# Shape-Preserving Animation of Deformable Objects

Sebastian Martin, Christoph Huber, Peter Kaufmann, Markus Gross

Computer Graphics Laboratory, ETH Zurich
Email: {smartin,chrhuber,peterkau,grossm}@ethz.ch

## Abstract

We present a novel approach for animating elastically deformable solids in a shape-preserving manner. Standard approaches to animate this kind of objects are based on classic FEM discretizations of the elasticity theory, combined with embedding techniques to deform highly-detailed object geometries. However, these approaches are usually not able to preserve fine geometric features at sub-element scales, showing visually disturbing deformations. We propose to use Green Coordinates (GC) for the representation of the deformation field to get shape-preservation 'by construction' and describe how to discretize the elastic energy using these cage-based coordinates. By linearizing the deformation field we arrive at a simple approach which leads to just a few additional terms compared to classic FEM discretizations.

## 1 Introduction

Physically-based simulation is one of the most important tools for the simple and efficient creation of realistic and visually attractive animations. The finite element method (FEM) is a widely used approach for the simulation of elastically deformable models. Due to its sound theoretical guarantees it is applied in engineering applications where stability and accuracy of the solution are crucial. Typical graphics applications have somewhat different requirements. High accuracy is less important, while efficiency, visual plausibility and esthetics are paramount. Often, these goals can be achieved by a larger variety of approaches than in engineering. As an example, the purely geometric shape matching approach of [MHTG05] shows excellent efficiency and stability properties, while just mimicking physical behavior.

For the task of geometric modeling, a large variety of different techniques exist. Space deformation techniques define the deformation of a 3D space, causing the embedded geometry to deform accordingly. Coordinate-based techniques, where the deformed space is restricted to a closed domain, have been investigated extensively in the past years. In particular, Lipman et al. [LLCO08] recently presented Green Coordinates (GC) that have the additional property of being shape preserving, in the sense that they minimize angular distortion. This helps to greatly improve the visual quality of the deformed geometry.

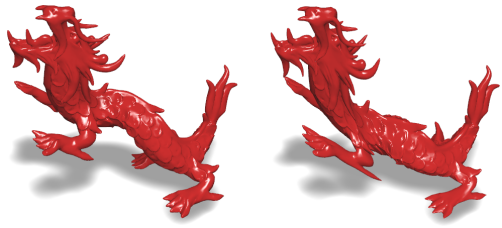

Figure 1: Shape preserving deformation of the dragon model using our method (left, 156 DOFs) compared to FEM-based deformation (right, 162 DOFs).

While these space deformation techniques are directly applicable to static geometric modeling, the automatic generation of full animation sequences is more difficult to realize. This paper therefore presents an approach of animating deformable objects in a shape-preserving manner, analogous to how the GC approach preserves shape for modeling. This is realized by performing a Galerkin discretization of an elastic energy on the subspace defined by the GC. Independently of the discretization resolution, such an approach allows the preservation of small features. In contrast, linear FEM results in locally affine deformations, distorting features at the sub-element level.

Simulations resulting from our approach are necessarily different from simulations performed with classical FEM since the deformations are constrained to a different subspace. Contrary to the approximation spaces chosen in classical FEM, the

subspace implied by GC does not allow for convergence under refinement since analytic solutions to elasticity are not shape preserving. Nevertheless, with respect to graphics requirements, the proposed method is able to give novel and visually pleasing results.

## 2 Related Work

Physically-based animation was introduced to computer graphics by Terzopoulos et al. in their seminal work [TPBF87]. From the vast body of literature in the field, we only cite the most relevant references and refer the reader to [NMK+06] for a wider overview. Our approach makes use of the Galerkin discretization, which is also the basis of all other finite element methods [Hug00]. Typically, the FEM employs simple element shapes like tetrahedra (e.g. [OH99]) for conforming or hexahedra (e.g. [JBT04]) for non-conforming surface representations. In our approach, we will also use a non-conforming representation, where the represented surface is embedded in a surrounding simulation domain.

While our approach aims for the preservation of important features of the deformed models, other techniques of physically-based animation have their specific areas of application, which we briefly mention here for completeness. Meshless models (e.g. [MG04]) are particularly suited for applications requiring frequent remeshing. Model reduction techniques (e.g. [JF03]) achieve high simulation performances by restricting themselves to characteristic deformation modes. For geometry-driven shape matching approaches (e.g. [MHTG05]), stability and performance are the main characteristics.

