Preserving Topology and Elasticity for Embedded Deformable Models

Matthieu Nesme[1,2,3,4] Paul G. Kry[1] Lenka Jeřábková[3,4] François Faure[2,3,4]

[1] McGill University
School of Computer Science
Centre for Intelligent Machines

[2] Grenoble Universities
[3] INRIA
[4] LJK-CNRS

Abstract

In this paper we introduce a new approach for the embedding of linear elastic deformable models. Our technique results in significant improvements in the efficient physically based simulation of highly detailed objects. First, our embedding takes into account topological details, that is, disconnected parts that fall into the same coarse element are simulated independently. Second, we account for the varying material properties by computing stiffness and interpolation functions for coarse elements which accurately approximate the behaviour of the embedded material. Finally, we also take into account empty space in the coarse embeddings, which provides a better simulation of the boundary. The result is a straightforward approach to simulating complex deformable models with the ease and speed associated with a coarse regular embedding, and with a quality of detail that would only be possible at much finer resolution.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: simulation, animation, embedded deformation, finite element method, heterogeneous materials

1 Introduction

Physical simulation of deformation is an important part of computer animation. One of the biggest obstacles to creating interactive simulations is the complexity of the deformable objects that they contain. Many of the geometric models typically used in graphics have tens of thousands of vertices, if not more. A combination of simplification, approximation, and precomputation is essential to reduce this complexity to the point where interactive physically based deformation is possible. The problem, however, is that many of these techniques do so at the cost of sacrificing important aspects of the object's mechanical behaviour.

Embedding is one popular approximation that simulates a highly detailed geometric model as a low complexity coarse grid of mechanical elements (*i.e.*, a coarse grid simulated using the finite element method). Embedding is a popular approach because of its simplicity, and its ability to preserve fine geometric features. Building nested grids is especially straightforward for hexahedral grids since it is as easy as identifying which vertices fall into which voxels. When any given elastic block undergoes a deformation, the contents are easily deformed using interpolation.

An important problem with the standard embedding technique is that it does not correctly model geometry with complex branching. Consider, for example, the veins in Figure 1. If material falls into

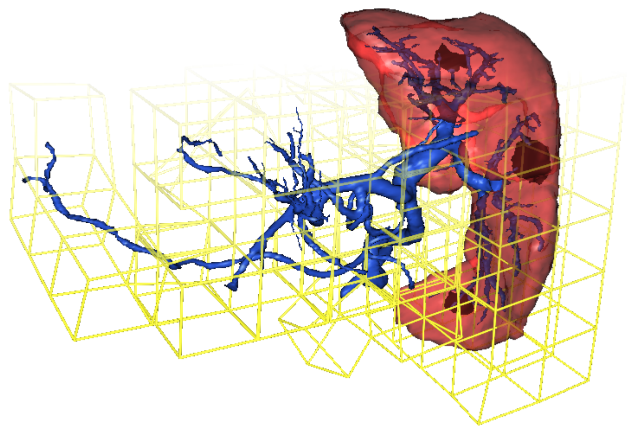


Figure 1: *A model of a liver with attached vascular system simulated with coarse resolution hexahedra. Our technique models the behaviour of the soft liver tissue, stiffer veins, and much stiffer tumors by taking into account a distribution of materials and the presence of empty regions in the embedding. The complex topological branching of the vascular system is preserved by superimposing elements.*

two adjacent voxels, then the embedding naturally includes two elements which are mechanically attached at the four corner vertices of the face they share. But adjacent elements should not always be mechanically connected, for instance, in the case where two veins run through a series of adjacent voxels. More problematic, however, is the case where two veins run through the same set of voxels; the two veins should be mechanically independent, but have the bad luck of falling into the same elements. Finally, there is also the case where a collection of connected fine branches all falls into the same voxel. The element used to simulate this structure should have a lower stiffness than an element which is uniformly full of the same material.

This leads us to another important problem with embedding complex models into coarse finite elements. Complex models can have many different parts with a variety of different material properties. In such cases, it is much more likely that a coarse element will contain a mix of materials, soft and hard, rather than just one material. Therefore, it is difficult to select appropriate material properties for the coarse elements so that they have the correct stiffness. Likewise, there is the problem that all the material inside an element, whether stiff or soft, will deform in the same way. For instance, consider the simple example of a grape with a seed shown in Figure 2. The seed is much stiffer than the rest of the grape, and as such it should keep its shape when the rest of the grape deforms.

In this paper we present a method for addressing the problems described above. Our solution has two main parts. First, we introduce a technique for improving the behaviour of a coarse element containing a mix of materials with different properties. The result, which we call a *composite element*, has the same number of degrees of freedom as a traditional element, but has a stiffness that

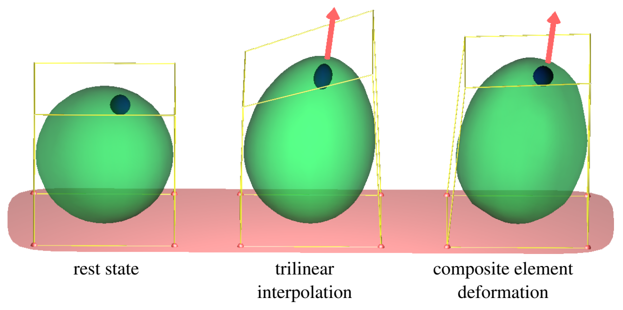


Figure 2: *A soft grape with a stiff seed, deformation with our method shown at right.*

much better represents the behaviour of its contents. Our composite elements use a displacement interpolation that approximates the static behaviour of the underlying geometry. This significantly improves the resulting deformation when elements contain both soft and hard materials. Second, we introduce a technique for building embeddings that correctly model the topology of the underlying geometry. Branching and holes are accommodated through the separation and or superposition of elements. There is no need to build a control skeleton as our method handles these cases automatically. Furthermore, topological branching and void inclusions (*i.e.*, empty regions in the embedding) have an influence on the stiffness, which our method properly takes into account.

The combination of these two main parts form a non-trivial extension of the traditional FEM embedded deformation technique. Our solution provides a substantial improvement in the quality of deformations when using coarse embeddings of complex models with minimal additional run-time cost. As such, we believe our approach has an important impact on how deformation can be simulated in interactive applications.

