple interpolation approach'—are presented in the results section.

The problem with the simple interpolation approach is that the topology of the deformable object is not considered, i.e., the trilinear interpolation of the distance field incorporates voxels that are 'inside' with respect to some material connectivity component, but 'outside' with respect to the specific material connectivity component that is associated with the considered composite element. Our approach addresses this problem by first classifying the eight interpolation voxels as inside or outside with respect to the material connectivity component that is associated with the considered composite element, and then temporarily switches the sign of the distance value of 'outside' voxels, if this value is negative. This topology-aware interpolation approach virtually completely eliminates errors due to the geometrical non-conformity of the hexahedral grid and the triangle surface mesh, and leads to a smooth object surface as seen by the collision handling algorithm.

The individual steps of our algorithm are as follows:

1. Starting from the eight surrounding interpolation voxels, determine the subset of those interpolation voxels for which the respective hexahedral element is existing and topologically belongs to the composite element.
2. Augment this subset by those interpolation voxels which are topologically connected to the interpolation voxels that are already contained in the subset.
3. The interpolation voxels in the subset are 'inside' voxels, the others are 'outside' voxels. Temporarily switch the sign of the distance value of 'outside' voxels to positive. Remark: The sign of the distance value of 'inside' voxels is negative, since each of these voxels corresponds to a hexahedral element (whose center is located inside of the deformable body). Distance field voxels outside of the simulation domain do not correspond to an hexahedral element, but already carry a positive distance value. Thus, no sign flipping is necessary for such voxels.

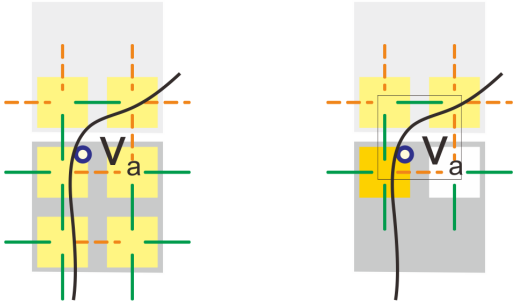

Fig. 4 Classification of voxels in our topological-aware interpolation approach. Left: We consider the lower composite element, more precisely, the left material part (note that there are two composite elements at the same location, indicated by the dark gray color). Right: From the four (in 2D) interpolation voxels of vertex v_a, one corresponds to a hexahedral element (orange) that topologically belongs to the considered composite element. Two further interpolation voxels correspond to hexahedral elements (yellow) that are topologically connected to the orange element. These three voxels are 'inside'. The remaining voxel (corresponding hexahedral element is shown in white color) is 'outside'—for this voxel, the sign of the distance value has to be switched to positive.

4. Trilinearily interpolate the distance value using the voxels' temporary distance values.

An example for the application of the algorithm is given in Figure 4. The effect of topology-aware interpolation is visualized in Figure 5. For a bunny model after cutting and deformation (left), we show the zero isocontour of the distance field returned by the simple interpolation approach (middle) and our topology-aware interpolation approach (right). Note that the zero isocontour corresponds to the object surface as seen by the collision handling algorithm. It is clearly visible that the topology-aware interpolation approach leads to a much smoother and much more precise isocontour.

To determine the penetration depth and direction in the deformed configuration, the closest surface point indicated by the interpolated vector distance is transformed into the deformed configuration. The penetration depth and direction then are given by the vector between the queried vertex and its transformed closest surface point.

5 Distance Field Computation

After each cutting operation, the signed distance field is updated in a region growing manner [27,28], starting from those hexahedral elements which are incident to a link that has been cut. The update is performed as follows. When the link between two hexahedral elements has been disconnected by a cut, we assign new vector distances to the respective voxels, and add these voxels into a fifo queue. After all newly disconnected links are processed, we remove a voxel h_i from the list, and check

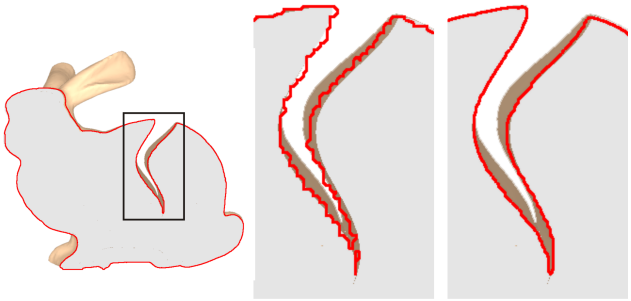

Fig. 5 Comparison between the zero isocontours of the distance fields obtained by the simple interpolation approach (middle) and our topology-aware interpolation approach (right).