Geometric modeling is another widely studied topic in computer graphics. We refer to [MJBF02, BS08] for surveys on the different classes of methods and pursue discussing the coordinate-based space deformation techniques relevant to our work. Classic barycentric coordinates are defined on simplicial domains like triangles (2D) and tetrahedra (3D) and have been known for centuries. Different attempts have been made to generalize them to more convex domains [Wac75, MLBD02, JSWM05]. Floater [Flo03] introduced Mean Value Coordinates which have later been generalized to 3D and were applied to surface deformation [FKR05, JSW05]. Joshi et al. introduced a different kind of cage-based coordinates called Harmonic Coordinates [JMD+07], which are defined on arbitrary non-convex polyhedral cages. All these coordinates are affine-invariant and are thus not able to preserve the shape of the enclosed geometry. Lipman et al. [LLCO08] introduced a new set of coordinates called Green Coordinates, having exactly this property. Only recently, Weber et al. [WBCG09] generalized GC to complex barycentric coordinates for the 2D case, however no generalization to 3D exists yet.

Next to the application of these coordinates to modeling tasks, different attempts have been made in graphics to take advantage of their properties in physical simulation problems. Wicke et al. [WBG07] defined basis functions by Mean Value Coordinates for FEM simulations, allowing for discretizations with more general convex elements. Martin et al. [MKB+08] generalized element types to arbitrary polyhedra by using Harmonic Coordinates, allowing them to circumvent complex remeshing operations during cutting, fracturing and adaptive simulations. In this paper, we follow the line of thought of these two previous works, however with a different goal in mind. By extending GC to act as a representation basis allows us to produce animations that are shape preserving. This can be compared to model reduction techniques [JF03] where a limited representation basis is constructed from a full, physically motivated basis, allowing for more efficient simulations in the reduced basis.

## 3 Elastic Solids

Before introducing GC (Section 4) and discussing their integration into a dynamic simulation framework (Section 5), basic concepts of continuum elasticity and its Galerkin discretization are reviewed in this section. More information on this topic can be found in [NMK+06] or [Hug00], for example.

We start with an object given in its rest state by $\bar{\mathbf{x}} \in \Omega \subset \mathbb{R}^3$ and describe the actual configuration by the deformation field $\mathbf{u} : \Omega \rightarrow \mathbb{R}^3$. The amount of stretch which the material undergoes during the deformation is measured by a $3 \times 3$ strain tensor. Typical choices are the nonlinear Green strain

$$\epsilon_G(\mathbf{u}) \;=\; \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T + \nabla\mathbf{u}(\nabla\mathbf{u})^T\right) \quad (1)$$

or the linearized Cauchy strain suited for small deformations

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

We will employ the linear Cauchy strain $\epsilon = \epsilon_C$ to get a simple linear elasticity formulation and will apply a corotation-based formulation to treat the common rotation artifacts [MG04].

For an elastic material, deformation leads to restoring forces which are represented by the $3 \times 3$ stress tensor $\sigma$. Assuming a Hookean material and representing the symmetric tensors by corresponding 6-vectors, the relation between stress and strain can be modeled by

$$\sigma(\mathbf{u}) = \mathbf{C}\epsilon(\mathbf{u}), \tag{3}$$

where $\mathbf{C}$ is the $6 \times 6$ material matrix containing the material's elastic coefficients. The internal elastic energy density is then simply the product of stress and strain. Assuming external forces $\mathbf{f}_e(\mathbf{x})$ acting on the object (e.g. gravity), the total potential energy of the object is

$$E(\mathbf{u}) = \frac{1}{2}a(\mathbf{u},\mathbf{u}) - (\mathbf{f}_e, \mathbf{u}), \tag{4}$$

where $a(\mathbf{u},\mathbf{v}) = \int_\Omega \epsilon(\mathbf{u})^T \mathbf{C}\epsilon(\mathbf{v})dV$ is a bilinear form and $(\mathbf{u},\mathbf{v}) = \int_\Omega \mathbf{u}^T\mathbf{v}dV$ is the standard $L^2$-inner product. Since we will be dealing with non-trivial integration domains, we approximate these integrals by voxel-based Gauss quadrature, similar to [MKB+08].