Our Contributions

The main contributions of this paper are the following. We introduce a *composite element formulation*, which allows the construction of coarse embeddings where the elements correctly capture the linear elastic behaviour, both stiffness and deformation interpolation, of the materials that they contain. A wide range of materials, from very rigid to very soft, can be included in any given element. We account for *void inclusions* in the embedding, which is crucial for simulating thinner geometric parts, and has important implications on the stiffness, deformation interpolation, and topological connectivity of the mechanical model. Our coarse embeddings correctly *preserve the mechanical topology* of the model, which is essential for the simulation of complex systems with branches, holes, or mechanically distant parts which live in close proximity. We present a technique for *embedded surface interaction* that correctly transfers forces from the highly detailed geometry to the degrees of freedom of our composite element mechanical model.

2 Related Work

Considerable work on physically based deformable models has been done since the pioneering work of [Terzopoulos et al. 1987]. We refer the reader to the excellent survey of [Nealen et al. 2005] on this topic. Different strategies have been proposed to reduce the large computation time induced by the large number of degrees of freedom (DOF).

The stability achieved by implicit time integration allows large time steps, which considerably improve simulation speed for stiff models, as shown in [Baraff and Witkin 1998] and widely used since then. More recently, comparable stability has been obtained using fast algorithms such as shape matching [Müller et al. 2005; Rivers and James 2007] or other position-based approaches [Müller et al. 2006], at the price of a lower physical accuracy.

The other, complementary strategy for reducing the computation time is to reduce the number of DOFs in the objects. Because objects typically interact through their surfaces, it is possible to express the physical equations at the surface points only [Bro-Nielsen and Cotin 1996; James and Pai 1999; James and Pai 2003; Pauly et al. 2004]. Such reductions require complex preprocessing and they are limited to small deformations. Multi-resolution geometric models have been used to refine geometry where needed [Grinspun et al. 2002]. This approach is well-suited for an initially smooth geometry which is refined where needed, but it is not a way of simplifying detailed geometry.

The simulation of detailed geometry can be dramatically simplified when the physical unknowns are the DOFs of a deformation field applied to the geometry, rather than the coordinates of the geometry itself. While displacement fields on the surface of quasi-rigid objects can be modeled as textures [Galoppo et al. 2006], the most popular approach is to embed a complex geometry in a volumetric mesh. The flexibility of this approach has been already exploited in commercial products (e.g., Hypermatter Maya plugin). [Faloutsos et al. 1997] have used Free Form Deformation grids [Sederberg and Parry 1986] with limited numbers of deformation modes to animate complex deformable shapes using a small number of DOFs. Dynamic Free Form Deformation (FFD) was later generalized to tetrahedral grids where the cells are finite elements [Capell et al. 2002; Teschner et al. 2004]. Geometry can also be embedded in arbitrary polyhedral elements using harmonic coordinates [Martin et al. 2008]. In [Debunne et al. 2001], surface geometry is attached to an internal multi-resolution tetrahedral mesh. Remeshing has been applied to materials undergoing very large deformations due to plasticity [Wojtan and Turk 2008]. Cube shaped FFD cells have also been used as finite elements [Müller et al. 2004; James et al. 2004], and [Barbič and James 2005] applied modal analysis to such grids in order to automatically perform DOF reduction using the most significant deformation modes. Finite elements [Bathe 1982], possibly using a co-rotational formulation [Müller et al. 2002; Hauth and Strasser 2004] to handle large rotations, are generally used to compute the internal forces due to grid cell deformations. [Botsch et al. 2007] apply glue forces between adaptive rigid cells and achieve the embedding using vertex blending. [Kaufmann et al. 2008] extend this approach to discontinuous deformable cells and compute a smooth, continuous embedding using the *moving least squares* method.

Collision detection and response is more realistic when applied to the embedded objects rather than to the simplified mesh. An important issue is thus to distribute the forces applied to the embedded mesh over the vertices of the coarse mesh. [Sifakis et al. 2007b] derive a formula for points embedded in elastic triangles, and experimentally extend it to non-elastic forces and rigid objects. [Martin et al. 2008] apply it to generalized barycentric coordinates. [Kaufmann et al. 2008] derive a similar formula for elastic forces applied to embedded points.

Most authors have not discussed the physically accurate computation of cell mass and stiffness in the presence of void or non-homogeneous material. The behaviour of coarse elements is improved in [Nesme et al. 2006] by weighting mass and stiffness based on material distribution inside elements. However, physical accuracy can not be obtained using standard shape functions, as discussed in Section 3.1. Impressive results are obtained in [Wojtan and Turk 2008] with thin objects using a method to account for mass distribution inside partially-empty elements. However, they do not discuss element stiffness, and they consider only homogeneous materials.

The handling of non-continuous material within volumetric cells has mostly been considered for cutting objects. Arbitrary cutting of surfaces can be obtained by duplicating the finite elements being cut, rather than remeshing elements, as shown by [Molino et al. 2004]. The elements are the cells of a tetrahedral FFD grid, each of them being initially full. This technique was later improved by [Sifakis et al. 2007a], allowing partially empty cells using an embedded surface mesh. Sophisticated polygon processing is used to dynamically generate cutting surfaces and to analyze the material connectivity between adjacent tetrahedra. [Wicke et al. 2007] cut polyhedra rather than duplicating them, leveraging their method for the simulation of arbitrary convex polyhera. The extended finite element method (XFEM), first proposed by [Belytschko and Black 1999] for the simulation of cracks in structural mechanics and then used for interactive cutting [Jeřábková and Kuhlen 2009], can effectively model discontinuity regions within an FEM mesh by introducing discontinuities in the shape functions.

3 Method Overview and Principles

In this section we first describe the main concepts used by our method in 1D, along with examples that allow the reader to clearly visualize the advantages of our formulation. Following this we will show how these same ideas extend to 2D and 3D in Section 4.

3.1 Computing Shape Functions

Figure 3 illustrates the limitation of homogeneous elements, and how we propose to fix it. We consider the extension of a bar composed of several materials with different stiffnesses (the darker the colour, the stiffer the material). In the real world, when extended due to external forces, this bar deforms non-uniformly, as shown in Figure 3b, and this is what one expects from a simulation.