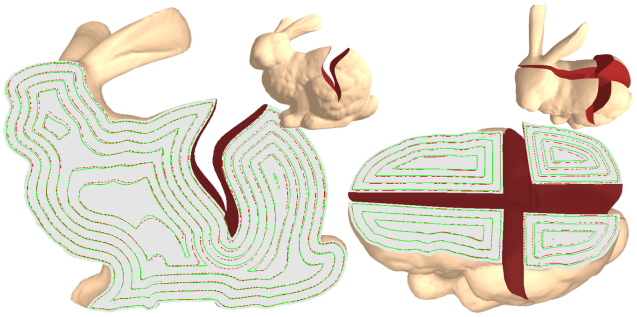

Fig. 6 Isocontours of the distance field computed directly in the deformed configuration (solid green), and of the distance field computed in the reference configuration and transformed into the deformed configuration (dashed red).

its six face-adjacent voxels. The vector distance $d(h_j)$ of a neighbor voxel h_j is updated to $d' = \overrightarrow{h_j h_i} + d(h_i)$, if $\|d'\|_2$ is smaller than $\|d(h_j)\|_2$. If the vector distance of the neighbor voxel has been updated and the neighbor voxel is not already contained in the list, it is added to the list.

Note that to obtain an absolutely accurate distance field, a priority queue would have to be used instead of a fifo queue. This, however, would increase the complexity of updating. For a reasonable collision response, an approximately monotone distance field is sufficient to provide fully reasonable results, as has for example been demonstrated in [19].

Sub-voxel accurate boundary values. Sub-voxel accurate distance values for the boundary voxels improve the accuracy of the distance field over a binary classification [29]. The exact distance value for a boundary voxels can be obtained by determining the distance from the voxel's center to the triangle surface mesh. In our approach, the exact intersection point of a link and the cutting blade as well as the blade's normal is stored when a link is cut. To obtain a sub-voxel accurate distance value for a boundary voxel, we consider the disconnected links that are incident to the corresponding hexahedral element, and compute the minimum distance of the voxel center to the planes spanned by link/blade intersection points and blade normals.

Accuracy of the deformed distance field. Figure 6 shows a comparison between an exact distance field, directly computed in the deformed configuration, and an approximate distance field, computed in the reference configuration and transformed into the deformed configuration. The solid green lines represent the isocontours of the exact distance field, whereas the dashed red lines are the isocontours of the approximate distance field. The approximate distance field decreases monotonically from the interior towards the boundary, and it converges to the exact one as the surface of the object is approached. Recomputing the distance field in the deformed configuration would take minutes, even when accelerated by means of a hierarchical structure [30], whereas locally updating the precomputed distance field in the reference configuration and transforming distance values into the deformed configuration can be performed in a few milliseconds.

6 Results

In the following, we analyze the potential of our proposed collision detection approach for interactive applications like virtual surgery simulations. Our experiments were run on a standard desktop PC equipped with an Intel Xeon X5560 processor running at 2.80 GHz (a single core is used), 8 GB of RAM, and an NVIDIA GeForce GTX 480 graphics card.

Examples. In the first example, we focus on the collision detection between newly created surfaces after cutting. Figure 1 (left) shows a deformable model consisting of hexahedral elements at a resolution of $101 \times 78 \times 100$, and composite elements (yellow grid) that are three levels coarser than the hexahedral elements. In the middle, the bunny, with its bottom fixed to the ground, is cut into two parts by a curved surface, resulting in the upper part sliding down along the cutting surface. On the right, four stamp-like cuts are applied to the bunny. The resulting cylindrical shapes slide out of the bunny's body, under the tight constraints of the holes. Contact forces are computed using a penalty force model, and together with their derivatives are fed into the time-implicit CFE elasticity simulation, which is based on an efficient geometric multigrid solver [3].