Finally, by taking the variational derivative of the energy, the restoring forces become $\mathbf{f}(\bar{\mathbf{x}}) = -\frac{\partial E}{\partial \mathbf{u}}$. These are essential to formulate the equations for the static ($\mathbf{f}(\bar{\mathbf{x}}) = 0$) and the dynamic problem ($\rho\ddot{\mathbf{u}} - \mathbf{f}(\bar{\mathbf{x}}) = 0$, with material density $\rho$).

The next step consists in performing a Galerkin discretization of the two PDEs above. By the structure of the PDEs, solutions to the stated problems must lie in the infinite-dimensional function space $H^1$, defined by $L^2$-integrable functions with $L^2$-integrable derivatives. We define a finite dimensional space $V_N \subset H^1$ by representing the solutions as

$$\mathbf{u}_N(\bar{\mathbf{x}}) = \sum_i^N \mathbf{u}_i\phi_i(\bar{\mathbf{x}}) \in V_N. \tag{5}$$

Note that the nodal basis functions $\phi_i$ (associated with positions $\bar{\mathbf{x}}_i$) must lie in $H^1$. Furthermore, they must be able to represent rigid body motions. This means that they have to build a partition of unity in order the represent translations and need to reproduce linear functions to represent rotations. As the basis functions are nodal, the degrees of freedom $\mathbf{u}_i$ directly correspond to the displacements at their respective nodes.

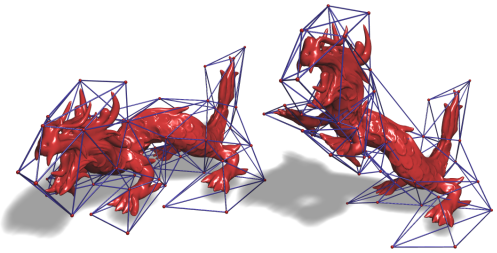

Figure 2: Embedded dragon model in undeformed (left) and deformed cage (right).

Considering the energy in $V_N$ instead of $H^1$ leads to its discrete version

$$E_N(\hat{\mathbf{u}}) = \frac{1}{2}\hat{\mathbf{u}}^T\mathbf{K}\hat{\mathbf{u}} - \hat{\mathbf{u}}^T\mathbf{f}, \tag{6}$$

where $\hat{\mathbf{u}}$ assembles all deformation vectors $\mathbf{u}_i$ into a single $3N$-dimensional vector. $\mathbf{K}_{ij} = a(\phi_i,\phi_j)$ are the $3 \times 3$ blocks of the stiffness matrix $\mathbf{K}$ and $\mathbf{f}$ consists of 3-vectors $\mathbf{f}_i = (\mathbf{f}_e, \phi_i)$. To simplify notation, whenever $a$ takes two scalar arguments, it evaluates to a $3 \times 3$ matrix $\mathbf{M} = a(s,t)$ with entries $\mathbf{M}_{kl} = a(s\,\mathbf{e}_k, t\,\mathbf{e}_l)$. Likewise, the vector $\mathbf{v} = (\mathbf{u}, s)$ has entries $\mathbf{v}_i = (\mathbf{u}, s\,\mathbf{e}_i)$.

Similar to the continuous case, the discrete forces can now be computed as $\mathbf{f}_i = -\frac{\partial E_N}{\partial \mathbf{u}_i}$, leading to the linear system

$$\mathbf{K}\hat{\mathbf{u}} = \mathbf{f} \tag{7}$$

for the static case and to the system of ODEs

$$\mathbf{M}\frac{\partial^2\hat{\mathbf{u}}}{\partial t^2} + \mathbf{K}\hat{\mathbf{u}} = \mathbf{f} \tag{8}$$

for the dynamic case, where $\mathbf{M}_{ij} = \mathbf{I}\cdot\int_\Omega \rho\phi_i\phi_j dV$ are the $3 \times 3$ blocks of the the mass matrix $\mathbf{M}$. If required, an additional damping term could also be added to model energy dissipation. However, since we are using a stable semi-implicit Euler integration scheme [BW98], there is no need for an additional damping term.

