EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

A Finite Element Method on Convex Polyhedra

Martin Wicke Mario Botsch Markus Gross

ETH Zurich

Abstract

We present a method for animating deformable objects using a novel finite element discretization on convex polyhedra. Our finite element approach draws upon recently introduced 3D mean value coordinates to define smooth interpolants within the elements. The mathematical properties of our basis functions guarantee convergence. Our method is a natural extension to linear interpolants on tetrahedra: for tetrahedral elements, the methods are identical. For fast and robust computations, we use an elasticity model based on Cauchy strain and stiffness warping. This more flexible discretization is particularly useful for simulations that involve topological changes, such as cutting or fracture. Since splitting convex elements along a plane produces convex elements, remeshing or subdivision schemes used in simulations based on tetrahedra are not necessary, leading to less elements after such operations. We propose various operators for cutting the polyhedral discretization. Our method can handle arbitrary cut trajectories, and there is no limit on how often elements can be split.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

1. Introduction

The finite element method (FEM) has become a central tool in computer graphics, and it is widely used for physically-based animation of deformations, fracture, fluids, smoke, or other affects. Most methods discretize the computational domain by tetrahedral or hexahedral elements and linear or trilinear interpolants, respectively. While such formulations lead to simple and efficient computations, they share limitations when it comes to topological changes during the simulation. In cases of fracture or cutting, for instance, the elements have to be split such that the discretization conforms to the new object boundary. This remeshing comes at a cost and potentially leads to ill-shaped elements leaving the computations unstable. Ill-shaped elements can be removed from the discretization by inserting new simulation nodes and remeshing the domain [She03]. Unfortunately, this is an inherently non-local and costly operation.

Many researchers have proposed methods to alleviate the aforementioned problems. A simple and efficient approach consists of restricting the split to the faces between two elements. This works well for high resolution meshes and fracture. Precise cutting, however, is not possible. Other algorithms duplicate elements, but are still somewhat limited by the initial mesh resolution [MBF04]. Meshless FEM, as suggested recently, avoids the mesh entirely. The lack of connectivity, however, requires additional processing and data structures to identify topologically separated particles and boundary conditions [PKA*05, SOG06]. Furthermore, dynamic up- and downsampling has to be performed in the vicinity of newly created boundaries, which tends to be non-trivial.

The central contribution of this paper is a novel approach to compute elastic deformations based on FE discretization of the 3D domain into convex elements. Our method is specifically useful for simulations involving topological changes of the simulation domain, as experienced during cutting or fracture. It significantly reduces computations necessary for mesh maintenance. In addition, it allows for an accurate representation of the cut or fracture surface. We are able to remove sliver elements in an entirely local operation, making the simulation robust and numerically stable.

To accomplish the FE discretization, we employ a recently introduced generalization of barycentric interpolants to convex simplicial polytopes [JLW07] for the design of our basis functions. The mathematical properties of the functions, such as positivity, partition of unity, and reproduction of linear polynomials, make them well suited for FEM. In particular, as the commonly used linear tetrahedral interpolants are obtained as special case for tetrahedral elements, the method is a seamless and natural extension of tetrahedral linear elements. Fig. 1 shows a simple example of the deformation of a single convex element with 24 nodes.

Nonlinear interpolation functions lead to non-constant strain within the elements, and require integration over the

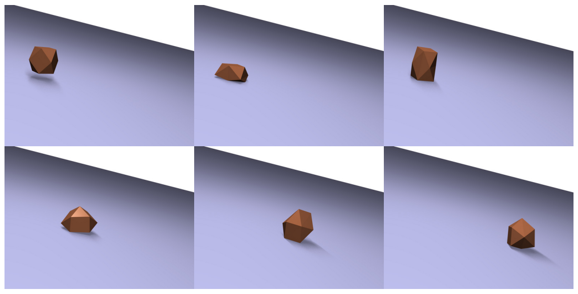


Figure 1: *An object consisting of a single element falls on a slope. Due to the nonlinearity of the basis functions, nonlinear deformations are possible even for a single element.*

elements in order to obtain the elastic energy. Such methods have rarely been used in computer graphics [RGTC98]. We demonstrate that in situations involving topological restructuring of the computational domain, the advantages of a flexible discretization easily outweigh the increased complexity of per-element integration.

Our technique uses linearized Cauchy strain and stiffness warping to avoid linearization artifacts for large deformations. We propose element splitting operations to accomplish the local restructuring of the simulation domain after cutting. A procedure for local sliver removal is also presented. We will show that the resulting maintenance operations after topological changes are minimal. Our method allows for progressive cuts through elements, arbitrary cut trajectories, and does not impose a limit on the number of times an element can be cut.

The remainder of this paper is structured as follows: Related work is discussed in more detail in Section 2. In Section 3 we introduce the interpolants and discuss their properties in the context of FEM. In addition, we present the technical details of our elasticity simulation. Our method for sliver removal is presented in Section 4. Section 5 describes the operations necessary to implement cutting. Finally, we show results and give performance figures in Section 6.

2. Related Work

Physically-based deformation of elastic solids was introduced to the field of computer graphics by Terzopoulos et al. [TPBF87]. Terzopoulos and Fleischer went on to simulate fracture in the form of tearing cloth [TF88]. The standard method to animate deformable objects has been FEM (e. g. [KGC*96, BNC96, OH99, DDCB01, GKS02, MG04]), although for many applications, especially those with real-time constraints, mass-spring systems have been used (the recent survey [NMK*06] gives a good overview). In both cases, the simulation domain is typically discretized using tetrahedral elements, even if nonlinear basis functions are considered for accuracy [RGTC98].

The easiest way to incorporate topological changes into such a system is to separate the material by removing springs (for mass-spring systems), or splitting the object along element boundaries [MG04]. Using this simple approach, newly created surfaces must conform to the initial discretization of the mesh. While this is acceptable in applications with hard real-time constraints, it is a severe limitation for animation. Accurate cutting is also not possible.

O'Brien et al. [OH99, OBH02] apply continuous remeshing to make the mesh discretization conform to the crack surface. [BG00, BGTG03] subdivide tetrahedra locally in order to accommodate the crack surface. This approach may lead to ill-conditioned elements, which cannot be robustly used in a FEM simulation. [SHGS06] try to avoid these problems by snapping the simulation nodes to the cut trajectory. In any case, the remeshing process is highly non-trivial, because ill-conditioned sliver tetrahedra have to be avoided in order to guarantee a stable simulation.