The second example, Figure 7 shows haptic cutting on a liver model. The haptic interaction loop is decoupled from the deformation simulation loop; collision detection between the scalpel and the soft liver is performed at 1 kHz haptic rates, while the deformation runs at about 37 simulation frames per second. To give the user an intuitive feedback force, we use a damping force model [31] to compute an elementary cutting force

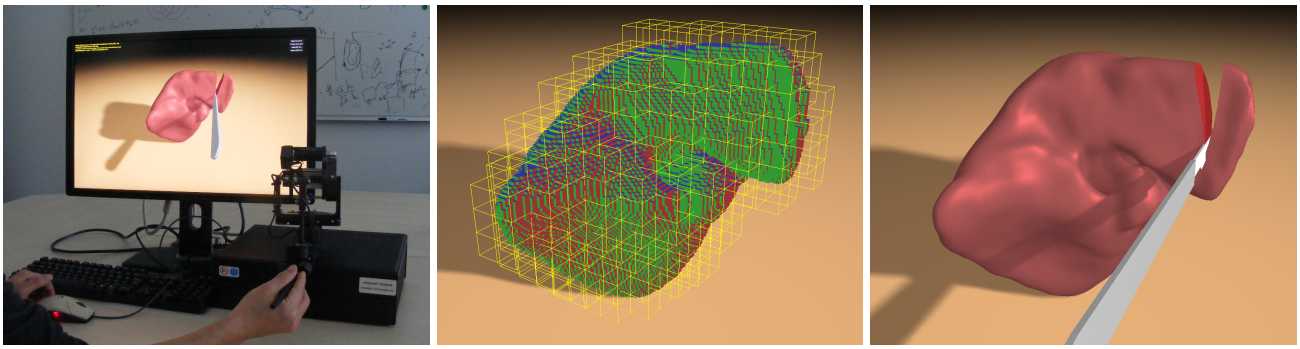

Fig. 7 Haptic cutting of a liver model. Left: Setup. Middle: High resolution hexahedral elements and coarse composite elements. Right: High quality surface mesh used for visualization. The collision detection between the scalpel and the deformable liver model is performed at 1 kHz haptic rates. Please see the accompanying video for dynamic effects.

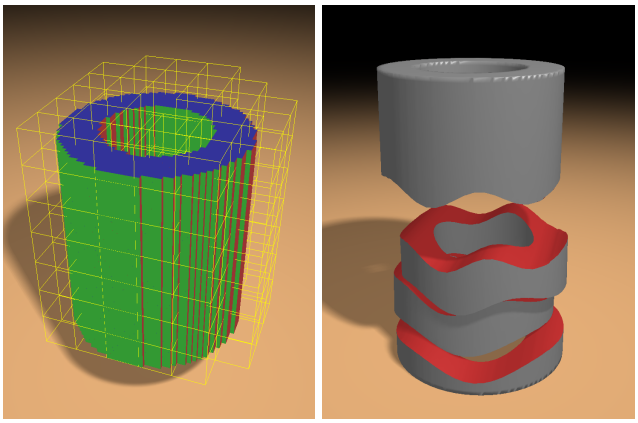

Fig. 8 Cutting of a cylinder shell model. Left: High resolution hexahedral elements and coarse composite elements. Right: After cutting, the parts fall down and stack. Collision detection and finite element simulation are performed at 136 fps, while the per cut updating is finished in 17 ms.

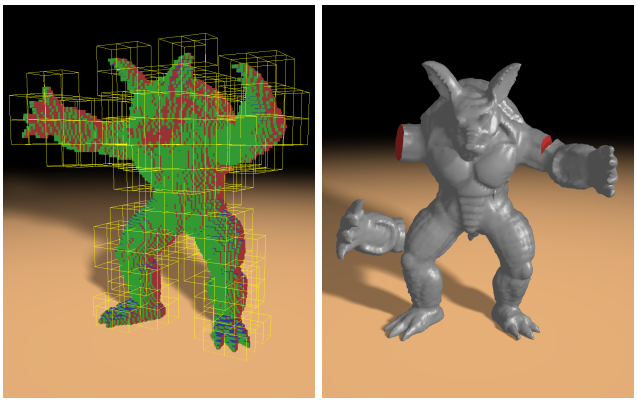

Fig. 9 Cutting of the Stanford Armadillo model. Left: High resolution hexahedral elements and coarse composite elements. Right: After cutting, the parts fall down. Collision detection and finite element simulation are performed at 30 fps, while the per cut updating is finished in 32 ms.

for each sampling segment of the scalpel. The overall cutting force is integrated from all sampling segments. For improved stability, we apply the virtual coupling method (see e.g., [31,32]) to render the feedback force.

Figure 8 demonstrates our collision detection applied to a cylinder shell model, the top of which is fixed. After cutting, the parts fall down, and stack on the ground. Figure 9 shows haptic cutting on the Stanford Armadillo model. For a real-time recording of the dynamic effects, we would like to refer the reader to the accompanying video.