## 4 Green Coordinates

In this section, we will shortly review the important definitions and properties of GC, presented originally in [LLCO08]. Similar to other commonly used coordinates, GC are also cage-based and thus defined inside a closed polytope. Classic barycentric coordinates describe a point inside the cage as a linear combination of cage vertices. Deforming the cage's shape then leads to a deformation of the enclosed space (Fig. 2). GC additionally

incorporate the faces' normals to achieve shape-preserving deformations, i.e., deformations that locally only allow a limited amount of change in angles. Achieving this property is not possible using Mean Value Coordinates or Harmonic Coordinates [Flo03, JMD+07], since they are affine invariant [LLCO08]; affine transformations like shearing and anisotropic scaling violate shape-preservation.

More formally, GC define how a point $\bar{\mathbf{x}}$ in the interior of the undeformed cage is deformed to a new position $\mathbf{x}$ by a linear combination of vertex positions $\mathbf{x}_i$ and face normals $\mathbf{n}_j$ of the deformed cage as

$$\mathbf{x} = \sum_{i \in I_V} \mathbf{x}_i \phi_i(\bar{\mathbf{x}}) + \sum_{j \in I_T} s_j \mathbf{n}_j \psi_j(\bar{\mathbf{x}}), \quad (9)$$

where $I_V$ is the set of all vertices and $I_T$ the set of faces. While GC can be defined on general cage types we will focus on cages represented as triangle meshes for which they can be described analytically. In order to achieve shape preserving deformations, the scaling factor $s_j$ has to be chosen as

$$s_j = \frac{\sqrt{|\bar{\mathbf{v}}|^2 |\mathbf{w}|^2 - 2(\bar{\mathbf{v}} \cdot \bar{\mathbf{w}})(\mathbf{v} \cdot \mathbf{w}) + |\mathbf{v}|^2 |\bar{\mathbf{w}}|^2}}{\sqrt{8}\, \mathrm{area}(t_j)}, \quad (10)$$

where $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$ are two arbitrary undeformed edges of the triangle $t_j$, with corresponding deformed edges $\mathbf{v}$ and $\mathbf{w}$. Please note here that the normals nonlinearly depend on the cage vertex positions which will be important for the discretization. Using Green's function $G(\mathbf{x}, \mathbf{x}') = \frac{1}{4\pi|\mathbf{x}-\mathbf{x}'|}$ and the piecewise linear hat function $\Gamma_i(\mathbf{x}')$ defined on the triangle mesh and centered at vertex $i$, the coordinates are defined as

$$\phi_i(\mathbf{x}) = \int_{\mathbf{x}' \in N_i} \Gamma_i(\mathbf{x}') \frac{\partial G(\mathbf{x}, \mathbf{x}')}{\partial \mathbf{n}(\mathbf{x}')} dS_{\mathbf{x}'} \quad (11)$$

and

$$\psi_j(\mathbf{x}) = -\int_{\mathbf{x}' \in t_j} G(\mathbf{x}, \mathbf{x}') dS_{\mathbf{x}'}, \quad (12)$$

where the integration domain $N_i$ consists of all triangles adjacent to vertex $i$. Representing the cage by a triangle mesh allows analytic computation of the coordinates and their derivatives. For pseudocode of the actual coordinate computation we refer to [LLCO08]. The computation of their derivatives, which we need additionally for strain computations, can be derived straightforwardly by taking the derivatives in the according lines of their pseudocode.

It can easily be checked that all necessary conditions for a Galerkin discretization are fulfilled [LLCO08]: Inside the cage, the coordinates are $C^\infty$, which is sufficient since we will choose the computation domain to lie inside the cage. Furthermore, the vertex-based coordinates form a partition of unity such that constant deformations can be represented. Finally, GC do also reproduce linear functions as required for the representation of rotations.

Considering GC in an FEM context, we note that the approximated deformation field is spanned by vertex- and normal-based basis functions. Cage vertex positions are the actual degrees of freedom since triangle normals are completely defined by corresponding vertex positions. Therefore we have a nonlinear map from the DOF to the deformation field. This is different from standard Galerkin approaches where this map is linear and leads to a linear subspace of $H^1$. Here, GC span a non-linear subspace (i.e. a manifold) such that the classic linear theory is not valid in our case. This, however, did not cause any problems in our simulations.

