Interactive Physically-Based Shape Editing

Johannes Mezger* Bernhard Thomaszewski† Simon Pabst‡ Wolfgang Straßer§
WSI/GRIS, University of Tübingen

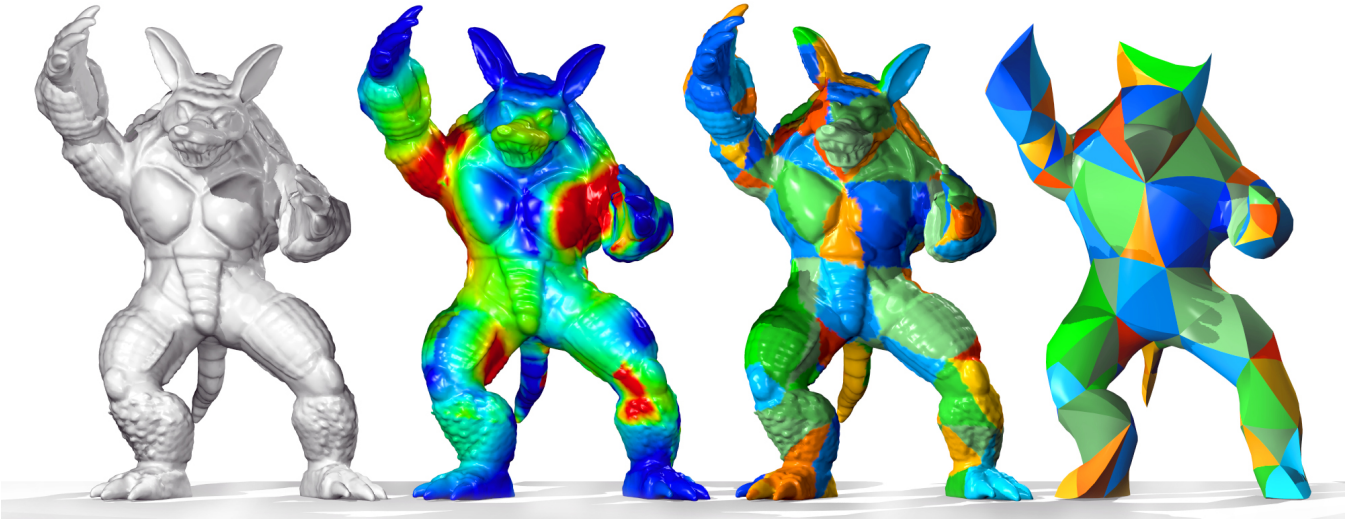

Figure 1: *Plastic deformation of a toy. On the second model the plastic strain is visualised. The two models on the right show the approximation with the quadratic finite elements.*

Abstract

We present an alternative approach to standard geometric shape editing using physically-based simulation. With our technique, the user can deform complex objects in real-time. The basis of our method is formed by a fast and accurate finite element implementation of an elasto-plastic material model, specifically designed for interactive shape manipulation. Using quadratic shape functions, we reduce approximation errors inherent to methods based on linear finite elements. The physical simulation uses a volume mesh comprised of quadratic tetrahedra, which are constructed from a coarser approximation of the detailed surface. In order to guarantee stability and real-time frame rates during the simulation, we cast the elasto-plastic problem into a linear formulation. For this purpose, we present a corotational formulation for quadratic finite elements. We demonstrate the versatility of our approach in interactive manipulation sessions and show that our animation system can be coupled with further physics-based animations like, e.g. fluids and cloth, in a bi-directional way.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—[Physically based modeling]

Keywords: mesh deformation, quadratic finite elements, plasticity

1 Introduction

With the advent of 3D data acquisition devices such as structured light scanners, highly detailed geometric surface models are now easily available. The last ten years have seen many approaches for editing such surfaces with the common goal of achieving globally smooth deformations while preserving surface details and shape volume. Most of the recent methods depart from a *purely geometric view* of this task leading to approaches which are rather detached from the actual problem. Instead of taking such a circuitous route, we propose to consider this problem as one of *elasto-plastic modelling*, which means resorting to physically-based simulation – a simple and direct approach.

For physical simulation, mass-spring systems remain the most widely used technique in the computer graphics community. Although they allow for efficient implementations, it is a well known fact that they are inherently unable to reproduce even simple isotropic materials correctly and fail to preserve volume. In contrast, (higher order) finite elements excel at these challenges. However, it is commonly believed that finite elements interchangeably stand for high computation times. In this work, we show that even a highly accurate non-linear approach can run at interactive rates.

*e-mail: mezger@gris.uni-tuebingen.de
†e-mail: thomaszewski@gris.uni-tuebingen.de
‡e-mail: pabst@gris.uni-tuebingen.de
§e-mail: strasser@gris.uni-tuebingen.de

1.1 Contributions

We present a new approach to shape editing using physically-based simulation which allows *intuitive interaction*. A dedicated *plasticity model* accounts for permanent deformations, which we think approximates the real problem of shape manipulation best. We emphasise that no artificial enforcing of *volume preservation* is needed since this property directly follows from the physical approach. Additionally, there is no need for explicitly distributing deformations induced by handles. Deformation automatically propagates through the body according to the forces applied by the user with different interaction tools. We are the first to use *quadratic finite element shape functions* in the context of shape editing. In comparison to linear elements they not only offer better numerical accuracy but also superior geometric approximation, even with a much smaller number of elements. To guarantee stability and real-time frame rates during simulation, a *corotational formulation* is combined with *implicit integration*. Due to the physical nature of our approach, the integration of shape manipulation within complex animations comes at no extra cost. As we show in our examples, the soft body simulation and manipulation can directly be combined with *fluid* or *cloth simulation*.

1.2 Related Work

Geometric Mesh Editing The problem of mesh editing can most simply be formulated as finding ways to create globally smooth and visually pleasing deformations while preserving surface details. Disturbing artefacts like surface distortion or significant change in volume have to be avoided. As a further requirement, a practically useful shape deformation algorithm has to be fast enough to deliver real-time frame rates and must offer intuitive interaction facilities [Botsch and Kobbelt 2004].