An alternative to remeshing is the virtual node algorithm introduced by Molino et al. [MBF04]. Instead of actually splitting simulation elements, the element that is to be split is duplicated. The surface is embedded in both elements and used for rendering and collision detection. The pitfalls of remeshing can be entirely avoided using this scheme, however, each element can be split at most three times. Hence, the original mesh resolution limits the resolution of fracture patterns or cuts, requiring a high input resolution for realistic results.

Meshless approaches for modeling elastic solids [MKN*04] do not require an underlying tetrahedral mesh. Interactions between particles are evaluated using the material distance between the particles. In the case of fracture, the material distance within the object might not be equal to the Euclidean distance any more, prompting recomputation of the shape functions of nearby particles [PKA*05]. This process is local, however, coverage issues complicate the fracturing process, making it necessary to adjust the particle distribution near cracks. Steinemann et al. [SOG06] have therefore reintroduced connectivity into a particle-based simulation. A distance graph is used to approximate the material distance. After cutting, this graph is modified to reflect the new situation. Still, resampling is necessary to maintain an adequate discretization near cracks.

Using nonlinear interpolation functions defined for convex elements [Wac71, Wac75, Flo03, HF06], it is possible to apply FEM directly on polygonal meshes (see [SM06] for a recent survey on 2D methods using different basis functions). One notable method in this class is the natural element method [Suk98], using natural neighbor interpolation. However, natural neighbor interpolation relies on Delaunay triangulation of the domain, and thus offers no advantages in settings with changing topology.

Warren [War96] generalized Wachspress coordinates to 3D. Recently, also Floater's mean value coordinates [Flo03] have been extended to 3D [JSW05, FKR05]. Ju et al. [JLW07] show connections of several methods that can be

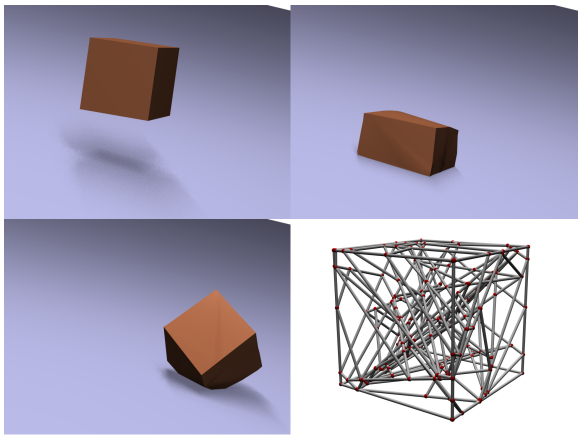


Figure 2: *A cube sampled with convex elements deforms on impact with the ground. The bottom-right picture shows the sampling. An initially hexagonal element has been subdivided with 30 random planes, resulting in 31 elements with 8 to 32 nodes.*

used to generate barycentric coordinates on convex 3D polyhedra. In our formulation, we use Floater's generalized construction as described in [JLW07] to obtain interpolation functions.

3. Elastic Deformation

Deformation of elastic material is governed by the equations of continuum elasticity, which are discretized using the finite element method. For an introduction to these topics, see for example [Chu96, Bat95]. In the following, we consider an object with material coordinates $\mathbf{x} = [x,y,z]^T$ deformed by a displacement field $\mathbf{u}(\mathbf{x}) = [u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]^T$.

After an introduction of the necessary details of continuum elasticity in Section 3.1, we propose interpolation functions for arbitrary convex polyhedra in Section 3.2. Using those, we discretize the continuous equations in order to derive the finite element formulation (Section 3.3). In the remaining subsections 3.4 and 3.5 we give details on numerical integration and dynamic simulation in our framework.

3.1. Continuum Elasticity

The elastic energy density of a deformable body is defined in terms of stress and strain within the object. For the latter, we employ the linear Cauchy strain ε, which depends on the Jacobian $\nabla\mathbf{u}$ of the deformation field $\mathbf{u}$:

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

The strain of the material in turn causes internal forces, represented by the 3×3 stress tensor σ. We assume a Hookean material, i.e., a linear stress–strain relationship, which gives

$$\sigma_{ij} \;=\; \sum_{k,l=1}^{3} \mathbf{C}_{ijkl}\, \varepsilon_{kl}\,, \quad i,j \in \{1,2,3\}, \tag{2}$$

with a 4-tensor $\mathbf{C}$ containing the elastic coefficients of the material. However, since ε and σ both are symmetric 3×3 matrices, their independent coefficients can also be written as 6D vectors. The strain, consisting of partial derivatives of $\mathbf{u}$, can then be written as

$$\varepsilon \;=\; \left[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial z}, \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}, \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right]^T. \tag{3}$$

Representing the stress by a 6D vector as well reduces the constitutive relation (2) to a simple 6×6 matrix product

$$\sigma \;=\; \mathbf{C}\varepsilon, \tag{4}$$

where the constitutive matrix $\mathbf{C}$ only depends on the material's elasticity modulus and Poisson ratio, controlling stiffness and volume preservation, respectively. In the following, we will use the 6D vector notation.

With stress and strain defined at any material point $\mathbf{x}$, the total elastic energy $U(\mathbf{u})$ can finally be computed as the integral of stress times strain over the object's volume

$$U(\mathbf{u}) \;=\; \frac{1}{2}\int_V \sigma^T\varepsilon \;=\; \frac{1}{2}\int_V \varepsilon^T\mathbf{C}\varepsilon. \tag{5}$$

3.2. Interpolation Functions for Convex Polyhedra

In order to discretize the energy equation (5) the continuum object is decomposed into a finite number of elements, and each node $i \in \{1,\dots,n\}$ of this decomposition is associated with a material position $\mathbf{x}_i$, a displacement value $\mathbf{u}_i = \mathbf{u}(\mathbf{x}_i)$, and a scalar shape function $\phi_i(\mathbf{x})$. With this, the continuous function $\mathbf{u}(\mathbf{x})$ can be approximated by

$$\mathbf{u}(\mathbf{x}) \;=\; \sum_{i=1}^{n} \mathbf{u}_i\,\phi_i(\mathbf{x})\,. \tag{6}$$

While graphics applications typically employ tetrahedral or hexahedral decomposition, our goal is to find an FEM formulation for convex polyhedra. This requires interpolation functions ϕ_i for convex polyhedra that are suitable for FEM computations.