# 5 GC Discretization

This section shows how we discretize the elastic solid model. First we formulate the problem nonlinearly and then show how we perform the linearization to arrive at our simple approach. Furthermore, we shortly explain how we handle corotation and boundary conditions in this setting.

## 5.1 Non-Linear Formulation

We have seen in the last section that GC fulfill all requirements for a Galerkin discretization as long as the problem domain $\Omega$ resides inside the cage. The problem domain can be represented by a high-resolution triangle mesh embedded in the volume of the cage. We do not present how to generate cages for given domains since they can easily be designed by hand using an arbitrary modeling tool. In the remainder we assume that our domain is equipped with a suitable triangular mesh representing a cage.

In order to formulate the Galerkin discretization, we first need a representation of the deformation field $\mathbf{u}(\bar{\mathbf{x}})$ in terms of GC. Representing both deformed and undeformed positions with GC and considering their difference we get

$$\mathbf{u}_G(\bar{\mathbf{x}}) = \sum_i \mathbf{u}_i \phi_i + \sum_j \mathbf{m}_j \psi_j, \quad (13)$$

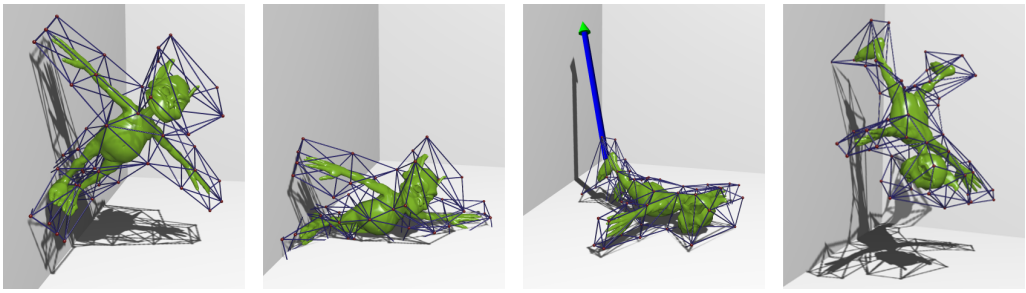

Figure 3: Animation of the goblin model showing external forces and collisions.

where $\mathbf{u}_i = \mathbf{x}_i - \bar{\mathbf{x}}_i$ and $\mathbf{m}_j = \mathbf{n}_j s_j - \bar{\mathbf{n}}_j$ (since $s_j = 1$ for the undeformed state).

Following the same line as classic Galerkin discretizations, we formulate the elastic energy using the new deformation field $\mathbf{u}_G$:

$$E_G = \frac{1}{2} a(\mathbf{u}_G, \mathbf{u}_G) - (\mathbf{f}_e, \mathbf{u}_G). \tag{14}$$

In order to derive internal forces, we again take the derivative of $E_G$ with respect to the degrees of freedom $\mathbf{u}_i$ (note that the $\mathbf{m}_j$ are nonlinear functions of the $\mathbf{u}_i$'s). Taking advantage of the linearity of $a(\cdot,\cdot)$ and $(\cdot,\cdot)$, the energy gradient reads as

$$\begin{aligned} \frac{\partial E_G}{\partial \mathbf{u}_i} &= \sum_j a(\phi_i, \phi_j)\mathbf{u}_j \\ &+ \sum_j a(\phi_i, \psi_j)\mathbf{m}_j \\ &+ \sum_{jk} \frac{\partial \mathbf{m}_j}{\partial \mathbf{u}_i}^T a(\psi_j, \phi_k)\mathbf{u}_k \\ &+ \sum_{jk} \frac{\partial \mathbf{m}_j}{\partial \mathbf{u}_i}^T a(\psi_j, \psi_k)\mathbf{m}_k \\ &- \sum_j \frac{\partial \mathbf{m}_j}{\partial \mathbf{u}_i}^T (\mathbf{f}_e, \psi_j) - (\mathbf{f}_e, \phi_i) \end{aligned} \tag{15}$$

which is obviously nonlinear in the DOFs. As mentioned before we aim at using the implicit Euler time integration scheme, where also second order derivatives of the energy are needed [BW98] whose computation is more involved. In order to simplify the resulting system, we will therefore perform a linearization described in the following section.