The first methods for mesh editing relied on multi-resolution representations, decomposing a model into low frequency components and detail displacements [Zorin et al. 1997; Kobbelt et al. 1998]. A more recent approach to preserving surface details under global deformations is based on differential coordinates [Alexa 2003]. In this context, detail preservation can be formulated as the minimization of an energy functional which is related to the change in differential coordinates after deformation [Sorkine et al. 2004; Yu et al. 2004]. The deformation (i.e. the editing objective) itself is incorporated as a set of positional constraints on the solution of the linear system arising from the minimization problem. Unfortunately, differential coordinates are not rotation-invariant, which means that large rotational deformations lead to disturbing surface distortions.

In the case of shape blending these rotations can be factored out locally [Alexa et al. 2000]. However, for general shape editing the problem is significantly harder since the final state is not known in advance. Pyramid coordinates [Sheffer and Kraevoy 2004] offer invariance under rigid body transformations but lead to a non-linear equation system with the associated computational and stability related problems. As an alternative, the rotation-invariant differential coordinates proposed by Lipman et al.[Lipman et al. 2005] only require the successive solution of two linear systems. A quasi-linear approach removes surface and volume distortion for large deformations [Lipman et al. 2007]. For a comparison of differential methods and their variational formulation we recommend the overview by Botsch and Sorkine [Botsch and Sorkine 2008].

Another important deformation constraint is the preservation of volume [Rappaport et al. 1996]. Combining differential coordinates with an approach for explicit volume preservation, Zhou et al. [Zhou et al. 2005] minimise both the change in surface details and shape volume. The skeleton constraint [Huang et al. 2006] is of interest when manipulating articulated shapes. Dedicated metrics in shape space are proposed by Kilian et al. [Kilian et al. 2007] to obtain isometric deformations. By introducing path line integration on divergence-free vector fields, von Funck et al. [von Funck et al. 2006; von Funck et al. 2007] obtain volume-preserving and intersection-free deformations. The adaptive volumetric discretisation by Botsch et al. [Botsch et al. 2007] is reminiscent of finite element models, but is not based on continuum mechanics and requires non-linear solvers. They show the importance of preserving the volume both globally and locally for deformations of bulky models.

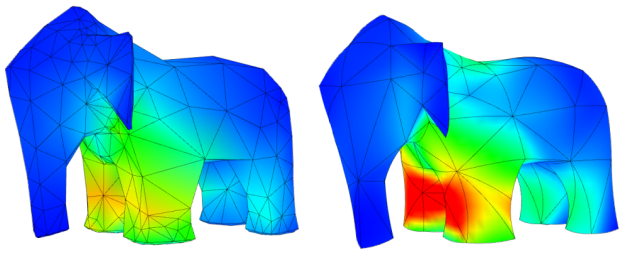

Figure 2: *Visualisation of pressure in a soft toy exposed to gravity, simulated with 1,200 linear (left) and 234 quadratic tetrahedra (right) using our new approach. Both simulations are real-time, but the linear model suffers from significant locking and bad shape approximation.*

Real-time Physically-based Simulation
Physically-based simulation of deformable objects in real-time has first been investigated in the context of general animation [James and Pai 1999]. As another important application, virtual surgery simulation poses most stringent requirements on both speed and accuracy. This has spurred the development of approaches based on continuum mechanics [Picinbono et al. 2000]. A finite element approach which uses a Bernstein-Bézier formulation focussing on highest possible accuracy rather than speed was presented by Roth et al. [Roth et al. 1998]. The simulation of clay-like materials [Dewaele and Cani 2003] or large plastic flow [Bargteil et al. 2007] can be useful for virtual sculpting rather than for shape editing where details have to be preserved during deformation. We also do not want to force the user to construct a skeleton first in order to deform the model [Capell et al. 2002]. Mesh-free finite element methods become feasible for real-time simulations if dedicated techniques like the visibility graph by Steinemann et al. [Steinemann et al. 2006] are used to speed-up the stiffness matrix and surface mesh updates. Since in our work we rather focus on the avoidance of topological changes, we stick to classic finite elements with fixed discretisation. In order to reduce the computational complexity the problem is usually recast into a linear formulation using linear finite elements and a small strain measure coupled with methods for extracting rotations [Hauth and Strasser 2004; Müller and Gross 2004]. Unfortunately, linear finite elements are very susceptible to numerical locking (see Fig. 2), which degrades accuracy substantially and lets soft objects appear overly rigid. This problem can be greatly alleviated using higher order basis functions as demonstrated in [Mezger and Straßer 2006]. The latter approach, however, uses Newton iterations to handle geometric non-linearities and can fail in finding a solution within a given period of time. Therefore, we additionally use a corotational formulation which we apply during the volume integration.

Plasticity The existing literature on mathematical and numerical plasticity is abundant and we refer the interested reader to [Zienkiewicz and Taylor 2000] and the references therein. In computer graphics, Terzopoulos et al. [Terzopoulos and Fleischer 1988] were the first to incorporate plasticity and fracture effects into deformable object simulation. O'Brien et al. [O'Brien et al. 2002] resorted to the more accurate continuum-mechanics setting, using the von-Mises yield criterion with linear plasticity and a second elastic regime which limits the plastic strain. They used an explicit integration scheme, which greatly simplifies implementation but leads to high computation times and only conditional stability. In [Müller and Gross 2004] Müller et al. account for plasticity effects using a model similar to [O'Brien et al. 2002]. However, since only tetrahedra with linear shape functions are used, curved surfaces are coarsely approximated, and nearly incompressible materials are likely to suffer from locking. For the plasticity model, we basically draw on the same idea and combine it with kinematic hardening and a prediction step for the elastic strain.

2 Background of Physical Soft Body Simulation

This section briefly outlines some mathematical and physical notions underlying our soft body simulator. For more details on solid mechanics we refer to [Zienkiewicz and Taylor 2000].