In finite elements, the displacements are interpolated inside the elements using a weighted sum of the function values at the nodes: $u(x) = \sum_i \phi_i(x) u_i$, where the ϕ_i are called shape functions. If we model the bar using a classical finite element, we use arbitrary shape functions such as the linear ones depicted in blue in Figure 3d. Consequently, all the points in the bar undergo the same extension, as shown in Figure 3c. This is not visually realistic, because softer parts should extend more than stiffer ones, as seen in Figure 3b. The ideal shape functions for this non-homogeneous element are the red ones in Figure 3d, which correspond to the real behaviour of the bar. Moreover, using the wrong shape functions leads to an incorrect stiffness matrix, resulting in the wrong bar extension shown in Figure 3c. The stiffness matrix of this one-dimensional bar, derived from standard elasticity [Bathe 1982], is

$$K = \int_0^{l_0} \begin{bmatrix} \frac{d\phi_A}{dx} & \frac{d\phi_D}{dx} \end{bmatrix}^T k \begin{bmatrix} \frac{d\phi_A}{dx} & \frac{d\phi_D}{dx} \end{bmatrix} dx, \tag{1}$$

where k is the material property (*i.e.*, local stiffness). With the linear shape functions, the slopes $\frac{d\phi}{dx}$ are too high inside the stiff part, resulting in an exaggerated overall stiffness; overall, the whole bar is mechanically too stiff, while the the middle part will have the appearance of being softer than it really is. Using higher-order shape functions would not fix the problem because they are designed independently of the material distribution within the element. In summary, heterogeneous materials in classical finite elements create both visual and mechanical artifacts.

Our approach to alleviate this problem is based on high-resolution mechanical analysis, performed as a precomputation step for each element. Consider the example in Figure 3 again, and suppose that we know the stiffness matrices of each subpart of the bar (we can see these as fine level elements). Assembling these matrices into one matrix K_a allows us to write the static equilibrium equation [Bathe 1982] relating the fine level nodal displacements (including the endpoints) in vector u, to the fine level nodal forces in vector f. That is,

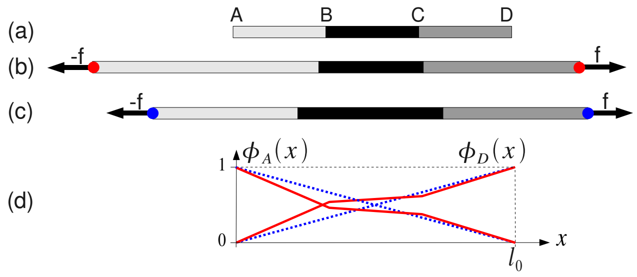


Figure 3: *Homogeneous and composite elements for a bar made of three parts with different stiffnesses. (a) Rest state. (b) Extended by an external force, simulated as one composite element (red nodes). (c) Extended by the same force, simulated as one homogeneous element (blue nodes). (d) Shape functions in the homogeneous element (dotted blue) and in our composite element (red).*

$$K_a u = f. \tag{2}$$

We can impose either a given force or a given displacement at each node, rearrange the equation system accordingly, and solve for the unknown at each node. Imposing the displacements of the endpoints $u_A = 0$, $u_D = 1$, and zero forces at the internal points, we can compute the associated f_A, f_D, and the displacements at the internal points. Point A being fixed and point D undergoing a unit displacement, the displacements at the internal points correspond to the value of the shape function ϕ_D at each of these points. Similarly, we can impose $u_A = 1$, $u_D = 0$, and compute the value of ϕ_A at each internal point.

In practice, we instead solve this in a way which is more easily generalizable to 2D and 3D, using constraints. We find the unknown embedded nodal displacements in the constrained static equilibrium

$$K_a u - G^T \lambda = 0, \tag{3}$$

where the coarse nodal displacements are fixed by constraints, G is the constraint gradient, and λ is a vector of Lagrange multipliers. We solve for the displacements of embedded nodes, then build a matrix H containing the displacement interpolation functions ϕ discretized at the nodal points (see Appendix A for more details). This matrix is a kinematic relation mapping displacements of the coarse system to those of the assembled system,

$$u = H u_g, \tag{4}$$

where vector u_g is the displacement of coarse nodes (the endpoints A and D in our example).

Using these discretized shape functions, we can update embedded points according to the control points, taking into account the internal properties of the element, and thereby providing more realism for heterogeneous elements than previous methods. We will now show how to distribute forces applied at embedded points to the coarse control points, based on our discretized shape functions.

3.2 Interaction on Embedded Points

Controlling embedded points is useful, especially for contact forces in 2D and 3D. Hard constraints can be computed through the pseudoinverse H^+ by adapting the direct manipulation of a FFD [Hsu et al. 1992], but it requires solving a system of equations. Moreover, when a detailed mesh is embedded, a large number of contact points can lead to singular equations. For these reasons, penalty forces are better adapted to embedded physical models. The forces applied to the embedded points must be converted to the equivalent forces applied to the control points. This can be done with a projection, which can be explained as follows. Let vector $\dot{u}_x$ represent the velocities of an arbitrary number of embedded points, and vector $\dot{u}_g$ represent the velocities of their control points. Let matrix H_x denote the linear relation between these velocities,

$$\dot{u}_x = H_x \dot{u}_g. \tag{5}$$

In finite elements, matrix H_x is also used for displacements, and simply consists of the shape functions ϕ evaluated at x. Given a vector of forces f_x applied to the embedded points, we want to compute the equivalent vector of forces f_g. To be equivalent, these forces must have the same power, that is, $\dot{u}_g^T f_g = \dot{u}_x^T f_x$. Using equation 5, we obtain $\dot{u}_g^T f_g = \dot{u}_g^T H_x^T f_x$. Since the latter is true for any given velocity $\dot{u}_g$, the equivalent force is necessarily

$$f_g = H_x^T f_x. \tag{6}$$

This relation holds for all embeddings in any dimension, and generalizes the hard bindings of [Sifakis et al. 2007b]. We use this relation to compute the stiffness of coarse elements in the following section.