## 5.2 Linearization

The complexity of the Galerkin discretization using GC results mainly from the nonlinear dependency between triangle normals and vertex positions. In order to simplify the discrete equations we make the following assumption: The change in normals between two sufficiently small timesteps is negligible such that it can be ignored during the force computation. This means that we will treat the $\mathbf{m}_i$ as constant and update them only after each timestep using the updated cage vertices.

Using this linearization of the deformation field, Eq. (15) simplifies to

$$\begin{aligned} \frac{\partial E_G}{\partial \mathbf{u}_i} &= \sum_j a(\phi_i, \phi_j)\mathbf{u}_j \\ &+ \sum_j a(\phi_i, \psi_j)\mathbf{m}_j - (\mathbf{f}_e, \phi_i), \end{aligned} \tag{16}$$

which we can rewrite as

$$\frac{\partial E_G}{\partial \hat{\mathbf{u}}} = \mathbf{K}\hat{\mathbf{u}} + \mathbf{H}\mathbf{m} - \mathbf{f}. \tag{17}$$

The additional stiffness matrix $\mathbf{H}$ relating normal differences to nodal forces consists of $3 \times 3$ blocks $\mathbf{H}_{ij} = a(\phi_i, \psi_j)$.

Comparing Eq. (17) to Eq. (7) or Eq. (8) we note that we now have additional normal-based forces $\mathbf{H}\mathbf{m}$ next to the classic vertex-based forces $\mathbf{f}$. This is the only change that needs to be made to the classic approach presented in Section 3.

## 5.3 Large Deformations

The linear strain measure in Eq. (2) is not invariant under rotations. We therefore apply a quadrature-based corotational approach as proposed in [MTPS08], for example. Additionally to the matrix $\mathbf{K}$, we also need to rotate the stiffness matrix $\mathbf{H}$. For each quadrature point $q$, the rotational part $\mathbf{R}_q$ of the local deformation gradient $\mathbf{I} + \nabla \mathbf{u}_G$ is extracted using polar decomposition. Each block $\mathbf{K}_{ij}^q$ and $\mathbf{H}_{ij}^q$ is then rotated

as described in [MTPS08], resulting in contributions $\mathbf{K} \Leftarrow \mathbf{R}_q \mathbf{K}_{ij}^q \mathbf{R}_q^T$ and $\mathbf{f} \Leftarrow (\mathbf{R}_q \mathbf{K}_{ij}^q - \mathbf{R}_q \mathbf{K}_{ij}^q \mathbf{R}_q^T)\bar{\mathbf{x}}$. The additional stiffness matrix $\mathbf{H}$ is processed analogously.

### 5.4 Boundary Conditions

Since the interpolation property of our choice of basis functions is not fulfilled on the domain boundary of the embedded mesh, a straightforward imposition of Dirichlet boundary conditions by means of fixing some DOFs to certain values is not possible. We therefore apply a penalty-based technique used for example in the context of meshless methods in computational mechanics [FM03].

In order to prescribe certain displacements $\mathbf{u}_c : \Omega_c \rightarrow \mathbb{R}^3$ for a given subdomain $\Omega_c \subset \Omega$, we add the following potential energy to $E$, penalizing the deviation from the prescribed displacements on the given subdomain:

$$E_c(\mathbf{u}) = \frac{\beta}{2} \int_{\Omega_c} |\mathbf{u} - \mathbf{u}_c|^2 \qquad (18)$$

By inserting the GC representation of the deformation into Eq. (18) and taking its derivative with respect to the DOFs, we find that the blocks $\mathbf{K}_{ij}^c = \mathbf{I} \cdot \beta \int_{\Omega_c} \phi_i \phi_j$ and $\mathbf{H}_{ij}^c = \mathbf{I} \cdot \beta \int_{\Omega_c} \phi_i \psi_j$ need to be added to the corresponding matrices and $\mathbf{f}_i^c = \beta \int_{\Omega_c} \phi_i \mathbf{u}_c$ is added to the force vector. The parameter $\beta$ steers the penalty force. In all our experiments, relatively small values in the order of $10^2$ were sufficient and did not harm the stability of the simulation. Furthermore, we chose $\Omega_c$ as a subset of the object's surface triangles and approximated the integral by a sum over the affected vertices.

**Collisions.** For simplicity, we only incorporated planar collisions within the semi-implicit integration. We perform simple nodal collision detection on the surface mesh and apply linear penalty forces to resolve the collisions. These are then discretized as external forces in Eq. (6). Due to their linear dependence on the displacement, we handle the penalty forces implicitly during time integration.