For 2D simulations, Wachspress coordinates [Wac71, Wac75] have been used for finite element simulations on convex polygons. To our best knowledge, no FEM simulation method on 3D polyhedra has been proposed as of today. Recently, Wachspress coordinates [War96, JSWM05] and mean value coordinates [FKR05, JSW05, JLW07] have been generalized to 3D — although not in the context of FEM simulation. We propose to employ 3D mean value coordinates as interpolating shape functions for FEM simulations on convex polyhedra. Since these functions are usually treated in the context of parameterization and texture interpolation, we will briefly review them below and discuss their properties in the context of FEM.

Following the recent formulation of [JLW07], we derive the mean value interpolation function $\phi_i(\mathbf{x})$ corresponding to a particular vertex $\mathbf{x}_i$ of a convex polyhedron with k vertices $\{1,\dots,k\}$. The generalized barycentric coordinates are only

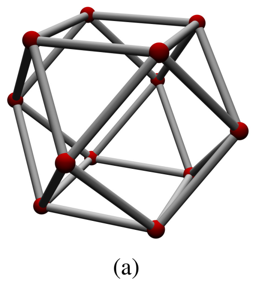


Figure 3: *(a) Wireframe view of the element shown in Fig. 1. (b) With triangulated faces.*

defined on convex polyhedra with triangular faces. We therefore triangulate the faces of our elements in order to compute the weight functions (see Section 3.2.1 for a discussion of numerical issues). Fig. 3 shows an element and a possible triangulation of its surface. Triangulating non-triangular faces is unproblematic compared to computing a tetrahedralization of the element.

Note that if two convex polyhedra share a common face, this face is necessarily planar, such that the exact nature of the triangulation does not change the shape of the elements. For the sake of consistency, we enforce this planarity constraint also for faces that lie on the boundary of the domain. Once barycentric coordinates defined for arbitrary polygonal faces become available, these can be used without changes to our method. One step in this direction has been taken by [Lan06].

Now consider the vertex $\mathbf{x}_i$ and its edge-incident one-ring neighbors enumerated as $\mathbf{x}_j$. The weight w_i is defined as a weighted sum of ratios of signed tetrahedra volumes

$$w_i(\mathbf{x}) = \sum_j \left[\frac{c_{j,j+1}}{V_{i,j,j+1}} + \frac{c_{i,j} V_{j-1,j+1,j}}{V_{i,j-1,j} V_{i,j,j+1}} \right], \qquad (7)$$

where the volume $V_{a,b,c} = V(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x})$ for vertex indices a, b, and c, and

$$c_{a,b} = \frac{\|(\mathbf{x}_a - \mathbf{x}) \times (\mathbf{x}_b - \mathbf{x})\|}{6} \arccos \left[\frac{(\mathbf{x}_a - \mathbf{x})^T (\mathbf{x}_b - \mathbf{x})}{\|\mathbf{x}_a - \mathbf{x}\| \|\mathbf{x}_b - \mathbf{x}\|} \right]. \qquad (8)$$

The mean value shape function ϕ_i is finally obtained by normalizing the weight function w_i:

$$\phi_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{l=1}^k w_l(\mathbf{x})}. \qquad (9)$$

See Fig. 4 for a visualization of ϕ_i.

The functions ϕ_i are true barycentric coordinates for convex polyhedra in the sense that they are positive inside the polyhedron (if it is convex), and that each point $\mathbf{x}$ inside the polyhedron can be written as a weighted sum of the vertices $\mathbf{x}_i$ with its coordinates as weights: $\mathbf{x} = \sum_{i=1}^k \phi_i(\mathbf{x}) \mathbf{x}_i$. This property implies partition of unity and reproduction of linear functions.

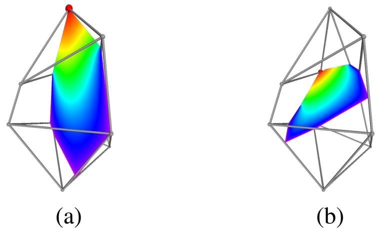


Figure 4: *Basis functions for the highlighted nodes visualized by hue interpolation on a plane through the element.*

Thus, the functions ϕ_i fulfill all properties necessary for convergence of our finite element approximation: They are positive and reproduce linear functions. Their support is limited to incident elements, yielding a sparse stiffness matrix. Continuity is C^1 within elements, and since they reduce to linear barycentric coordinates on the faces of the triangulation, the functions are C^0 continuous across element boundaries. Hence, the basis functions ϕ_i are in the Sobolev space H^1, and the finite elements approximation converges [Bat95].

In order to evaluate the strain, we require the first order partial derivatives of the shape functions ϕ_i. The derivatives of (7) and (9) can be computed analytically, the corresponding expressions are given in the appendix.

Note that in the case $k = 4$, the ϕ_i are linear, and equal to the shape functions commonly used in tetrahedral element discretizations. Hence, our method integrates seamlessly into FE methods using constant strain tetrahedra. Optimized code can be used for tetrahedral elements.

3.2.1. Numerical Issues

The shape functions defined as above are sums of volume ratios, which are problematic to compute if the volumes in (7) approach zero. This can occur in two situations: 1) if the point $\mathbf{x}$ lies on the boundary of the element, or 2) if a surface triangle has zero area.

The ϕ_i are not well defined on the boundary of the element, however, they converge to barycentric coordinates on the faces. Thus, this special case can easily be resolved. In practice, we can choose where to evaluate the interpolant during per element integration (see Section 3.4), and thus avoid evaluating ϕ_i or $\nabla\phi_i$ on the faces, except in the case of sliver elements. We remove such slivers from the discretization as described in Section 4.

Triangles with zero area make no net contribution to the weights w_i. Although not obvious from Equations (7) or (9), this can be easily seen in the equivalent formulation as presented by [JSW05]. Since it was specifically designed to be numerically robust, we also implemented the algorithm given in [JSW05] using numerical differentiation to compute the gradients of the basis functions. A comparison of these two methods showed no significant differences in simulation behavior.