3.3 Stiffness, Mass, and Damping

Using the shape functions computed in Section 3.1, we now have a more general way to find the stiffness of the bar in Figure 3 modeled as a single element AD while *taking into account its heterogeneous composition*. Equations 2, 4, and 6 let us project the assembled stiffness of the high-resolution system to form the stiffness of the coarse element. Specifically,

$$K = H^T K_a H. \tag{7}$$

This can also be seen as equivalent to the integral in Equation 1 through conversion to a sum and application of a discrete differentiation operator to H.

Accurate evaluation of the coarse mass and damping matrices are also done via the same projection. Note that the static deformation solution of Equation 3 used to compute the shape functions additionally provided us with the coarse element stiffness K. While this is strictly equivalent in 1D, it is not the desired result in 2D and 3D, as explained in Section 4.

3.4 Composite Elements at Object Boundaries

When embedding meshes, elements are not necessarily full of material and can contain void parts around the object surface. Clearly this void must be taken into account in the mechanical behaviour of boundary elements, yet to our knowledge no previous work has addressed this problem. Consider the example in Figure 4, in which part BC is void. Directly applying the approach presented in Section 3.1 does not work, because the stiffness K_{BC} is null and the equation systems used to compute the shape functions are singular. Indeed, forces applied to point C can not be propagated to the rest of the bar through a part with zero stiffness. This problem is easily solved by removing point C and considering only element AB. Note that this can be interpreted as using a zero compliance (*i.e.*, rigid) part BC instead of a zero stiffness void. This is illustrated in Figure 4b, which corresponds to the shape functions depicted in Figure 4c. The zero compliance element is handled in the static analysis through the addition of a constraint, in this case, $u_B = u_C$.

3.5 Preserving Topology

We handle disconnected material within one element by duplicating the element, each of the resulting elements carrying one connected component, as illustrated in Figure 5. We then compute the mass

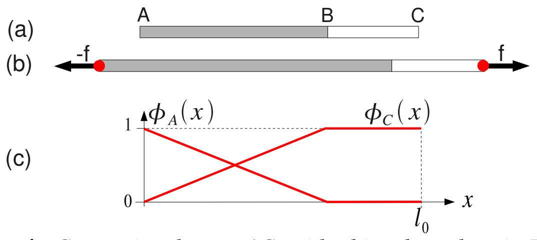


Figure 4: *Composite element AC, with object boundary in B. (a) Rest state. (b) Extended by an external force. (c) Shape functions.*

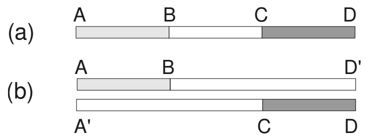


Figure 5: *Elements with disconnected material (a) are split in distinct, initially superimposed elements (b).*

and stiffness matrices according to the actual amount of material in the elements, as explained in Section 3.3. This contrasts with [Molino et al. 2004] who do not discuss the stiffness issue, and give both elements the mass of the original in order to avoid stability problems with small masses. We have not encountered such problems, probably because our computations make mass and stiffness consistent with each other, and we do not artificially increase the total mass.

4 Composite Elements in 3D

This section presents a practical means of implementing the theory explained in Section 3 for three dimensional geometry. The issues that arise in 3D are also present in 2D; we will describe parts of our technique for the 2D case when it is best for the sake of simplicity. In general, however, we are considering the case of a 3D object with high geometric detail and varying elasticity properties embedded into a hexahedral grid (though these concepts also apply to elements of other shapes). Since each step in this process is slightly more complex in 3D, we revisit each of the main concepts. First, we describe how we compute the displacement interpolation H, and in turn the stiffness matrix of the coarse element. Next, we address the situation where there are voids present among the fine elements that we are combining to produce the coarse composite element. Finally, we will discuss how to accommodate topological branching and its implications on the stiffness and interpolation.

As input data, a fine voxelization representing the object with material properties for each voxel is needed in order to compute the fine static analysis. Such a representation can be obtained from volumetric images or from surface meshes. In our implementation we use the voxelization described in [Nesme et al. 2006]. The idea is to decompose the coarse mechanical grid following an octree scheme, where the number of levels is given by the user. Mechanical properties can be defined for each independent part of the initial polygonal mesh, and can so be assigned at the finest voxelization. Additionally, we consider boundary voxels as half-full with a corresponding adjustment to the stiffness and mass (though this is just one of many possible choices). An overall ordering resolves the case where multiple polygonal meshes with different materials enclose the same fine voxels.

When the object is represented by a polygonal mesh, a surface vertex s is trilinearly interpolated with the finest voxelization (using a matrix B_s). Since the fine node displacements are interpolated from coarse node displacements with the matrix H, the displacement at a surface vertex u_s has the form $u_s = B_s u = B_s H u_g$.

4.1 Interpolation and Stiffness

Our composite elements use the co-rotational approach, except that each element's constant stiffness (along with its displacement interpolation) provides a far better approximation of the behaviour of the underlying geometry.

Recall that the correct stiffness and displacement interpolation of a coarse composite element was easy to find in 1D. This is because the effective overall stiffness of the coarse element, along with the displacement interpolations for the fine nodes, could be found from a static deformation analysis. An important difference in 1D is that the static solution was naturally constrained to have the fine DOFs remain on the line between the two coarse level end points. In 2D and 3D, we still want to use a static deformation analysis of the element to find the displacement interpolations, but fine DOFs along an edge may not stay along that edge (for instance, if the Poisson ratio is not zero). This is illustrated in Figure 6 (a) where the black arrow and disks represent constraints applied to the nodes of the coarser element, red arrows show fine displacements, and the thin grey lines show a deformed fine model. In this example, the upper right fine element is stiffer than the others, and the overall shape of the element does not match the desired shape of the deformed coarse element, shown in blue in Figure 6 (b).

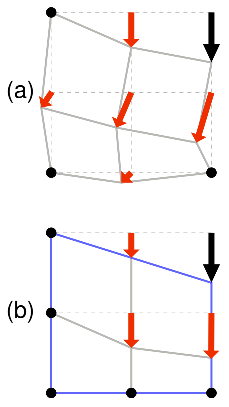


Figure 6: *Constraints in a 2D static solution.*

We intend to assemble many composite elements into a larger system, allowing large global deformation with our improved stiffness matrices, but we do not want adjacent elements to interpenetrate or separate during deformation. Additionally, a coarse node displacement should not influence embedded nodes on the far boundary. Finally, the stiffness of our coarse element is known in the homogeneous case, and we want our computation of the interpolation operator, H, to lead to this correct answer when the fine-resolution assembled homogeneous stiffness is projected using Equation 7.