2.1 Continuum Mechanics

In the most abstract view of continuum mechanics, there are three important concepts: the *strain* $\boldsymbol{\varepsilon}$, which is a dimensionless deformation measure, the *stress* $\boldsymbol{\sigma}$, which is a force per unit area, and a *material law* relating the two to each other as $\boldsymbol{\sigma} = \mathcal{C}(\boldsymbol{\varepsilon})$, where $\mathcal{C}$ is the elasticity tensor. For the simplest case of linear isotropic elasticity, this tensor has only two independent entries which are related to the well known Lamé constants λ and μ. Alternatively, the Young modulus E and the Poisson ratio ν can be employed instead. Quantities in relation to the deformed state of the body (e.g. strain) are commonly expressed in terms of a fixed reference configuration $\Omega \subset \mathbb{R}^3$. The configuration mapping $\boldsymbol{\varphi} : \Omega \times [0, T]$ transforming material particles from their reference positions $\mathbf{x}^0$ to current positions $\mathbf{x}$ can be written as

$$\mathbf{x}(t) = \boldsymbol{\varphi}(\mathbf{x}^0, t) = \mathrm{id} + \mathbf{u}(\mathbf{x}^0, t) \,,$$

where $\mathbf{u}$ is a displacement field from the initial configuration. For later use, we define the deformation gradient $\nabla\boldsymbol{\varphi}$ and the non-linear strain tensor $\boldsymbol{\varepsilon}$ as

$$\nabla\boldsymbol{\varphi} = \frac{\partial\boldsymbol{\varphi}}{\partial\mathbf{x}^0} \quad \text{and} \quad \boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\boldsymbol{\varphi}^T\nabla\boldsymbol{\varphi} - \mathbf{I}), \quad \mathbf{I} = \mathrm{diag}(1)_{3\times 3} \,,$$

leading to the strain energy of the deformed configuration,

$$W = \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\sigma}(\mathbf{u})\, d\Omega \,.^1$$

Including viscous stress contributions with the damping tensor $\boldsymbol{\sigma}_\mathrm{v}$ and inertia with the mass density ρ, the total energy Π follows as

$$\Pi(\mathbf{u}) = \int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\sigma}(\mathbf{u}) \; + \dot{\boldsymbol{\varepsilon}}(\dot{\mathbf{u}}) : \boldsymbol{\sigma}_\mathrm{v}(\dot{\mathbf{u}})\, d\Omega \; + \tfrac{1}{2}|\ddot{\mathbf{u}}^2|\rho\, d\Omega \,.$$

Carrying out a variation of the above expression and taking into account external forces acting on the body, an equilibrium equation is obtained, which is the starting point for numerical discretisation.

[1] In this context the inner tensor product denoted by ":" reads $\boldsymbol{\varepsilon} : \boldsymbol{\sigma} = \mathrm{tr}(\boldsymbol{\varepsilon}^T\boldsymbol{\sigma})$.

2.2 Finite Element Discretisation

In a Ritz-Galerkin finite element method, the discretisation arises from a continuous partitioning of the domain into tetrahedra with locally defined shape functions $\mathbf{N}$ that interpolate the vertices $\mathbf{P}$ and their displacements $\mathbf{U}$, e.g.

$$\boldsymbol{\varphi}(\mathbf{x}) = \sum_{i=0}^{N-1} \mathbf{P}_i\mathbf{N}_i(\mathbf{x}) \quad \text{with} \sum_{i=0}^{N-1} \mathbf{N}_i = 1, \;\; \mathbf{N}_i(\mathbf{P}_j) = \delta_{ij} \,, \tag{1}$$

and

$$\nabla\boldsymbol{\varphi}(\mathbf{x}) = \nabla\mathbf{u}(\mathbf{x}) + \mathbf{I} = \mathbf{U}\,\nabla\mathbf{N}(\mathbf{x}) + \mathbf{I} \,.$$

Choosing isoparametric basis functions $\mathbf{N}$, they further interpolate force and mass densities and yield the ODE

$$\mathbf{F}(\mathbf{U}) + \mathbf{F}_\mathrm{v}(\dot{\mathbf{U}}) + \mathbf{M}\ddot{\mathbf{U}} = \mathbf{F}^\mathrm{ext} \tag{2}$$

with elastic forces $\mathbf{F}$ (Sec. 3.3), viscous forces $\mathbf{F}_\mathrm{v}(\dot{\mathbf{U}})$, dead external forces $\mathbf{F}^\mathrm{ext}$ and the mass matrix

$$\mathbf{M} = \int_V \rho\, \mathbf{N}\mathbf{N}^T\, dV \,. \tag{3}$$

The accuracy of this approximation strongly depends on the choice of the shape functions $\mathbf{N}$ and the size h of the elements. In engineering applications it is usually avoided to use linear basis functions as they achieve a convergence which is only linear in $1/h$. This weak convergence is caused by $\nabla\boldsymbol{\varphi}$ being constant and consequently also $\mathbf{F}$ being constant on the whole element.

Especially in the case of almost incompressible materials the linear elements suffer from numerical locking effects, i.e. solving (2) results in significantly smaller displacements than expected (Fig. 2). Moreover, many small elements have to be placed at the object boundaries in order to approximate irregular shapes. The approach presented in the following exploits the benefits of quadratic basis functions regarding the demands of interactive shape deformation, namely good approximation of shape and fast convergence in the presence of plastic, nearly incompressible materials.

3 Real-time Soft Body Simulation

In order to achieve real-time shape editing performance, we focus on efficiently updating the stiffness matrix which is needed for implicit time integration with arbitrarily large step sizes.

3.1 Quadratic Shape Functions

Representing the N shape functions, $4 \le N \le 10$, by using the general form

$$\mathbf{N}_i([x, y, z]^T) = \sum_{j=0}^{N-1} \alpha_{ij}\; x^{e_{j1}} y^{e_{j2}} z^{e_{j3}}, \quad i = 0..N\text{–}1,$$

with quadratic exponents $e_{jk} = 0..2$, the conditions (1) define a linear system with N^2 equations that is solved for the shape coefficients α_{ij}. This is performed once for the (unstressed) reference state of the object, storing the shape coefficients for later use (confer [Mezger and Straßer 2006]).