3.3. Finite Element Discretization

With the shape functions defined in the last section, we discretize the continuous energy (5) using the approximation (6). If we consider a particular element e with vertices $i = 1,\dots,k$, only the shape functions $\phi_1,\dots,\phi_k$ of its vertices are non-zero. Hence, within e the displacement interpolation (6) is

$$\mathbf{u}(\mathbf{x}) = \underbrace{\begin{bmatrix} \phi_1(\mathbf{x}) & & & & \phi_k(\mathbf{x}) & & \\ & \phi_1(\mathbf{x}) & & \cdots & & \phi_k(\mathbf{x}) & \\ & & \phi_1(\mathbf{x}) & & & & \phi_k(\mathbf{x}) \end{bmatrix}}_{=:\mathbf{H}_e(\mathbf{x})} \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_k \end{bmatrix}, \tag{10}$$

with a $3 \times 3k$ interpolation matrix $\mathbf{H}_e(\mathbf{x})$. Based on this, the 6D strain vector (3) inside this element can then be written as

$$\varepsilon(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \mathbf{H}_e(\mathbf{x})\,\hat{\mathbf{u}} =: \mathbf{B}_e(\mathbf{x})\,\hat{\mathbf{u}}. \tag{11}$$

From the stress-strain relationship (4) we get the element's energy density similar to the continuous formulation (5):

$$U_e = \frac{1}{2}\hat{\mathbf{u}}^T \left(\int_{V_e} \mathbf{B}_e^T \mathbf{C} \mathbf{B}_e \right) \hat{\mathbf{u}} =: \frac{1}{2}\hat{\mathbf{u}}^T \mathbf{K}_e \hat{\mathbf{u}}. \tag{12}$$

The element's $3k \times 3k$ stiffness matrix $\mathbf{K}_e$ is built by integrating products of partial derivatives of the shape functions. In contrast to tetrahedral meshes with linear basis functions, these partial derivatives are not constant for arbitrary convex elements. Hence, we perform numerical integration to compute $\mathbf{K}_e$, which will be discussed in Section 3.4. Note that for linear elasticity, $\mathbf{K}_e$ is only computed in the rest state of the material, i. e. in a state where all elements are guaranteed to be convex (see also Section 6.1).

Once the elements' stiffness matrices $\mathbf{K}_e$ are precomputed, the global $3n \times 3n$ stiffness matrix $\mathbf{K}$ is assembled [Bat95]. If we denote by $\mathbf{U}$ the vector of nodal displacements $[\mathbf{u}_1^T,\dots,\mathbf{u}_n^T]^T$, the discrete version of the elastic energy (5) becomes

$$U(\mathbf{u}) = \frac{1}{2}\mathbf{U}^T \mathbf{K} \mathbf{U}. \tag{13}$$

3.4. Integration

In order to compute the per element stiffness matrix $\mathbf{K}_e = \int_{V_e} \mathbf{B}_e^T \mathbf{C} \mathbf{B}_e$, we have to integrate over each element. For linear tetrahedral elements, these integrals are trivial to compute since $\mathbf{B}_e$ is constant over the element. For other simple element shapes, for example hexahedral elements, integrals can be evaluated using Gauss quadrature. For irregularly shaped elements, such quadrature rules are unwieldy. Instead, we approximate the integrals using a low number of sample points $\mathbf{p}$ heuristically placed throughout the element. In our implementation, we use one integration sample per vertex of the element, plus one sample for each face of the triangulation of the element surface. We place the vertex integration samples between the element centroid $\mathbf{c}$ and the vertex $\mathbf{x}_i$, at $\mathbf{p}_i = 0.8\mathbf{x}_i + 0.2\mathbf{c}$. The face samples are placed similarly, at $\mathbf{p}_f = 0.9\mathbf{c}_f + 0.1\mathbf{c}$, where $\mathbf{c}_f$ is the face centroid. The exact location of these sample points does not have a critical influence on the simulation result.

Using the same one-rings as in Section 3.2, we define the volume fraction μ_i associated with the vertex i as

$$\mu_i^e = \frac{\sum_j V(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_{j+1}, \mathbf{c})}{3V_e}. \tag{14}$$

Similarly, the volume fraction for the face sample of face f with vertices j_1, j_2, and j_3 is

$$\kappa_f^e = \frac{V(\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \mathbf{x}_{j_3}, \mathbf{c})}{V_e}. \tag{15}$$

The element stiffness matrix $\mathbf{K}_e$ is then computed as

$$\mathbf{K}_e = \sum_i \frac{\mu_i^e}{2} \mathbf{B}_e^T(\mathbf{p}_i)\,\mathbf{C}\mathbf{B}_e(\mathbf{p}_i) + \sum_f \frac{\kappa_f^e}{2} \mathbf{B}_e^T(\mathbf{p}_f)\,\mathbf{C}\mathbf{B}_e(\mathbf{p}_f). \tag{16}$$

We have compared this method with Monte Carlo integration with a high number of samples (around 10000) per element, and have found no tangible difference in the behavior of the simulation.

In the special case of a tetrahedral element, we use only one integration point. Note that while computing the element stiffness matrix for arbitrary elements by integration is more complex than in the tetrahedral case, this has only minor impact on the overall simulation complexity. For linear elasticity, the stiffness matrices of the elements are constant throughout the simulation, unless the discretization is changed. This can happen in simulations involving adaptive refinement, or due to element splitting after fracture or cutting. Since stiffness matrices are computed in a preprocess, the computational complexity during the actual simulation is mainly dependent on the total number of nodes (see also Section 6).

3.5. Simulation Loop

The discrete energy (13) leads to the discrete equations of motion

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{D}\dot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}, \tag{17}$$

where $\mathbf{M}$ and $\mathbf{D}$ are the mass and damping matrices, respectively, and $\mathbf{F}$ represents external forces [Bat95].

We employ mass lumping to obtain a diagonal mass matrix $\mathbf{M}$. Assuming constant density per element, the mass of each element can be trivially computed from its volume. We then take the volume fractions μ_i^e introduced in Section 3.4

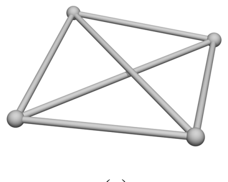


Figure 5: *Sliver elements: (a) A tetrahedral sliver element. Note that all faces can have reasonable areas, and no edge is too short. (b) Allowing arbitrary convex polyhedra can lead to more complex slivers.*

to obtain a mass for each node by summation over all incident elements:

$$m_i = \sum_e \mu_i^e V_e \rho_e, \tag{18}$$

where ρ_e is the density of element e. Our experiments suggest that the volume ratio μ_i^e is a good approximation of the integral over the shape function ϕ_i within the element.