Our solution to ensure a correct coarse stiffness in the homogeneous case requires H to interpolate displacements in each axis independently. This is exactly like the standard trilinear interpolation, except that in the non-homogeneous case we allow different weights for each axis direction. That is, the influence of a coarse node on an interior node can be a non-uniform scaling of the coarse displacement. The easiest way to do this is to solve for the static equilibrium for each axis of each coarse node, where Equation 3 has additional constraints added to force embedded nodes to respect the coarse boundary with movement restricted to the given axis (see Appendix A for more details). Figure 6 (b) shows an example of one such static solution. Once we have the response for each coarse node axis aligned displacement (the columns of H), we follow this by a renormalization to ensure that all rows of H sum to one. With this final modification, projecting the fine-resolution assembled stiffness in the homogeneous case gives *exactly* the expected result.

Since H scales displacements non-uniformly in each axis, we effectively have a piecewise linear interpolation function in each axis for each coarse node. Because the displacements are weighted in a locally rotated frame, they can differ slightly across coarse element boundaries. We resolve this by blending the embedded points at the boundary of coarse elements, which results in a small additional run-time cost in comparison to a traditional embedding.

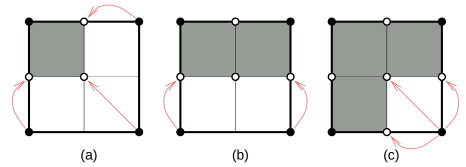


Figure 7: *Kinematic relation between disconnected coarse nodes (black) and fine nodes (white) for all cases in 2D. The red arrows represent a rigid connection.*

4.2 Void elements

The nodes of a coarse grid do not necessarily lie within the embedded object. In the 1D case it is easy to define how to relate such coarse nodes with the fine voxelization (see Section 3.4). Unfortunately, in higher dimensions the problem becomes more complicated because there is no easy solution for kinematically linking these disconnected coarse nodes to the fine nodes in the general case.

Instead of solving a static analysis to bring the properties of the fine voxelization directly to the coarse grid, we decompose the problem into a sequence of easier sub-problems. Specifically, we create a single composite element from a set of eight fine elements in 3D (four in 2D), and repeat the process using an octree (quadtree in 2D). Using recursion we create the final coarse approximation of the fine scale mechanical behaviour. This recursive calculation of stiffness produces results identical to a direct analysis when there are no empty elements.

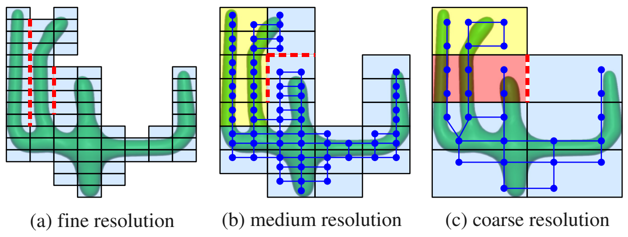
For each sub-problem, the choice of the kinematic relation linking disconnected coarse nodes to non-empty fine elements is straightforward. In 2D, there are only three possible cases, as presented in Figure 7 (note that the fourth case, with two diagonal empty elements, is handled by topological branching). The red arrows correspond to a rigid connection. Similar to the method discussed in 1D, this is handled by adding constraints on the fine nodes during the static analysis. For case (a), the resulting composite element has exactly the same stiffness matrix as the fine element. For case (b), the resulting composite element has the same stiffness matrix as if we were to fit a single element containing only the two non-empty elements. For case (c), the resulting composite element has a more complicated stiffness matrix. While the two inner arms of the material will not be able to bend independently at the coarse level (there are not enough DOFs), the overall behaviour is preserved.

In 3D, the same principle applies; linking constraints can be applied automatically. It results in an H where the disconnected nodal displacements are connected to their closest embedded nodes with constraints.

4.3 Topological Branching

When disconnected parts of a model fall into the same element we create separate yet superimposed embeddings for these different parts. This can be seen as preserving the topology of the model as mechanical loops are not formed when topologically distant parts of the embedded surface fall in the same voxel.

The connectivity of the mechanical mesh is determined at a fine level embedding. The octree subdivision (quadtree in 2D) is used as described in Section 4.2. Before mechanically connecting adjacent cells at this finest level, we check that there is material crossing the given boundary. For cells that are only partially filled with material, this is done by inspecting the intersections between the rectangular cell boundary and the triangles of the embedded fine geometry. Once we have the correct mechanical topology at the fine level, we can build a coarse representation while preserving the connectivity from the fine level.



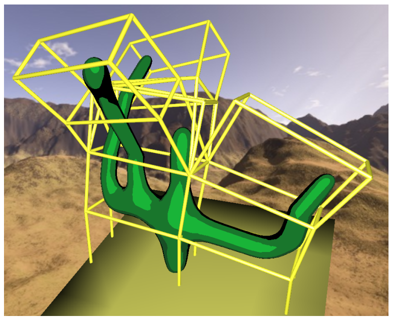


Figure 8: *Resulting branching for a cactus. Top: blue, yellow, and red shading denote one, two, and three superimposed elements respectively. (b, c): the connectivity graph from the previous level is used to detect independent material components within each coarse cell. Bottom: an example of 3D deformation.*

Figure 8 shows an example of the different cases that arise. At the fine level (a), it is assumed that each cell only contains one continuous piece of material; note that this particular implementation has the limitation of not separating independent components contained in the same fine element. The dotted line between the first and second and the second and third cell column marks cell borders which are not mechanically connected. Based on this, a connectivity graph is created, with nodes representing the finite elements of the fine level connected by edges when linking material is present. For each cell of the coarse level we determine the number of independent material components using the connectivity graph from the previous finer level (b, c). One finite element is created for each independent, connected graph component within a cell. The number of superimposed elements per cell is not limited. While all of the superimposed finite elements at a given level will have the same extent as the cell they occupy, they will be only partially filled with material as explained in Section 4.2. Each element is connected to its adjacent elements according to the corresponding graph edges crossing the cell borders. The bottom image of Figure 8 shows an example deformation which reveals the resulting branching structure after two coarsening steps.