The number of nodes and shape functions respectively can be chosen arbitrarily from four up to ten. For any $N > 4$ additional nodes are placed on the edges of the (linear) standard tetrahedron (Fig. 3). We construct the nodal positions $\mathbf{P}^0$ of the reference state as depicted later in Sec. 5.

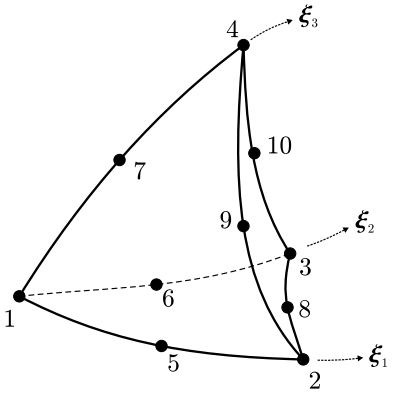

Figure 3: *Curved quadratic 10-node tetrahedron with curvilinear coordinates.*

3.2 Volume Integration

Choosing $N = 10$, each component $\mathbf{N}_i$ of the vector of shape functions is a complete quadratic polynomial and the error of this Galerkin approximation is bounded to $O(h^3)$ [Zienkiewicz and Taylor 2000]. It is crucial to preserve this quadratic convergence by accurately integrating the matrices of (2) over the volume of the tetrahedron. This is achieved by a three-dimensional quadrature (cubature) using the four-point Gauss-Legendre rule at the curvilinear coordinates

$$\boldsymbol{\zeta}_1 = [r,r,r]^T,\ \boldsymbol{\zeta}_2 = [s,r,r]^T,\ \boldsymbol{\zeta}_3 = [r,s,r]^T,\ \boldsymbol{\zeta}_4 = [r,r,s]^T$$
$$\text{with} \quad r = \tfrac{1}{4} - \tfrac{1}{20}\sqrt{5} \quad \text{and} \quad s = \tfrac{1}{4} + \tfrac{3}{20}\sqrt{5} \quad \text{[Stroud 1971]} .$$

Thus e.g. the consistent mass matrix (3) of a tetrahedron with vertices $\mathbf{P}^0$ is precomputed using

$$\mathbf{M} = \int_0^1 \int_0^{(1-\boldsymbol{\xi}_1)} \int_0^{(1-\boldsymbol{\xi}_1-\boldsymbol{\xi}_2)} \det(\mathbf{P}^0\ \nabla\hat{\mathbf{N}})\ \rho\ \hat{\mathbf{N}}\ \hat{\mathbf{N}}^T\ d\boldsymbol{\xi}_3\, d\boldsymbol{\xi}_2\, d\boldsymbol{\xi}_1$$
$$\approx \frac{1}{4\cdot 6}\sum_{i=1}^{4} \det(\mathbf{P}^0\ \nabla\hat{\mathbf{N}})\ \rho\ \hat{\mathbf{N}}(\boldsymbol{\zeta}_i)\ \hat{\mathbf{N}}(\boldsymbol{\zeta}_i)^T . \qquad (4)$$

The integration is performed in curvilinear coordinates $\boldsymbol{\xi}_i$, where the determinant accounts for the volume transformation and the factor ⅙ is the volume of the unit tetrahedron.

3.3 Corotated Quadratic Tetrahedra

The elastic forces $\mathbf{F}(\mathbf{U})$ generally depend non-linearly on $\mathbf{U}$ due to the geometric non-linearity of the strain tensor even if $\mathcal{C}$ is linear. Simply using the linear Cauchy strain tensor

$$\boldsymbol{\varepsilon}^{\mathrm{C}} = \tfrac{1}{2}(\nabla\boldsymbol{\varphi} + \nabla\boldsymbol{\varphi}^T) - \mathbf{I} \qquad (5)$$

does not produce satisfying results as soon as significant deformations occur. A corotational formulation linearises (2) by first applying element-wise rotations $\mathbf{R}$ to the displacement vector and then solving the linear system

$$\mathbf{F}(\mathbf{R}\mathbf{U})\ \mathbf{R}^T + \mathbf{F}_{\mathrm{v}}(\dot{\mathbf{U}}) + \mathbf{M}\ddot{\mathbf{U}} = \mathbf{F}^{\mathrm{ext}} .$$

Unfortunately, a single rotation matrix $\mathbf{R}$ is not enough to rotate a quadratic tetrahedron into a configuration that leaves a rotation-free deformation gradient $\mathbf{R}\nabla\boldsymbol{\varphi}$. Hence, we apply a separate polar decomposition of $\nabla\boldsymbol{\varphi}$ at each cubature point to obtain the corotated strain tensor

$$\boldsymbol{\varepsilon}^{\mathrm{CR}} = \tfrac{1}{2}(\mathbf{R}\nabla\boldsymbol{\varphi} + \nabla\boldsymbol{\varphi}^T\mathbf{R}^T) - \mathbf{I} \qquad (6)$$

and the rotation-invariant stress tensor $\boldsymbol{\sigma}^{\mathrm{CR}} = \mathcal{C}(\boldsymbol{\varepsilon}^{\mathrm{CR}})\mathbf{R}^T$. The polar decomposition is obtained by applying a QR factorisation first as described in [Hauth and Strasser 2004]. The linear elastic forces at the cubature points simply become

$$\mathbf{F}(\mathbf{R}\mathbf{U})\ \mathbf{R}^T = \nabla\mathbf{N}\ \boldsymbol{\sigma}^{\mathrm{CR}}$$

and are calculated efficiently with precomputed $\nabla\mathbf{N}$. Furthermore, because of the constant gradient $\boldsymbol{\varepsilon}^{\mathrm{CR}}_{,\mathbf{U}}$ the element stiffness matrix

$$\mathbf{F}_{,\mathrm{U}} = \frac{\partial\mathbf{F}}{\partial\mathbf{U}} = \mathbf{R}\ \mathbf{F}^{\mathrm{C}}_{,\mathbf{U}}\ \mathbf{R}^T ,$$

depends only on the current rotation $\mathbf{R}$ and a constant matrix $\mathbf{F}^{\mathrm{C}}_{,\mathbf{U}}$.