The linear elasticity model (3) is not invariant to rotations. We use stiffness warping [MG04] in order to control linearization artifacts. This method requires that we compute per-element rotation matrices $\mathbf{R}_e$. The shape matching method described by Müller et al. [MG04] only works for tetrahedral elements. Therefore, we adopt the registration method presented by Horn [Hor87] instead, which has the additional advantage of being stable even for degenerate planar elements. Once the per-element rotations are known, the global stiffness matrix $\mathbf{K}$ has to be reassembled using the rotated element stiffness matrices $\mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e$.

Finally, implicit Euler integration is used to solve for the dynamic behavior of the object. Since most real-world deformable objects are strongly damped, the numerical damping introduced by the integration scheme is acceptable. For undamped simulations, symplectic integration can be used [KWT*06].

Note that even though the construction of the basis function is more complicated than the construction using constant strain tetrahedra or other simple elements, the complexity of the resulting equation system only depends on the number of nodes. Hence, using arbitrary convex polytopes instead of simpler element shapes does not result in a slower simulation. On the contrary, after some topological changes in the domain, tetrahedral remeshing is likely to have produced more elements and nodes than necessary for arbitrary convex elements.

4. Sliver Removal

Contrary to tetrahedral meshes, it is unclear what criteria determine the quality of a convex polyhedral element. However, our experiments suggest that bad elements are typically almost planar. These elements give rise to numerical problems during simulation, since the gradients of basis functions inside such elements cannot be evaluated robustly. Following Shewchuck [She02a, She02b], we call elements that are (almost) planar due to degenerated edges *needles*, while planar elements without degenerated edges are called *slivers*. Needles are easy to avoid: Whenever a node is created (during initial meshing or remeshing due to topological changes), it is snapped to existing nodes if incident edges would become too short. Most widely available meshing software, such as TetGen, can control the minimum edge length, and checking for nearby nodes during cutting is relatively easy.

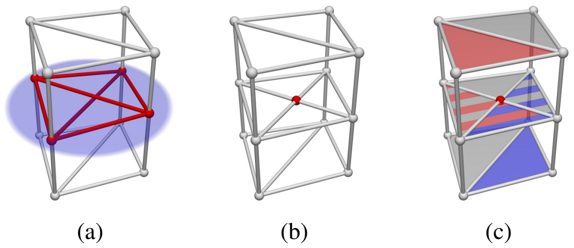


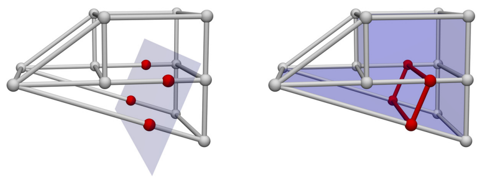
Figure 6: *Removing sliver elements: (a) A sliver element and its neighbors. (b) New nodes are created at edge intersections. (c) After tessellating the sliver plane, new faces are connected to their neighboring elements. The faces in the sliver plane are colored with the color of both elements they are connected to. Note that the shape of the adjacent elements is not changed, only their connectivity is modified to eliminate the sliver.*

Slivers are more problematic. One such element configuration in the tetrahedral case is shown in Fig. 5 (a). For general convex polyhedra, more complicated sliver elements are also possible, see Fig. 5 (b) for an example.

Slivers are notoriously hard to avoid. Remeshing algorithms based on constrained Delaunay tetrahedralization insert more nodes into adjacent elements and remesh the neighborhood of the sliver element. This process is costly, and is not guaranteed to be local [She03].

Since we are not restricted to tetrahedral elements, we can instead merge the sliver element with neighboring elements. This process consists of the following steps (see Fig. 6 for an illustration): We first compute a least-squares plane through the element, which we call the *sliver plane*. All vertices of the sliver element are projected onto the sliver plane. If this creates nodes that are too close to each other, they are merged. Then, all edges in the sliver plane are intersected and new nodes are inserted at the intersection points. Finally, the sliver plane is retessellated and each new face is connected to the two elements that were attached to the old faces it intersects. The sliver element can then be deleted.

Note that this procedure does not change the volume of the adjacent elements. Therefore, no new degenerate elements are created by removing the sliver.



(a) (b)

Figure 7: *(a) The bottom element is split along a plane. New simulation nodes (red) are added where the cutting plane intersects the original geometry of the element. (b) New integration samples are created in all elements that were changed by the split (shaded blue). Not all neighboring elements need to be updated.*

If the elements were tetrahedral before the sliver was removed, exactly one new node is created using this technique. The neighboring elements have five nodes each after the sliver is deleted. In more complex cases, more nodes might be inserted.

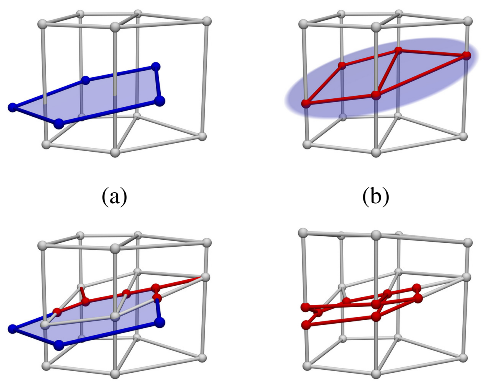
Projecting the nodes of a sliver element onto the sliver plane may lead to slightly non-convex elements in the neighborhood. However, this is not a problem for our method, since mean value coordinates, from which our basis functions are derived, are well-behaved even for non-convex elements [JSW05].

5. Cutting

Changing the topology of the simulation mesh reveals the strengths of our method. As our only requirement to the mesh is that all elements must be convex, maintaining a valid simulation mesh after cutting operations is easy. In this section, we review the atomic operations necessary to cut meshes consisting of convex elements. We first discuss the case of elements that are split entirely by a plane, before moving on to progressive cuts through elements.

5.1. Splitting Elements

Splitting a convex polytope along a plane results in two convex polytopes. Thus, after planar element split operations, no remeshing is necessary in order to maintain a valid discretization of the simulation domain. Wherever the splitting plane intersects existing edges, new simulation nodes are created. We compute displacement samples for the new nodes using our interpolant **u**, which is linear on the edges of the discretization. During these edge splits, care has to be taken not to create nodes too close to existing nodes to avoid degenerate edges. In practice, nodes that would be created too close to existing nodes are snapped to the existing geometry. For all elements that were changed in the process, new integration samples are computed. In order to avoid computing integration samples multiple times, reinitialization of element integration samples is deferred until the end of the timestep. Fig. 7 illustrates the procedure. Note



(a) (b) (c) (d)

Figure 8: *(a) Two elements are cut with a polygonal cut shape (blue). (b) All intersected elements are split along the cutting plane. (c) The polygon edges are intersected with the existing edges of the mesh, and new nodes are inserted at intersections. Faces in the cutting plane are split to create a consistent tessellation. (d) Nodes inside the cut shape are duplicated, the material is separated along the cut.*

that t-junctions might be created during splitting. This is not a problem, since the only requirement to the discretization is that all elements are convex.