Effects on the Static Analysis

Because the topological branching can superimpose elements and nodes, the number of fine nodes and fine elements that we need to consider in our static analysis is not fixed. For example, the C-shaped object in Figure 9 has a doubled node in its intermediate level system. Handling this topology is essential to obtain a good behaviour at the coarse level. That is, we want to ensure that separated branches within the same element are interpolated independently and therefore, in this case, maintain their thickness when deformed. Note that more complex situations can arise when elements are superimposed, resulting in a static analysis involving more than eight elements (four elements in 2D). It suffices to link each coarse node to its corresponding fine nodes in order to interact with all superimposed elements.

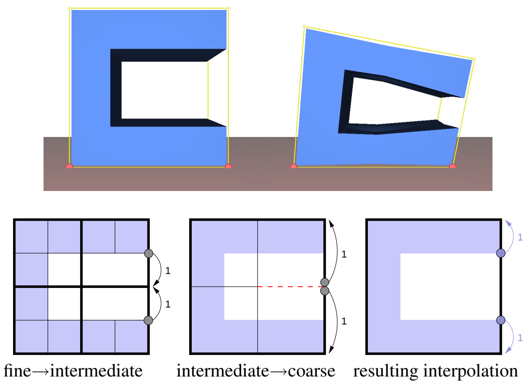


Figure 9: *An example of a composite element created with a void interior. The top row shows the rest and deformed pose with a single composite element. The bottom row shows how fine nodes end up moving rigidly with coarser nodes, while topologically separate parts are merged into one coarse composite element.*

5 Results

We demonstrate our method on a variety of models with a wide range of materials and different topologies. First, we discuss some simple examples that demonstrate that our composite elements produce the expected behaviour. Then, we show more complex examples with detailed models.

Although any time integration scheme can be used, all our examples use implicit Euler integration. To handle collisions and self-collisions, the fast GPU-based method presented in [Faure et al. 2008] is used. Precomputation times are in the range of a few seconds to a few minutes, while the interactive frame rates we reported for our simulations were recorded on a computer with a 2.5 GHz dual-core CPU, GeForce 9800 GX2, and 3 GB of RAM. Note that our implementation will be freely available for download at http://www.sofa-framework.org/ in an upcoming release of SOFA.

5.1 Validations

Figure 10 shows a number of important test cases, comparing results of fine level simulation, a single homogeneous element, and a single composite element. Example (a) shows a homogeneous soft cube under gravity. The estimated interpolation H results in a composite element with the exact same stiffness matrix as a coarse uniform element (as expected). Example (b) shows that a cube with a soft middle layer is well approximated with a single composite element, as the deformation behaviour is similar to the high resolution simulation. In contrast, the uniform element provides a poor approximation in the compliant direction. Examples (c) and (d) show a half void element where the composite element matches the behaviour of the coarse uniform element that exactly fits the object. Note that if the uniform element was the size of the composite element (not shown), it would be too stiff. Examples (e) and (f) show a block embedded in an element which results in a composite element which is mostly empty, yet the expected behaviour still matches that of the coarse uniform element which conforms to the shape of the block.

Figure 11 demonstrates a validation using cylindric beams deformed under gravity while being fixed on one side. All simulated cylinders are modeled with the same material properties. On the left, in transparent pink, and superimposed with all other examples is scanned real data [Marchal et al. 2008]. Next to this is a simulation with classical fitted finite elements [Bathe 1982]. In the center is our composite method, which matches closely with the scanned data. The last two examples correspond to fill-weighted uniform

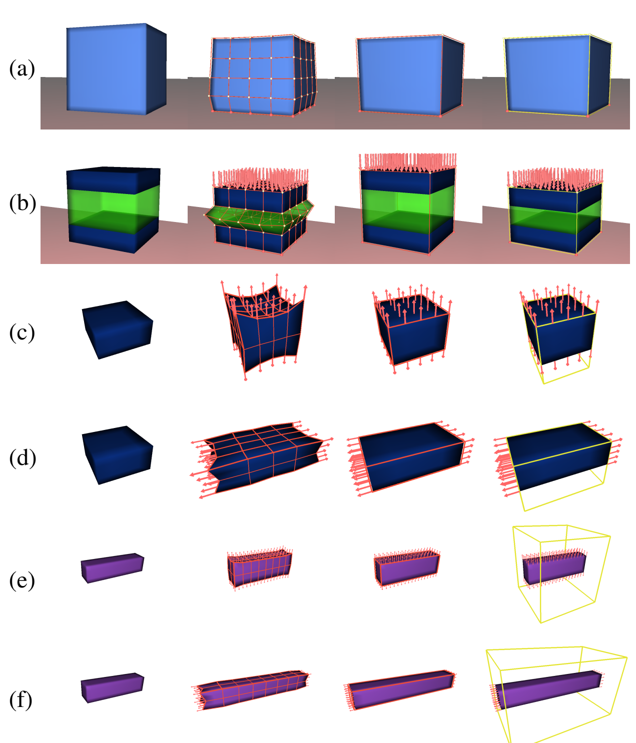


Figure 10: *Validation of composite element behaviour. From left to right, the rest shape, simulation at high resolution, simulation with one uniform element, and simulation with one composite element. Arrows denote forces, red nodes are fixed, red lines denote classical uniform elements, yellow lines denote composite elements.*

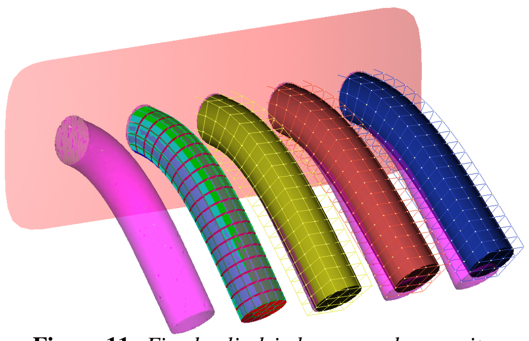


Figure 11: *Fixed cylindric beams under gravity.*

hexahedra [Nesme et al. 2006], and full uniform hexahedra [Müller et al. 2004; James et al. 2004] respectively. Both result in a noticeably stiffer behaviour than desired.