For the viscous forces no corotation is applied. Instead, simple linear damping is achieved with the time-derivative of the Cauchy strain tensor using

$$\mathbf{F}_{\mathrm{v}}(\dot{\mathbf{U}}) = \nabla\mathbf{N}\ \mathcal{D}(\dot{\boldsymbol{\varepsilon}}^{\mathrm{C}}) ,$$

since the damping is only used to remove oscillations from the material and not to model exact viscoelastic behaviour or energy dissipation.

3.4 Lazy Corotation

A significant speed-up in the solution of the ODE (2) is achieved by a "lazy" update of the rotation matrices. It is motivated by the observation that small changes of the stiffness matrix lead to imperceptible changes of the static equilibrium and that slightly deferred revaluations of the stiffness matrix are not noticeable at all in a dynamic simulation.

We roughly estimate the change of the corotation $\mathbf{R}$ of an element between the last evaluation at time t_1 and the current time t by means of the maximum absolute row sum norm

$$d(t) = \|\nabla\boldsymbol{\varphi}(t) - \nabla\boldsymbol{\varphi}(t_1)\|_\infty$$

with respect to the difference of the two deformation gradients. This expression can be evaluated efficiently, and the polar factorization is not necessary in order to determine whether the rotation changed significantly. Empirically, the stiffness matrix should be recomputed if $d(t)$ exceeds a tolerance value of 0.1 in at least one cubature point.

3.5 Guaranteed Framerate

While explicit time integration methods require adaptive time stepping to enforce stability for stiff problems, unconditionally stable implicit rules allow constant step sizes. In order to safely limit the computation time of a single large time step, we solve the corotated problem with a direct linear solver. Thus, the update rate during the modelling does not depend on the current deformation and can be kept constant. In the worst case, the simulation will require recomputing the corotations, refactoring the system matrix, and solving the linear system. With our implementation, current CPU cores perform this task within 40 milliseconds for more than 1500 linear tetrahedra or more than 300 quadratic tetrahedra, allowing a frame rate of 25Hz. The lazy corotation further serves for reducing the CPU load, but of course does not increase the lower bound of the frame rate.

4 Implicit Time Integration of Elasto-Plastic Material

The constitutive law addressed in the previous sections leads to material behaviour independent of the deformation history (also called a hyperelastic material). Once the loading is removed the deformation will (possibly delayed by viscous effects) recover a state of zero deformation. This assumption of ideal elasticity is only a rough approximation and real world materials do not obey this model. In fact, every solid material will fail, i.e. undergo irreversible deformation or even fracture, if the applied loading exceeds a certain threshold. The effect of irreversible deformation actually is the most general definition of *plasticity*, which we will use as the basic mechanism for conveying permanent shape deformation in the following.

4.1 Plasticity and Hardening

In order to extend the elastic model to account for plasticity effects we first introduce the decomposition of the total strain $\boldsymbol{\varepsilon}^{\mathrm{tot}}$ as

$$\boldsymbol{\varepsilon}^{\mathrm{tot}} = \boldsymbol{\varepsilon}^{\mathrm{el}} + \boldsymbol{\varepsilon}^{\mathrm{pl}} .$$

The total strain can be interpreted as the true geometric strain, which is readily evaluated using the finite element approximation (see Eq. (6)). As a result, the elastic stress can always be expressed as

$$\boldsymbol{\sigma} = \mathcal{C} : \boldsymbol{\varepsilon}^{\mathrm{el}} = \mathcal{C} : (\boldsymbol{\varepsilon}^{\mathrm{tot}} - \boldsymbol{\varepsilon}^{\mathrm{pl}}) . \tag{7}$$

We will generally assume that the material behaves ideally elastic up to a certain point of stress where the plastic deformation regime begins. Using a yield function F, this criterion can be expressed as $F(\boldsymbol{\sigma}) = 0$ which, depending on the current state of stress, indicates whether plastic deformation occurs or not. Similar to [O'Brien et al. 2002] we will restrict our considerations to an isotropic von-Mises yielding model, which is particularly simple. In this case F does not depend on the hydrostatic (i.e. volumetric) part of the stress tensor and, hence, plastic deformation does not affect the volume. As a consequence, e.g. twisting of a mesh will not result in unrealistic loss of volume – an important aspect which is hard to achieve with previous surface based shape editing methods.

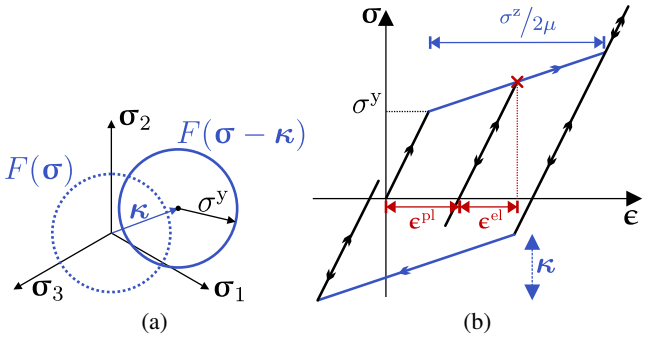

Figure 4: *With kinematic hardening the yield surface F is allowed to translate by the backstress $\boldsymbol{\kappa}$ (a). In the exemplary uni-axial loading cycle (b) the sense of traversal is indicated by arrows. Unloading is always elastic and reveals the stored plastic strain when the stress vanishes.*

The condition $F = 0$ can best be pictured as an implicit (yield) surface in stress space (circles in Fig. 4a), where the radius of the surface is a material property. Inside the yield surface the material behaves entirely elastic. Once the elastic stress reaches the yield surface, it cannot further increase and any additional deformation will result in plastic deformation. In the simplest model, the location of the surface (i.e. the centre and radius) stays fixed and as a consequence the plastic strain will increase while the elastic stress stays constant. Because this is impractical for our application, we include the effect of kinematic hardening. Here, the centre of the yield surface is allowed to move along the direction of the deviatoric strain.