5.2. Progressive Cuts

When progressive cuts are considered, elements are not necessarily split entirely in a single timestep. Instead, we have to deal with an arbitrary surface intersecting the element [BGTG03, SHGS06]. We therefore describe how to cut the simulation mesh with arbitrary planar polygons. Any given cut surface can be tessellated and the simulation mesh is sequentially cut with each of the resulting polygons.

As our *cut shape*, consider a polygon with vertices $\mathbf{p}_1 \dots \mathbf{p}_k$, lying in a *cutting plane* P. We first split all elements that intersect the cut shape along P, as described in Section 5.1. Then, we insert simulation nodes at the polygon points $\mathbf{p}_1 \dots \mathbf{p}_k$, and connect them with edges. The polygon edges are intersected with edges in the simulation mesh and all intersection points are added as new simulation nodes. The faces in the cut plane are now split to accommodate the new nodes and edges. We enforce that all generated faces are convex. Finally, all simulation nodes inside the cut shape can be duplicated, and their incident elements are separated along the cutting plane. Fig. 8 illustrates the necessary steps.

If there are no simulation nodes inside the cut shape, the cut cannot open. In such cases, we insert one additional simulation node at the centroid of the cut shape, and connect it with edges to the polygon nodes at $\mathbf{p}_1 \dots \mathbf{p}_k$, thus creating a pocket in the material.

Note that for non-tetrahedral elements, cuts through a single element might not be planar. In these cases, the surface

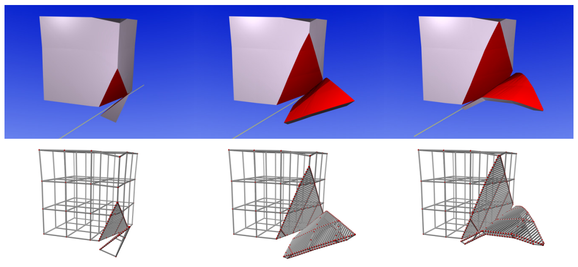


Figure 9: *A block of material is sliced. Very few additional elements are created as tetrahedral subdivision is not necessary. The bottom row shows the elements. Note that even though many nodes are created to accurately represent the cut surface, elements do not need to be split.*

within the element has to be tessellated after its edges have been intersected. The element is then sequentially cut with all faces of the tessellation as described above. We approximate the surface in each element by the triangulation of the intersection points on edges of the elements.

6. Results and Discussion

Table 1 gives simulation times for the scenes shown here, as well as element and node counts. Timings were measured on a Pentium 4, 3GHz, and do not include rendering time.

Fig. 1 shows a single element with 24 nodes that falls on a slope and deforms on impact. Note that due to the nonlinearity of basis functions, even a single element can undergo nonlinear deformations. Another example of elasticity using convex polyhedra is shown in Fig. 2. The elements in this example were created by repeatedly splitting a cube with random planes. The elements in this example have 8 to 32 nodes.

Simple cuts are shown in Fig. 9. A cube consisting of $3 \times 3 \times 3$ hexagonal elements is sliced. During the first cut, three elements are added. Note that contrary to an implementation using tetrahedral elements only, elements do not have to be subdivided in order to accommodate new vertices, leading to fewer elements and nodes. Nevertheless, the cut surface is represented accurately, and smooth, nonlinear deformations are possible.

Fig. 10 shows more complicated cuts. Elements created by cutting operations are regular elements and can be cut again. We used TetGen to generate an initial tetrahedral mesh. After all cuts are complete, 51% of the elements are still tetrahedral, the elements with the highest number of nodes has 27 nodes. Performing the same sequence of cuts using a state of the art tetrahedral subdivision method [SHGS06] results in more than 75000 nodes and more than 300000 elements, even if the cuts are executed non-progressively. The exact number of nodes and elements depends on snapping thresholds.

	Start		End		avg. time/
Fig.	#Nodes	#Elem	#Nodes	#Elem	frame [s]
1	24	1	24	1	0.012
2	184	31	184	31	0.22
9	64	27	1214	60	0.8 (0.04)
10	24079	8278	46132	44118	6.08 (3.56)

Table 1: *Node count, element count, and computation time. The time in parenthesis is the computation time for the dynamic update not including recomputation of basis functions.*

Highly optimized code is available for tetrahedral FE simulation. On purely tetrahedral models, our implementation of the above algorithm is slower than these methods. However, the theoretical complexity in these cases is the same. As soon as topological changes are considered, the number of elements and nodes in the discretization grows faster when only tetrahedral elements are allowed.

6.1. Limitations

Our method is limited to linear elasticity. Nonlinear elasticity would require evaluating the basis functions and their gradients also for the deformed state of the simulation mesh. This is possible in principle, but the elements may not be convex in the deformed state. While our basis functions are smooth even for non-convex elements (contrary to Wachspress coordinates), they lose the property of positivity. One possible approach might be to subdivide the deformed elements into convex parts in each time step. Irving et al. [ITF06] treat problems arising for nonlinear strain in hexahedral elements, however, their method is not directly applicable to arbitrary convex polyhedra.

In shell-like situations, strain concentrates at element boundaries. This artifact can be observed in Fig. 9, see also the accompanying video. Note that this limitation applies to all finite element methods with only C^0 continuity across element boundaries, including constant strain tetrahedra.

So far, the faces of elements have to be decomposed into triangles in order to evaluate the basis functions. Although we have not experienced numerical difficulties caused by face triangulation, a formulation that does not require triangular faces would of course be more elegant.

7. Conclusion and Future Work

We have presented a novel finite element method based on discretization of the domain into convex elements. The basis functions within these elements are derived from recently introduced 3D mean value coordinates. These basis functions fulfill all necessary criteria to prove convergence of our method. In the case of tetrahedral elements, the functions are linear, and constant strain tetrahedra emerge as a special case of our approach.