5.2 Complex Systems

In this section we illustrate the behaviour of our method for complex models as can be found in real world applications. In the first two examples, the simulated objects consist of different materials. In Figure 12, two balls, one very soft and the other very stiff, are surrounded by soft material. If this composite object is pressed between two planes, the total deformation in the compression direction is the same in the vicinity of both the soft and stiff balls. However, due to the nonuniform displacement interpolation described in Section 4.1, the soft ball is compressed much more than the stiff ball and more than the surrounding matter, whereas the stiff ball remains nearly in its rest shape. In a second experiment, the same composite object is deformed under gravity while its bottom side is supported by a plane. The surrounding soft material above and below the two spheres undergoes the same amount of deformation, whereas the soft sphere is deformed much more than the stiff sphere. Therefore, in the resulting deformed shape we can observe a depression above the soft sphere and a bump above the stiff sphere.

A second example, in Figure 13, shows a medical simulation based on real patient data. A nonhomogeneous liver consisting of soft tissue with stiffer blood vessels is embedded into a coarse simulation grid. The mechanical properties at a fine discretization level were derived directly from available medical data. Moreover, three nearly rigid tumors are present. During an interactive manipulation with a tool, the veins are deformed less than the surrounding liver tissue, whereas the tumors remain almost undeformed.

The next two examples illustrate the handling of voids and branched topologies. Figure 14 shows an extreme example of handling topological branching on a very coarse grid. The eight tentacles of an octopus are embedded in a very coarse grid of $(2 \times 7 \times 2)$ cells using superimposed finite elements as described in Section 4.3. Despite the coarse grid resolution, very reasonable results are obtained.

Our last example in Figure 15 demonstrates the handling of void parts and thereby a way to simulate thin geometry details using coarse elements. Indeed, thin body parts of the mouse, such as the legs, arms, neck and tail stay stiff in the longitudinal direction but are softer for bending than when simulated using classical elements, which are not able to take voids into account.

5.3 Limitations

Of course, composite elements suffer from some of the same problems as co-rotational formulations. Specifically, the constant stiffness matrix requires the local deformation to remain relatively small. Likewise, a well known problem of finite elements is that they bend less when the embedding is too coarse. Nonetheless, our examples demonstrate a number of complex models undergoing large deformations with very coarse grids of composite elements.

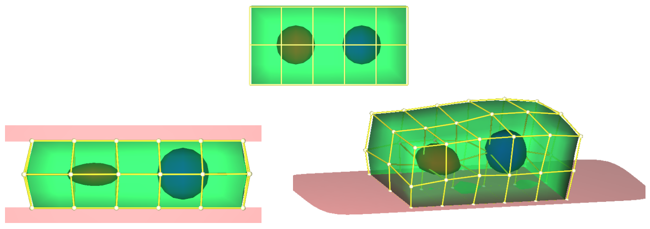


Figure 12: *A very soft (red) and a very stiff (blue) beads are included into soft material (green). Top: the rest shape, bottom left: compressed by two planes, bottom right: deformed under gravity.*

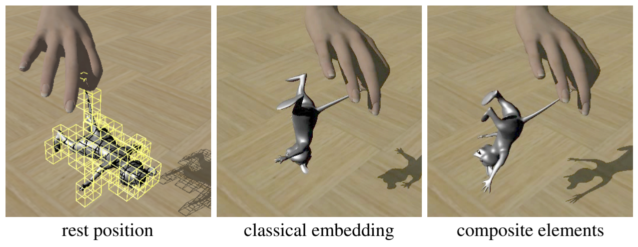


Figure 15: *A mouse is shaken by the tail* $(10 \times 10 \times 10$ *mechanical grid). The snapshots in the middle and on the right are taken at the same instant of the same animation. Classical elements result in overly stiff behaviour because voids are not considered, while composite elements give more natural deformations.*

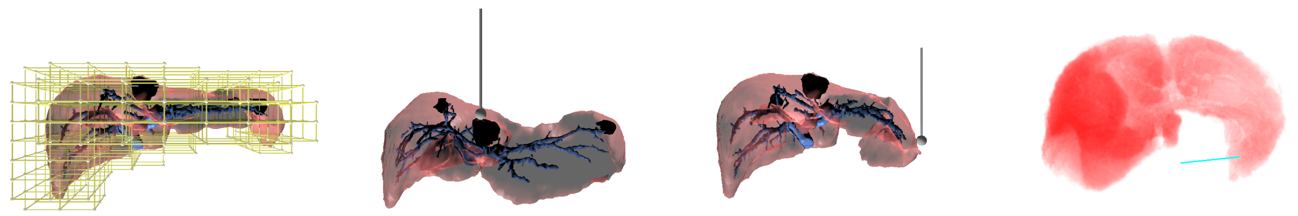


Figure 13: *A liver represented by a very soft tissue (red transparent), stiffer blood vessels (blue) and three very stiff tumors (black). Even with a coarse mechanical mesh (left picture in yellow), the different material properties can be simulated. When a tool deforms the organ, tumors are not deformed (10 fps including collision with tool). On the right, the liver simulation rendered with the volumetric medical data that was used to define the fine level stiffness properties (110 fps).*

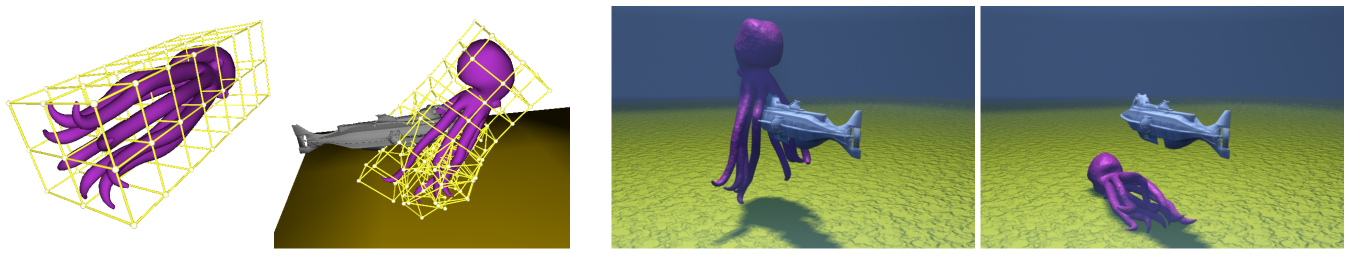


Figure 14: *An octopus, embedded in an extremely coarse grid ($2 \times 7 \times 2$), collides with a submarine (20-30 fps with self-collisions). The topology is preserved by superimposing tentacle elements. At right, the same animation with an off-line rendering.*

6 Conclusion

We have presented a method for simulating highly detailed geometric models with heterogeneous material properties using very coarse grids. Our method overcomes important problems with the traditional embedding technique by taking into account the topology and material properties of the underlying geometry. As a result, deformations with our composite element embedding are much more faithful to the embedded object.