With the linear kinematic hardening, a resilient plastic strain can be cancelled by a corresponding deformation in the opposite direction, leading to the stress-strain relationship depicted in Fig. 4b. The slope in the plastic phase is subject to

$$\dot{\boldsymbol{\kappa}} = H \dot{\boldsymbol{\varepsilon}}^{\mathrm{pl}}$$

with the kinematic hardening factor H. However, it would still be possible to achieve arbitrarily large plastic deformation which turned out to be inconvenient for the user. For this reason we limit the range of plastic deformation by another, user-defined threshold, producing the rightmost branch of the curve. Beyond, purely elastic behaviour is regained. The curve corresponds to a rate independent material, i.e. effects due to viscosity are not considered. Taking into account viscous stress contributions (3.3) the sharp transitions are actually smoothed according to the strain rate $\dot{\boldsymbol{\varepsilon}}^{\mathrm{el}}$.

4.2 Time Integration

With an explicit integration scheme, the time stepping of Eq. (7) is straightforward, since it requires only quantities from the current state, which are trivially known. Using implicit integration is more involved because the unknown strains $\boldsymbol{\varepsilon}^{\mathrm{tot}}(t+\Delta t)$ and $\boldsymbol{\varepsilon}^{\mathrm{pl}}(t+\Delta t)$ are required. We solve this problem using a return map algorithm similar to [Auricchio and da Veiga 2003]. Assuming that the time step will be entirely elastic, we predict the total strain $\tilde{\boldsymbol{\varepsilon}}^{\mathrm{tot}}$ at time $t+\Delta t$ in an explicit manner using the current strain rate $\boldsymbol{\varepsilon}^{\mathrm{tot}}(t)$ by

$$\tilde{\boldsymbol{\varepsilon}}^{\mathrm{tot}} = \boldsymbol{\varepsilon}^{\mathrm{tot}}(t) + \Delta t \, \dot{\boldsymbol{\varepsilon}}^{\mathrm{tot}}(t) , \tag{8}$$

where the tilde denotes trial quantities. The predicted elastic strain is computed as

$$\tilde{\boldsymbol{\varepsilon}}^{\mathrm{el}} = \tilde{\boldsymbol{\varepsilon}}^{\mathrm{tot}} - \boldsymbol{\varepsilon}^{\mathrm{pl}}(t) ,$$

and the deviatoric strain follows as

$$\tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} = \tilde{\boldsymbol{\varepsilon}}^{\mathrm{el}} - \tfrac{1}{3} \operatorname{tr}(\tilde{\boldsymbol{\varepsilon}}^{\mathrm{el}}) \, \mathbf{I} .$$

If we further assume that the plastic strain $\boldsymbol{\varepsilon}^{\mathrm{pl}}$ remains constant we can evaluate the yield function,

$$F(\mathcal{C} : \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}}) = ||2\mu \, \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} - \boldsymbol{\kappa}(t)|| - \sigma^{\mathrm{y}} = 0 ,$$

where σ^{y} is the yield stress (cf. Fig. 4). If the yield function signals that in the next step there will be no transition to the plastic range, the assumption holds and we can safely use the standard implicit formulation to integrate the elastic forces. Otherwise, the plastic strain for the end of the time step is computed as

$$\boldsymbol{\varepsilon}^{\mathrm{pl}}(t+\Delta t) = \boldsymbol{\varepsilon}^{\mathrm{pl}}(t) + \lambda \frac{2\mu \, \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} - \boldsymbol{\kappa}(t)}{||2\mu \, \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} - \boldsymbol{\kappa}(t)||}$$

The consistency parameter λ ensures that the yield condition is met after the time step and reads

$$\lambda = \frac{||2\mu \, \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} - \boldsymbol{\kappa}(t)|| - \sigma^{\mathrm{y}}}{2\mu \, \tilde{\boldsymbol{\varepsilon}}^{\mathrm{dev}} + H} .$$

To limit the norm of the plastic stress components by σ^z, the plastic strain is clamped to $\sigma^z/2\mu$ (cf. O'Brien et al. [O'Brien et al. 2002]). Finally, we have to compute the new backstress using

$$\boldsymbol{\kappa}(t+\Delta t) = H \cdot \left(\boldsymbol{\varepsilon}^{\mathrm{pl}}(t+\Delta t) - \boldsymbol{\varepsilon}^{\mathrm{pl}}(t)\right) \ .$$

Similar to Müller et al. [Müller and Gross 2004] we obtain a linear system of equations which, for the sake of simplicity, we abbreviate as

$$\left(\mathbf{M} - \mathbf{R}\,\nabla\boldsymbol{\mathcal{F}}\,\mathbf{R}^T\right)\mathbf{Y} = \mathbf{F}^{\mathrm{const}} \ ,$$

where the state vector $\mathbf{Y}$ is the concatenation of nodal positions and velocities. Because of the prediction step (8) the plastic strain does not depend on $\mathbf{Y}$ and can be considered as a dead load during the time step. Hence, we are able to keep the stiffness matrix $\nabla\boldsymbol{\mathcal{F}}$ constant over time. Note that in contrast to explicit time integration schemes the mass matrix $\mathbf{M}$ does not have to be inverted and we can use the non-diagonal consistent mass matrix (4) without computational drawbacks.

4.3 Annealing

In the course of repeated deformation and sculpting by the user, substantial plastic strains can accumulate. While the simulation always remains stable, very large deformations are likely to degrade computational efficiency and accuracy. This problem can be avoided by *annealing* the solid from time to time. When the user stops deforming the object for a moment, its rest state is recomputed from the current deformation and the geometric as well as the plastic strains are reset. This procedure can be carried out as a background operation without the user taking notice of it, or, as shown in the accompanying video, by manually clicking a button to "commit" the current plastic state and to continue with further manipulations.

5 Geometric Model Reduction and Detail Preservation

In order to provide the user with as much freedom as possible we do not make specific assumptions on the size and resolution of the input model. We do, however, assume a closed manifold surface mesh of the object to be deformed. Such meshes can be obtained, e.g. using a geometry acquisition device like a structured light scanner and a subsequent post-processing step (i.e. reconstruction and triangulation). The resolution of meshes obtained in this way is usually very high. Since globally smooth deformations can be captured on a much coarser level it is common practice to treat high frequency surface details apart from possibly large low frequency deformations.