Evaluation of accuracy. To evaluate the sub-voxel accurate collision detection, we analyze the change of the kinematic energy of the simulation objects over time, which is a measure for the smoothness of simulation. Jumps in the kinetic energy can easily lead to visual artifacts and numerical instabilities. Figure 10 compares the change of kinetic energy of a bunny model using the simple distance field interpolation approach with our topology-aware interpolation approach. It can be observed that the change of kinematic energy for the topology-aware interpolation is much smoother than for the simple interpolation. Figure 11 shows the simulation of a cut in a zero-gravity environment. Since the simple interpolation approach leads to an error in penetration depth computation, the cut opens. In contrast, using our topology-aware interpolation approach, the cut remains closed.

Performance. Table 1 shows detailed timing statistics for different models and different resolutions of the hexahedral element grid. The second group of columns gives information about the employed models: the resolution of the hexahedral element grid, the number of simulation DOFs (using a composition level of three), and the number of surface triangles. The third group lists timings for mesh updating (adaptive octree refinements along the cutting surface and creation of the surface mesh), updating of the FEM stiffness matrices, updating of the distance field for collision detection, and finally the total time required for update operations. Note that the time required for updating is dependent on the spatial extent of a cut. In our tests, the cuts were realized such that the number of newly created hexa-

Model	Hexahedral Elements	# Sim. DOFs	# Tris × 1k	Per Cut Updating [ms]				Simulation [ms] (Speedup)		
				Mesh	Stiffness	Distance	Total	Col. Det.	FEM	Total
Bunny	101×78×100	3,669	69	8.9	25.6	2.4	36.9	4.6 (43.5×)	19.6 (253.6×)	24.2 (213.2×)
Bunny	51×39×50	933	17	3.1	5.8	0.4	9.3	1.5 (26.8×)	3.7 (93.2×)	5.2 (67.4×)
Liver	82×83×101	2,928	59	11.7	24.4	1.9	38.0	3.2 (58.4×)	23.3 (103.7×)	26.5 (98.2×)
Liver	41×42×51	717	13	2.8	5.6	0.3	8.7	1.2 (21.3×)	4.3 (67.2×)	5.5 (57.0×)
Cylinder	42×42×50	1,338	30	5.7	11.3	0.2	17.2	2.1 (32.5×)	5.2 (118.4×)	7.3 (93.3×)
Armadillo	85×77×100	2,346	50	9.3	22.6	0.4	32.3	3.7 (26.1×)	29.7 (31.0×)	33.4 (30.5×)

Table 1 Performance statistics for different models and different resolutions of the hexahedral element grid. All models are simulated with a composition level of three.

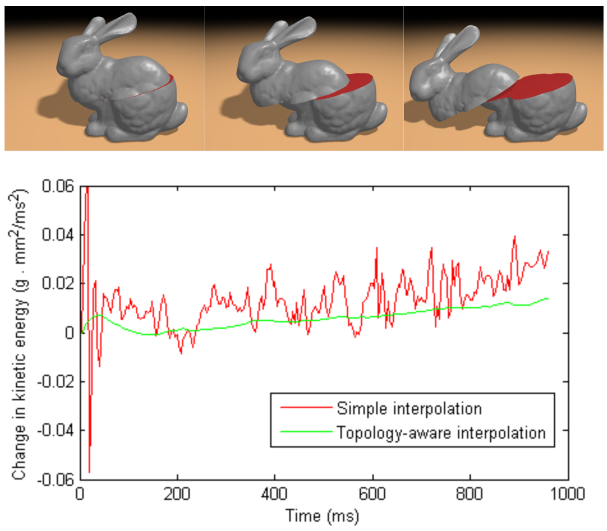

Fig. 10 Analysis of the change of kinetic energy over time. Top: After cutting the bunny, the upper part slides down. Bottom: With our topology-aware interpolation, the change of kinetic energy (green) is significantly smoother than when the simple interpolation is used (red).