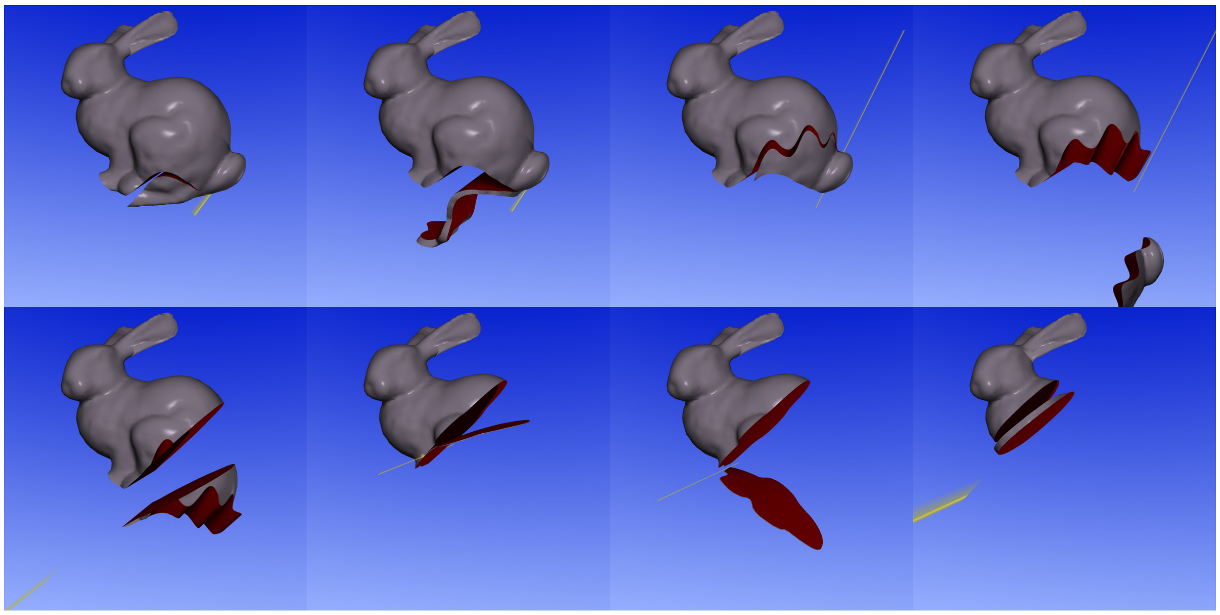


Figure 10: *Slicing the Stanford Bunny. The cut trajectories are accurately represented. We can cut extremely thin slices without mesh restructuring — the simulation method is stable in these cases.*

Our method is more flexible than a pure tetrahedron-based approach. Splitting elements does not require remeshing, reducing the number of elements and nodes created during cutting operations, especially if consecutive cuts through the same region are considered.

As future work, we plan to do a thorough numerical analysis of the proposed approach, including comparisons regarding accuracy. The effect of our integration heuristic on accuracy and convergence needs to be examined. Meshing a domain into a set of convex polyhedra, as opposed to tetrahedra, is also an open problem.

Acknowledgments

We would like to thank Bernd Bickel for his work on the raytraced illustrations and Denis Steinemann for comparison data. We are grateful to Christoph Schwab for the helpful discussion. Martin Wicke is funded by the Swiss National Commision for Technology and Innovation (CTI), project no. 7560.1 ESPP-ES.

References

[Bat95] BATHE K.-J.: *Finite Element Procedures*. Prentice Hall, 1995.

[BG00] BIELSER D., GROSS M.: Interactive Simulation of Surgical Cuts. In *Proceedings of Pacific Graphics'00* (2000), pp. 116–125.

[BGTG03] BIELSER D., GLARDON P., TESCHNER M., GROSS M.: A State Machine for Real-Time Cutting of Tetrahedral Meshes. In *Proceedings of Pacific Graphics'03* (2003), pp. 377–386.

[BNC96] BRO-NIELSEN M., COTIN S.: Real-time Volumetric Deformable Models for Surgery Simulation Using Finite Elements and Condensation. In *Proceedings of Eurographics'96* (1996), pp. 57–66.

[Chu96] CHUNG T. J.: *Applied Continuum Mechanics*. Cambridge University Press, New York, 1996.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic Real-Time Deformations using Space and Time Adaptive Sampling. In *Proceedings of SIGGRAPH'01* (2001), pp. 31–36.

[FKR05] FLOATER M. S., KOS G., REIMERS M.: Mean Value Coordinates in 3D. *Computer Aided Geometric Design 22* (2005), 623–631.

[Flo03] FLOATER M. S.: Mean Value Coordinates. *Computer Aided Geometric Design 20*, 1 (2003), 19–27.

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A Simple Framework for Adaptive Simulation. In *Proceedings of SIGGRAPH'02* (2002), pp. 281–290.

[HF06] HORMANN K., FLOATER M. S.: Mean Value Coordinates for Arbitrary Planar Polygons. *Transactions on Graphics 25*, 4 (2006), 1424–1441.

[Hor87] HORN B. K. P.: Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America 4* (1987), 629–642.

[ITF06] IRVING G., TERAN J., FEDKIW R.: Tetrahedral and Hexahedral Invertible Finite Elements. *Graphical Models 68*, 2 (2006), 66–89.

[JLW07] JU T., LIEPA P., WARREN J.: A General Geometric Construction of Coordinates in a Convex Simplicial Polytope. *Computer Aided Geometric Design* (2007). preprint.

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean Value Coordinates for Closed Triangular Meshes. In *Proceedings of SIGGRAPH'05* (2005), pp. 561–566.

[JSWM05] JU T., SCHAEFER S., WARREN J., M.DESBRUN: Geometric Construction of Coordinates for Convex Polyhedra using Polar Duals. In *Proceedings of the Symp. on Geometry Processing'05* (2005), pp. 181–186.

[KGC*96] KOCH R. M., GROSS M. H., CARLS F. R., VON BÜREN D. F., FRANKHAUER G., PARISH Y. I. H.: Simulating Facial Surgery Using Finite Element Models. In *Proceedings of SIGGRAPH'96* (1996), pp. 421–428.

[KWT*06] KHAREVYCH L., WEIWEI, TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, Variational Integrators for Computer Animation. In *Proceedings of the Symp. on Computer Animation'06* (2006), pp. 43–51.

[Lan06] LANGER T.: Spherical Barycentric Coordinates. In *Proceedings of the Symp. on Geometry Processing'06* (2006), pp. 81–88.