5.1 Geometric Model Reduction

We adopt this strategy and combine geometric model reduction techniques with an elegant detail preservation algorithm which arises in a natural way from our finite element approach. Low resolution tetrahedral meshes like the ones in Fig. 5 (bottom row) are created following a three-step algorithm:

Coarse Surface Generation For the initial mesh simplification we employ standard triangle mesh reduction techniques available for polygonal modelling software (e.g. MeshLab), producing approximate Delaunay triangulations (Fig. 5 top row).

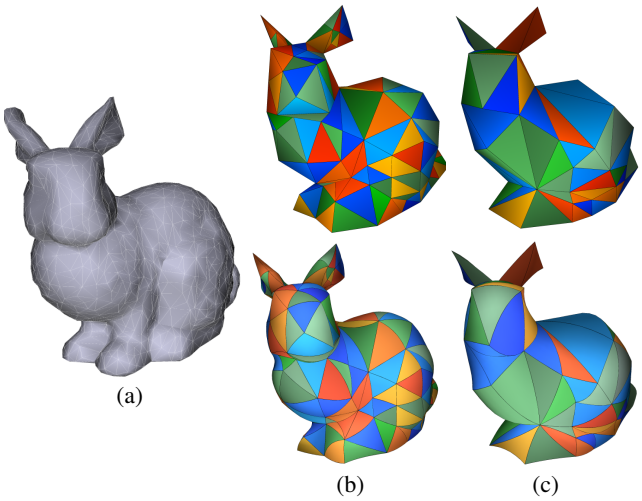

Figure 5: *Automatic generation of quadratic tetrahedral meshes in two resolutions (b, c) from a detailed triangle mesh (a). The top row shows the result of the surface mesh simplification, the bottom row the boundary of the conformed quadratic FE meshes.*

Linear Tetrahedral Meshing The Delaunay property alleviates the subsequent generation of tetrahedral volume meshes, which is achieved using standard mesh generators. In our experience they often fail in generating coarse quadratic meshes for a given smooth surface. For this reason, a standard meshing first produces an FE model with linear 4-node tetrahedra.

Quadratic Surface Conforming Afterwards, each tetrahedron of the coarse tetrahedral mesh is completed by the missing nodes on the six edges. All surface nodes are adjusted to lie on the initial high resolution surface (Fig. 6a). For this purpose, the closest surface face in the normal direction is found. If its distance exceeds a specific limit or if the surfaces turn out to be too spiky to determine a consistent normal direction, the surrounding area is searched for a closer face (Fig. 6b). Care is taken not to invert surface elements if the detailed surface lies inwards (Fig. 6c). In such rare cases the node must not be moved and it is advisable to increase the FE mesh resolution in the critical region.

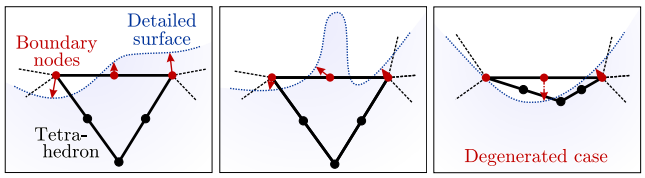

Figure 6: *In the conforming step the surface nodes of the quadratic tetrahedral mesh are adjusted to lie on the detailed surface. (Simplified 2D drawing.)*

While the whole reduction algorithm is rather simple, it turned out to be extremely effective and does not demand any FE modelling knowledge from the user. Curved element edges are preliminarily created at the FE mesh boundaries, but do of course also emerge in the inside during deformations. Hence, the full degrees of freedom of the quadratic tetrahedra are exploited by the simulation.

5.2 Detail Preservation

Since the plastic deformations are computed on the coarse FE mesh, the simulation does not affect surface details. To interpolate detailed surface features, we map the initial surface points by the isoparametric shape functions of the associated deformed tetrahedron, which provides the necessary smoothness and affine invariance. That is, at any time t, by

$$\mathbf{S}^t = \boldsymbol{\varphi}(\mathbf{S}^0, t) = \sum_{i=0}^{N-1} \mathbf{P}_i^t \, \mathbf{N}_i(\mathbf{S}^0) \, .$$

the interpolated detailed surface point $\mathbf{S}^t$ is calculated from the initial position $\mathbf{S}^0$ of the vertex. The shape functions $\mathbf{N}_i$ interpolate the nodes $\mathbf{P}^t$ of the tetrahedron as defined in (1). Cracks at the transitions from one quadratic tetrahedron to the next are avoided because the FE surface nodes lie on the detailed surface which is further interpolated smoothly by the curved edges. During annealing the rest positions $\mathbf{S}^0$ are reset to the current coordinates.

The colours $c(\mathbf{x}, t)$ for stress visualisations on the detailed surface first are linearly extrapolated from the cubature points to the nodes $\mathbf{P}_i^0$ of the reference state. Afterwards, the colours $c(\mathbf{S}^t, t)$ for the detailed surface are mapped again by the precomputed shape functions using

$$c(\mathbf{S}^t, t) = \sum_{i=0}^{N-1} c(\mathbf{P}_i^0, t) \, \mathbf{N}_i(\mathbf{S}^0) \, .$$

This is an intuitive and computationally efficient way of stress visualisation, which, to our knowledge, was not addressed in literature so far.

6 Results

The described techniques were applied successfully to perform several shape editing tasks on a Dual Xeon 5140, where only one core was used by our implementation. An implicit second order BDF (backward differentiation formula) solver ensured stable time integration. Reasonably stiff materials ($\lambda > 5\text{kPa}$) and a moderate kinematic hardening ($H \approx 0.1\mu$) produced a precise perception of the resistance to deformations. The choice of the Poisson ratio ν, which controls the compressibility and hence the volume preservation of the material, did not have a visually significant influence on the results. Using $\nu = 0.4$ similar to clay was sufficient for all editing examples.[2]

6.1 Examples

The examples were created in interactive sessions with a time step size of 40 milliseconds. Only the simulation of the first example, the twisting and bending of a bar, was scripted to allow for comparison with linear element shape functions. The accompanying video demonstrates the feasibility of individual shape manipulation tasks ranging from large-scale deformations to tweaks of fine features.