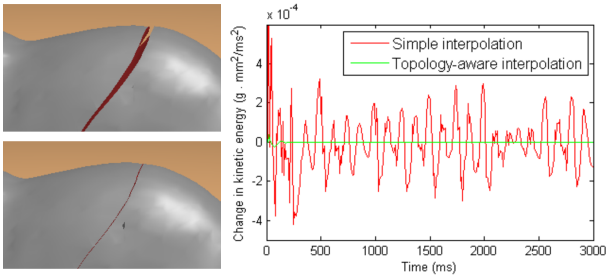

Fig. 11 Cutting in a zero-gravity environment. Left top: Using the simple interpolation, the cut opens since the collision detection is not accurate. Left bottom: With out topology-aware interpolation, the width of the cut remains zero. Right: Change of kinetic energy over time.

hedral elements (due to adaptive octree refinements) in average was about 2% to 4% of the total number of elements. The last group gives timings for collision detecting and FEM simulation. The red numbers in parentheses indicate the performance gain resulting from using composite elements, compared to performing collision detection and finite element simulation on the underlying hexahedral elements. With no composition, the broad phase takes about 86% of the total time for collision detection. As the composition level l increases, the narrow phase is becoming the bottleneck. To increase the overall performance, in the narrow phase we test every 2^lth vertex rather than all vertices. It is worth noting that the resulting speedup factor in the narrow phase (8 when $l = 3$) is far below the overall speedup factor achieved for collision detection (between 21.3 and 58.4).

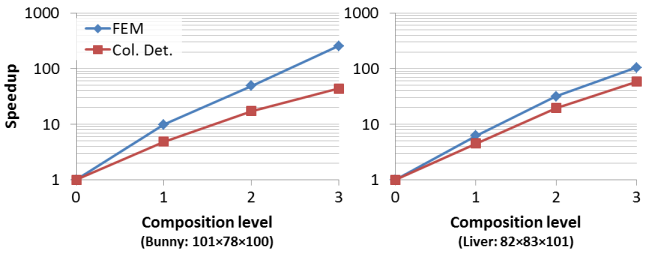

Fig. 12 Speedup with respect to different composition levels.

Our statistics indicate that the composition not only leads to a significantly faster deformation simulation, but also to a highly efficient collision detection. With three levels composition, the speedup of the overall performance can reach about two orders of magnitude. Figure 12 shows the speedups for different composition levels. Exponential growth of the speedup factor can be observed.

7 Conclusion

We have presented an efficient collision detection method for CFE simulation of cuts in deformable bodies. By exploiting the composition structure, we achieve a speedup factor which scales exponentially with respect to the level of composition. By analyzing the topology of underlying hexahedral elements, our approach accurately interpolates the penetration depth for non-conforming geometries. These advances make CFEs an ideal candidate for applications where interactive simulation update rates are required.

In the future, we plan to work on the physically based modeling of the fracture mechanics between a scalpel and soft tissues, in order to provide a fully realistic haptic feedback for surgical training applications. Another direction of research is constraint-based contact handling for deformable bodies. Here, it would be

interesting to investigate the potential of CFEs to accelerate this computationally intensive approach.