**Algortihm Overview.** Fig. 4 summarizes the actual simulation loop. The blocks $\mathbf{K}_{ij}^q$ and $\mathbf{H}_{ij}^q$ can be precomputed since they remain constant throughout the simulation.

```
1  Set K, H and f to zero
2  // Stiffness matrix and BC assembly
3  for all i, j ∈ I_V:
4    K ⇐ K_ij^c
5    f ⇐ f_i^c
6    for all quadrature points q:
7      extract rotation R^q
8      K ⇐ R^q K_ij^q R^qT
9      f ⇐ (R^q K_ij^q − R^q K_ij^q R^qT) x̄_j
10   end
11 end
12 for all i ∈ I_V, j ∈ I_T:
13   H ⇐ H_ij^c
14   for all quadrature points q:
15     f ⇐ −R^q H_ij^q (R^qT n_j s_j − n̄_j)
16   end
17 end
18 // External forces
19 compute forces according to Eq. (6)
20 // Time integration
21 implicit integration of M ∂²û/∂t² + Kû = f
22 update m with new positions of cage vertices
```

Figure 4: Summary of the simulation loop.

## 6 Results

**Comparisons.** In Fig. 5 and Fig. 1 we compare our method to linear FEM where we embed the surface geometry into a hexahedral element mesh. For the standard FEM simulations, the boundary conditions were applied in the same manner as described in Section 5.4. For the typical situation where the simulation resolution is much smaller than the surface resolution, our method clearly shows better handling of fine-scale details. Fig. 3 shows an animation sequence demonstrating the handling of boundary conditions, collisions and external forces which can also be seen in the accompanying video.

**Modeling.** The use of boundary conditions in combination with solving the static problem gives also raise to a simple and intuitive GC modeling

| Scene | # DOFs | # Q | $t_{\text{init}}$ | $t_{\text{step}}$ |
|---|---|---|---|---|
| Animation (Fig. 3) | 168 | 200 | 3132 | 529 |
| Goblin (Fig. 5) | 168 | 200 | 3236 | 503 |
| Dragon (Fig. 1) | 156 | 300 | 4615 | 653 |
| Armadillo (Fig. 6) | 159 | 300 | 4451 | 784 |

Table 1: Statistics and timings for the examples shown. We list number of DOFs, quadrature points, time to set up all $\mathbf{K}_{ij}^q$ and $\mathbf{H}_{ij}^q$ and the time to perform one timestep, in milliseconds.

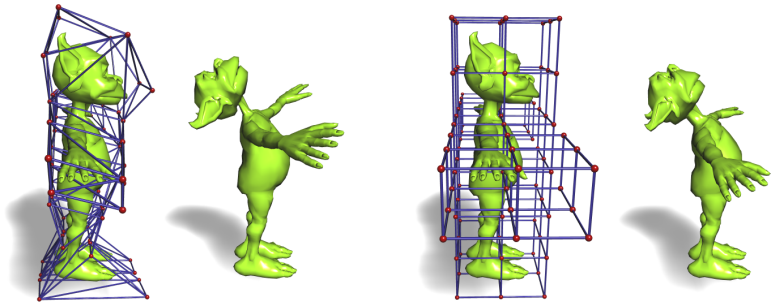

Figure 5: The goblin's head is pulled back using Dirichlet BCs. Note the preservation of the head's overall shape using our approach (left, 168 DOFs) compared to hexahedral FEM (right, 297 DOFs).

tool. In contrast to cage-based modeling, where a shape is deformed indirectly over cage manipulations, our approach allows direct modeling of the shape. Fig. 6 and the accompanying video show the armadillo model with boundary constraints at its extremities and how they can be manipulated for modeling the geometry. Since boundary conditions are imposed vertex-wise, different effects can be achieved by fixing one, two, or more vertices. Constraining only a single vertex prescribes only its position but no orientation. Constraining two vertices results in one remaining rotational degree of freedom, while fixing more than two vertices constrains position and orientation. We constrained one vertex in each hand of the armadillo and all feet vertices. Furthermore, solving the dynamics instead of the static problem allows to model animation sequences including secondary motions due to inertia as shown in the video.