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A Virtual Node Algorithm for Changing Mesh Topology during Simulation. In *Proceedings of SIGGRAPH'04* (2004), pp. 385–392.

[MG04] MÜLLER M., GROSS M.: Interactive Virtual Materials. In *Proceedings of Graphics Interface'04* (2004), pp. 239–246.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point-Based Animation of Elastic, Plastic and Melting Objects. In *Proceedings of the Symp. on Computer Animation'04* (2004), pp. 141–151.

[NMK*06] NEALEN A., MULLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical Modeling and Animation of Ductile Fracture. In *Proceedings of SIGGRAPH'02* (2002), pp. 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of SIGGRAPH'99* (1999), pp. 137–146.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTR&#233; P., GROSS M., GUIBAS L. J.: Meshless Animation of Fracturing Solids. In *Proceedings of SIGGRAPH'05* (2005), pp. 957–964.

[RGTC98] ROTH M., GROSS M., TURELLO S., CARLS F. R.: A Bernstein-Bézier Based Approach to Soft Tissue Modeling. In *Proceedings of Eurographics'98* (1998), pp. 285–294.

[She02a] SHEWCHUCK J.: What Is a Good Linear Finite Element? Interpolation, Conditioning, and Quality Measures. In *Proceedings of the 11th International Meshing Roundtable* (2002), pp. 115–126.

[She02b] SHEWCHUCK J.: What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures. unpublished extended version, 2002.

[She03] SHEWCHUCK J.: Updating and Constructing Constrained Delaunay and Constrained Regular Triangulations by Flips. In *Proceedings of the 19th Annual Symposium on Computational Geometry* (2003), pp. 181–190.

[SHGS06] STEINEMANN D., HARDERS M., GROSS M., SZEKELY G.: Hybrid Cutting of Deformable Solids. In *Proceedings of the IEEE VR'06* (2006), pp. 35–42.

[SM06] SUKUMAR N., MALSCH E. A.: Recent Advances in the Construction of Polygonal Finite Element Interpolants. *Archives of Computational Methods in Engineering 13*, 1 (2006), 129–163.

[SOG06] STEINEMANN D., OTADUY M. A., GROSS M.: Fast Arbitrary Splitting of Deforming Objects. In *Proceedings of the Symp. on Computer Animation'06* (2006), pp. 63–72.

[Suk98] SUKUMAR N.: *The Natural Element Method in Solid Mechanics*. PhD thesis, Northwestern University, Chicago, USA, 1998.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. In *Proceedings of SIGGRAPH'88* (1988), pp. 269–278.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically Deformable Models. In *Proceedings of SIGGRAPH'87* (1987), pp. 205–214.

[Wac71] WACHSPRESS E. L.: A Rational Basis for Function Approximation. In *Lecture Notes in Mathematics* (1971), vol. 228, Springer, pp. 223–252.

[Wac75] WACHSPRESS E. L.: *A Rational Finite Element Basis*. Academic Press, 1975.

[War96] WARREN J.: Barycentric Coordinates for Convex Polytopes. *Advances in Computational Mathematics 6* (1996), 97–108.

Appendix A: Derivatives of Shape Functions

Consider a convex polytope with vertices at positions $\mathbf{x}_i$, $i = 1 \dots k$. Recalling (9) from page 4, we obtain for the gradient of the basis function ϕ_i

$$\nabla\phi_i = \frac{\partial\phi_i}{\partial\mathbf{x}} = \frac{\nabla w_i \sum_{l=1}^{k} w_l - w_i \sum_{l=1}^{k} \nabla w_l}{\left(\sum_{l=1}^{k} w_l\right)^2}. \tag{19}$$

With the one-ring vertices of vertex i enumerated as $\mathbf{x}_j$, the gradient of the weight w_i is

$$\nabla w_i = \sum_j \Big[\frac{\nabla c_{j,j+1} V_{i,j,j+1} - c_{j,j+1} \nabla V_{i,j,j+1}}{V_{i,j,j+1}^2} + \frac{(\nabla c_{i,j} V_{j-1,j+1,j} + c_{i,j} \nabla V_{j-1,j+1,j}) V_{i,j-1,j} V_{i,j,j+1}}{V_{i,j-1,j}^2 V_{i,j,j+1}^2} - \frac{c_{i,j} V_{j-1,j+1,j} (V_{i,j-1,j} \nabla V_{i,j,j+1} + \nabla V_{i,j-1,j} V_{i,j,j+1})}{V_{i,j-1,j}^2 V_{i,j,j+1}^2} \Big]. \tag{20}$$

The gradient of a tetrahedron volume $V_{a,b,c}$ has the magnitude of one third the triangle area $A(a,b,c)$ and points in the direction of the triangle normal:

$$\nabla V_{a,b,c} = \frac{\|(\mathbf{x}_c - \mathbf{x}_a) \times (\mathbf{x}_b - \mathbf{x}_a)\|}{6}. \tag{21}$$

We define $\mathbf{d}_i = \mathbf{x} - \mathbf{x}_i$ and $\hat{\mathbf{d}}_i = \mathbf{d}_i / \|\mathbf{d}_i\|$. The gradient of the term $c_{a,b}$ is given by

$$\nabla c_{a,b} = \frac{1}{6} \Big[\big(\hat{\mathbf{d}}_a \cdot \hat{\mathbf{d}}_b\big) \big(\hat{\mathbf{d}}_a \|\mathbf{d}_b\| + \hat{\mathbf{d}}_b \|\mathbf{d}_a\|\big) - \mathbf{d}_a - \mathbf{d}_b + \frac{\arccos\big(\hat{\mathbf{d}}_a \cdot \hat{\mathbf{d}}_b\big)}{\|\hat{\mathbf{d}}_a \times \hat{\mathbf{d}}_b\|} \left[(\hat{\mathbf{d}}_a \times \hat{\mathbf{d}}_b) \times (\mathbf{x}_b - \mathbf{x}_a) \right] \Big]. \tag{22}$$

Note that since $\frac{\arccos(\hat{\mathbf{d}}_a \cdot \hat{\mathbf{d}}_b)}{\|\hat{\mathbf{d}}_a \times \hat{\mathbf{d}}_b\|} = \frac{\alpha}{\sin\alpha}$ and $\lim_{x \to 0} \frac{x}{\sin x} = 1$, (22) can be robustly evaluated in all cases.