Voids are handled gracefully; they are fully rigid with respect to interpolation, yet fully compliant with respect to stiffness. This novel treatment of the empty space improves the representation of mass and stiffness at the coarse level. Mass is not inflated, and when a composite element has smaller mass, it also has smaller stiffness. As such, the condition of our system does not degrade as quickly in these cases. The treatment of voids also makes possible the preservation of fine mechanical topology, on very coarse regular grids, through the superposition of composite elements.

Finally, a current limitation with our approach is that we do not handle changes in topology. While we do handle the topology of the underlying model, we do not have a means of modifying it on the fly. We believe that this would be an interesting extension to this work.

Acknowledgements

This work was supported in part by grants from NSERC, FQRNT, and CORDIS project PASSPORT (223894). Special thanks to Michaël Adam and Florent Falipou for the mouse example, Adrien Bernhardt for the cactus model, Midori Hyndman for the proofreading, and all the SOFA team.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998*, ACM, 43–54.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St.Venant-Kirchhoff deformable models. *ACM Transactions on Graphics 24*, 3, 982–990.

BATHE, K. 1982. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall.

BELYTSCHKO, T., AND BLACK, T. 1999. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering 45*, 5, 601–620.

BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. 2007. Adaptive space deformations based on rigid cells. *Computer Graphics Forum 26*, 3, 339–347.

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

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics 21*, 3, 586–593.

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

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3, 201–214.

FAURE, F., BARBIER, S., ALLARD, J., AND FALIPOU, F. 2008. Image-based collision detection and response between arbitrary volumetric objects. In *ACM Siggraph/Eurographics Symposium on Computer Animation*.

GALOPPO, N., OTADUY, M. A., MECKLENBURG, P., GROSS, M., AND LIN, M. C. 2006. Fast simulation of deformable models in contact using dynamic deformation textures. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 73–82.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Transactions* *on Graphics 21*, 3, 281–290.

HAUTH, M., AND STRASSER, W. 2004. Corotational simulation of deformable solids. In *J. Winter School of Computer Graphics*, vol. 12, 137–145.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, ACM Press, New York, NY, USA, 177–184.

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

JAMES, D. L., AND PAI, D. K. 2003. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics 22*, 1, 47–82.

JAMES, D. L., BARBIČ, J., AND TWIGG, C. D. 2004. Squashing cubes: Automating deformable model construction for graphics. In *ACM SIGGRAPH Conference on Sketches & Applications*.

JEŘÁBKOVÁ, L., AND KUHLEN, T. 2009. Stable cutting of deformable objects in virtual environments using XFEM. In *IEEE Computer Graphics and Applications*, vol. 29, 61–71.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2008. Flexible simulation of deformable models using discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 105–115.

MARCHAL, M., ALLARD, J., DURIEZ, C., AND COTIN, S. 2008. Towards a framework for assessing deformable models in medical simulation. In *International Symposium on Biomedical Simulation*, 176–184.

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

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

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

MÜLLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *IEEE Computer Graphics International*, 26–33.

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

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2006. Position based dynamics. In *Eurographics Virtual Reality Interactions and Physical Simulations*, 71–80.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Computer Graphics Forum*, vol. 25 (4), 809–836.

NESME, M., PAYAN, Y., AND FAURE, F. 2006. Animating shapes at arbitrary resolution with non-uniform stiffness. In *Eurographics Virtual Reality Interactions and Physical Simulations*.

PAULY, M., PAI, D. K., AND GUIBAS, L. J. 2004. Quasi-rigid objects in contact. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 109–119.

RIVERS, A. R., AND JAMES, D. L. 2007. FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics 26*, 3, 82:1–82:6.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, ACM, 151–160.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, 73–80.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 81–90.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, ACM, 205–214.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., AND GROSS, M. 2004. A versatile and robust model for geometrically complex deformable solids. In *Computer Graphics International*, 312–319.

WICKE, M., BOTSCH, M., AND GROSS, M. 2007. A finite element method on convex polyhedra. *Computer Graphics Forum 26*, 3, 355–364.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics 27*, 3, 47:1–47:8.

A Constrained Computation of H

Interpolation weights in the matrix H come from the static equilibrium in Equation 3, $K_a\,u - G^T\lambda = 0$, where G is the constraint gradient, and λ is a vector of Lagrange multipliers. Because the displacements u consist of both the unknown u_e and the imposed u_g, we split $K_a\,u$ and bring all known quantities to the right, giving

$$K_e\,u_e - G^T\lambda = -K_g\,u_g. \tag{8}$$

Solving gives desired interpolated displacements u_e, and constraint forces; since u_g is arbitrary, the displacement interpolation functions ϕ for the embedded points, i.e., H_e, is the top portion of

$$-\begin{bmatrix} K_e & -G^T \end{bmatrix}^{-1} K_g, \tag{9}$$

while the bottom portion computes the constraint forces. Adding rows to H_e that map coarse displacements to themselves gives the kinematic relation between the coarse and assembled system displacements, $u = H\,u_g$, seen in Equation 4.

In 2D and 3D, simple velocity constraints on the embedded points ensure that the coarse boundary is respected. We add these constraints to Equation 3, and bring the newly constrained coordinates u_p to the right hand side by setting their position with a trilinear interpolation H_p of the coarse nodes ($u_p = H_p u_g$). Thus, the unknown portions of H come from the top part of

$$-\begin{bmatrix} K_e & -G_p^T & -G^T \end{bmatrix}^{-1} (K_g + K_p H_p). \tag{10}$$