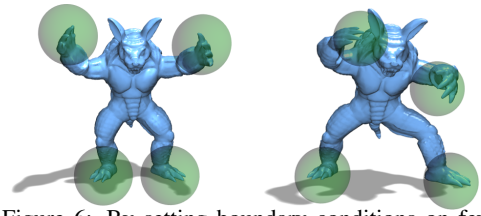

Figure 6: By setting boundary conditions on few vertices and using the static solution, our method can also be used as a modeling tool.

**Timings.** For the examples shown, Table 1 summarizes timing information, taken on an Intel Core2 Duo, 2.4 GHz. Note that the computation time is approximately linear in the number of quadrature points and quadratic in the number of DOFs due to the dense stiffness matrices.

**Limitations.** Due to the globally supported basis functions, both stiffness matrices are dense, restricting the total number of DOFs as well as the runtime performance since the dense matrix assembly needs to be performed in each timestep (see Fig. 4). While our approach works with single cages, extensions to multiple cages (i.e. multiple elements) are thinkable, leading to GC-based FEM with sparse matrices and improved performance. Assuming the normals to be constant in each timestep introduces damping in the angular momentum, especially for large timesteps. We expect that higher order approximations of the nonlinear problem can reduce this artifact.

## 7 Conclusion

We have introduced a novel approach to achieve shape-preserving deformations in physically-based animations by means of Green Coordinates. The proposed linearization of the nonlinear problem leads to a straightforward approach, just slightly modifying common FEM approaches. Next to the use as animation tool, the method also allows for intuitive GC-based geometric modeling.

## References


[BS08] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Trans. on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proc. of SIGGRAPH'98*, pages 43–54, 1998.

[FKR05] Michael S. Floater, G. Kos, and M. Reimers. Mean value coordinates in 3d. *Computer Aided Geometric Design*, 22:623–631, 2005.

[Flo03] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[FM03] T.-P. Fries and H.G. Matthies. Classification and overview of meshfree methods. Technical Report 2003-03, TU Brunswick, Germany, 2003.

[Hug00] Thomas J. R. Hughes. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.

[JBT04] Doug James, J. Barbič, and C. Twigg. Squashing cubes: Automating deformable model construction for graphics. In *Proc. of SIGGRAPH '04 Sketches and Applications*, 2004.

[JF03] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Trans. on Graphics*, 22(3):879–887, July 2003.

[JMD+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Trans. on Graphics*, 26(3):71, 2007.

[JSW05] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *Proc. of SIGGRAPH'05*, pages 561–566, 2005.

[JSWM05] T. Ju, S. Schaefer, J. Warren, and M.Desbrun. Geometric construction of coordinates for convex polyhedra using polar duals. In *Proc. of the Symp. on Geometry Processing'05*, pages 181–186, 2005.

[LLCO08] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. In *ACM Trans. on Graphics*, volume 27, pages 1–10, 2008.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proc. of Graphics Interface'04*, pages 239–246, 2004.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Trans. on Graphics*, 24(3):471–478, 2005.

[MJBF02] Tim Milliron, Robert J. Jensen, Ronen Barzel, and Adam Finkelstein. A framework for geometric warps and deformations. *ACM Trans. on Graphics*, 21(1):20–51, 2002.

[MKB+08] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross. Polyhedral finite elements using harmonic basis functions. *Computer Graphics Forum (Proc. of Symp. on Geometry Processing)*, 27(5):1521–1529, 2008.

[MLBD02] Mark Meyer, Haeyoung Lee, Alan Barr, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7:13–22, 2002.

[MTPS08] Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Interactive physically-based shape editing. In *Proc. of ACM Symp. on Solid and Physical Modeling*, pages 79–89, 2008.

[NMK+06] Andrew Nealen, Matthias Muller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH'99*, pages 137–146, 1999.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of SIGGRAPH'87*, pages 205–214, 1987.

[Wac75] Eugene L. Wachspress. *A Rational Finite Element Basis*. Academic Press, 1975.

[WBCG09] Ofir Weber, Mirela Ben-Chen, and Craig Gotsman. Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2), 2009.

[WBG07] Martin Wicke, Mario Botsch, and Markus Gross. A finite element method on convex polyhedra. In *Proc. of Eurographics'07*, pages 355–364, 2007.