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

References

1. W. Hackbusch and S. Sauter, "Composite finite elements for the approximation of pdes on domains with complicated micro-structures," *Numerische Mathematik*, vol. 75, pp. 447–472, 1997.
2. M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, "Preserving topology and elasticity for embedded deformable models," *ACM Trans. Graph.*, vol. 28, pp. 52:1–52:9, July 2009.
3. C. Dick, J. Georgii, and R. Westermann, "A hexahedral multigrid approach for simulating cuts in deformable objects," *IEEE Trans. Vis. & Comput. Graph.*, vol. 17, no. 11, pp. 1663–1675, 2011.
4. L. Jeřábková, G. Bousquet, S. Barbier, F. Faure, and J. Allard, "Volumetric modeling and interactive cutting of deformable bodies," *Progress in Biophysics and Molecular Biology*, vol. 103, no. 2-3, pp. 217 – 224, 2010.
5. J. Wu, C. Dick, and R. Westermann, "Interactive high-resolution boundary surfaces for deformable bodies with changing topology," in *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, pp. 29–38, 2011.
6. S. Sauter and R. Warnke, "Composite finite elements for elliptic boundary value problems with discontinuous coefficients," *Computing*, vol. 77, no. 1, pp. 29–55, 2006.
7. T. Preusser, M. Rumpf, and L. O. Schwen, "Finite element simulation of bone microstructures," in *Proceedings of the 14th Workshop on the Finite Element Method in Biomedical Engineering, Biomechanics and Related Fields*, pp. 52–66, University of Ulm, July 2007.
8. F. Liehr, T. Preusser, M. Rumpf, S. Sauter, and L. O. Schwen, "Composite finite elements for 3d image based computing," *Comput. Vis. Sci.*, vol. 12, no. 4, pp. 171–188, 2009.
9. M. Seiler, D. Steinemann, J. Spillmann, and M. Harders, "Robust interactive cutting based on an adaptive octree simulation mesh," *Vis. Comput.*, vol. 27, no. 6, pp. 519–529, 2011.
10. N. Pietroni, F. Ganovelli, P. Cignoni, and R. Scopigno, "Splitting cubes: a fast and robust technique for virtual cutting," *Vis. Comput.*, vol. 25, no. 3, pp. 227–239, 2009.
11. T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," *ACM Trans. Graph.*, vol. 21, pp. 339–346, 2002.
12. M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Straßer, and P. Volino, "Collision detection for deformable objects," *Comput. Graph. Forum*, vol. 24, no. 1, pp. 61–81, 2005.
13. M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross, "Optimized spatial hashing for collision detection of deformable objects," in *the Vision, Modeling, and Visualization Conference*, pp. 47–54, 2003.
14. K. E. Hoff, III, A. Zaferakis, M. Lin, and D. Manocha, "Fast and simple 2d geometric proximity queries using graphics hardware," in *I3D '01: Symposium on Interactive 3D graphics*, pp. 145–148, ACM, 2001.
15. B. Heidelberger, M. Teschner, and M. Gross, "Detection of collisions and self-collisions using image-space techniques," *Journal of WSCG*, vol. 12, no. 3, pp. 145–152, 2004.
16. F. Faure, S. Barbier, J. Allard, and F. Falipou, "Image-based collision detection and response between arbitrary volume objects," in *SCA '08: Symposium on Computer Animation*, pp. 155–162, EG, 2008.
17. M. Otaduy, O. Chassot, D. Steinemann, and M. Gross, "Balanced hierarchies for collision detection between fracturing objects," in *VR '07: IEEE Virtual Reality Conference*, pp. 83 –90, march 2007.
18. J.-P. Heo, J.-K. Seong, D. Kim, M. A. Otaduy, J.-M. Hong, M. Tang, and S.-E. Yoon, "Fastcd: fracturing-aware stable collision detection," in *SCA '10: Symposium on Computer Animation*, pp. 149–158, 2010.
19. L. Glondu, S. Schvartzman, M. Marchal, G. Dumont, and M. Otaduy, "Efficient collision detection for brittle fracture," in *SCA '12: Symposium on Computer Animation*, pp. 285–294, 2012.
20. D. James and D. Pai, "Bd-tree: output-sensitive collision detection for reduced deformable models," in *ACM Trans. Graph.*, vol. 23, pp. 393–398, ACM, 2004.
21. S. C. Schvartzman, J. Gascón, and M. A. Otaduy, "Bounded normal trees for reduced deformations of triangulated surfaces," in *SCA '09: Symposium on Computer Animation*, pp. 75–82, 2009.
22. J. Barbič and D. James, "Subspace self-collision culling," *ACM Trans. Graph.*, vol. 29, no. 4, p. 81, 2010.
23. G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs, "An implicit finite element method for elastic solids in contact," in *Proc. the Computer Animation*, pp. 136 – 254, 2001.
24. S. Fisher and M. C. Lin, "Deformed distance fields for simulation of non-penetrating flexible bodies," in *Eurographic workshop on Computer animation and simulation*, pp. 99–111, 2001.
25. A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis, "Efficient elasticity for character skinning with contact and collisions," *ACM Trans. Graph.*, vol. 30, pp. 37:1–37:12, July 2011.
26. D. Todd, "b4wind user's guide - trilinear interpolation." www.grc.nasa.gov/WWW/winddocs/utilities/b4wind_guide/trilinear.html. [Online; accessed 30-July-2012].
27. M. Jones, J. Baerentzen, and M. Sramek, "3d distance fields: a survey of techniques and applications," *IEEE Trans. Vis. & Comput. Graph.*, vol. 12, pp. 581 –599, july-aug. 2006.
28. O. Cuisenaire, "Region growing euclidean distance transforms," in *The 9th International Conference on Image Analysis and Processing*, pp. 263–270, 1997.
29. R. Satherley and M. Jones, "Hybrid distance field computation," in *Proc. Volume Graphics*, pp. 195–209, 2001.
30. E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," tech. rep., Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.
31. J. Wu, D. Wang, C. Wang, and Y. Zhang, "Toward stable and realistic haptic interaction for tooth preparation simulation," *Journal of Computing and Information Science in Engineering*, vol. 10, no. 2, p. 021007, 2010.
32. J. Barbič and D. L. James, "Six-dof haptic rendering of contact between geometrically complex reduced deformable models," *IEEE Trans. Haptics*, vol. 1, pp. 39–52, Jan. 2008.