Twisting and Bending In order to provide a better comparison to previous work we use a standard example to show the advances of our method. A bar with a square profile is exposed to heavy twisting (Fig. 7) and bending (Fig. 8). Due to the regularity of the undeformed shape, artefacts of the deformation process become clearly visible in the result. The simulation is performed first using 2,212 linear tetrahedra and second using 298 tetrahedra with quadratic shape functions. In general, a computation with quadratic tetrahedra takes about 8 times longer than the computation with the same number of linear tetrahedra. Hence, in this case both simulations approximately have the same running time.

[2] $\nu = 0.5$ would denote incompressibility. Real materials have $-1 < \nu < 0.5$.

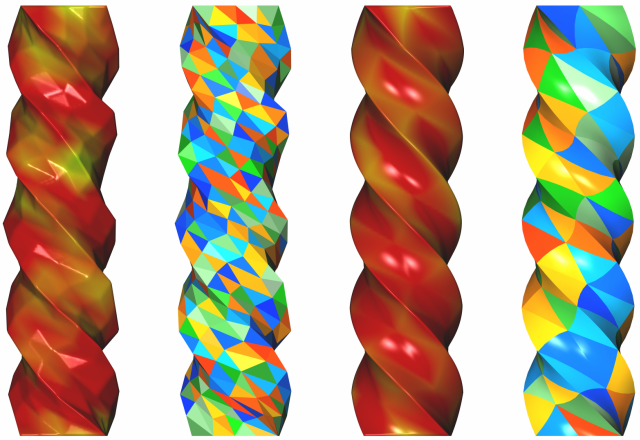

Figure 7: *Twisting of a bar with 19,802 surface vertices, simulated with 2,212 linear tetrahedra (left) and 298 quadratic ones (right). While the computation times are almost identical, the quality of the quadratic simulation is noticeably better.*

However, the linear tetrahedra fail completely in representing both the twisted and the bent bar. In Fig. 7 the left picture shows the surface mesh (19,802 vertices) of the twisted bar with visible artefacts from the underlying, distorted linear tetrahedra. The colours visualise the volumetric pressure at the surface, red denoting maximal values. The second picture reveals that the FEM discretisation with the linear tetrahedra is not able to sufficiently approximate the solution of the PDE. The third and the fourth rendering show the respective results with quadratic tetrahedra, which achieve significantly lower distortions and a much better approximation.

For the bending (Fig. 8) we added two more pictures which show the surface mesh as viewed from the bottom. Blue colours denote zero pressure, i.e. regions without volume change. Since the position of the caps of the bar is preset, the locking of the linear tetrahedra does not result in reduced deformation, but causes distortions where the material bulges. The renderings use orthographic projections to show the effects of volume preservation.

As long as the Poisson ratio ν is greater than 0.2, it does not have a significant influence on the final shape. Pictured are the results for $E = 10\text{kPa}$ and $\nu = 0.49$, which gives a volume loss of 11.6% (linear tetrahedra) and 9.0% (quadratic) for the extreme twisting, and 4.0% and 3.0% respectively for the bending.

Editing of High-Resolution Meshes For interactive shape editing, our modelling interface lets the user mark regions on the surface mesh which should be fixed, and others which should follow the mouse pointer by applying surface forces depending on the distance from the pointer. This technique allows an intuitive modelling without the need to assign handles to the regions. The forces are distributed equally to the selected region, and the surface force vectors point to the same direction. Hence, within the selected region the surface details only change in response to internal elastic forces, e.g. if the material is being bulged or stretched.

The ability of the real-time simulation to preserve volume and to correctly handle situations of extremely large deformation can be observed in Fig. 9. The surface meshes are coloured based on the

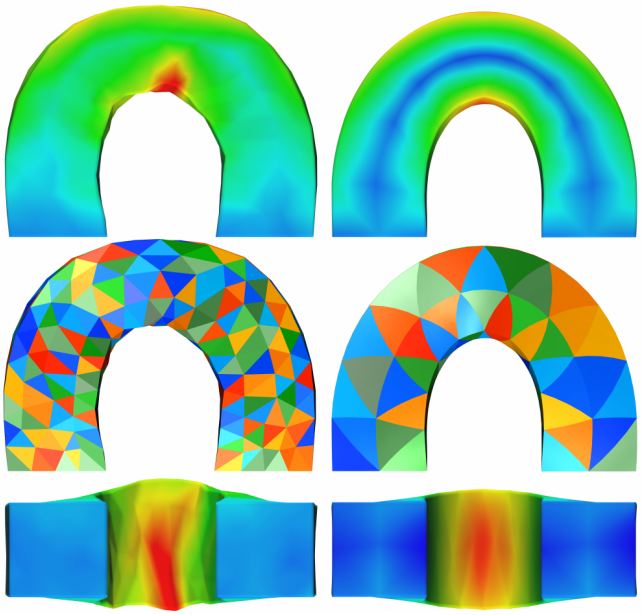

Figure 8: *Bending of the bar from Fig. 7, again with linear tetrahedra on the left and quadratic ones on the right. The surface mesh is shown at the top and the FE surface in the middle. Another perspective is used at the bottom to show the bulges emerging from the volume conservation.*

norm of the plastic strain, red denoting that the plastic stress component is close to the limit σ^z. To the right hand side the surface of the tetrahedral meshes is visualised. Fig. 1 demonstrates that a complex surface (60,000 vertices) with rich features is interpolated smoothly by the quadratic basis functions (229 tetrahedra). The mesh conforming is capable of creating quadratic tetrahedral meshes within a wide range of resolutions (Fig. 10). While the lower resolutions are dedicated for the fast modelling of global deformations, the higher resolutions provide still interactive editing of the global as well as the local shape. Even with comparably few finite elements, a high quality surface interpolation and interactive, volume preserving deformations are obtained (Fig. 11).

A drawback of our method is that excessive deformations could cause some tetrahedra to become inverted and that they usually cannot recover from this state due to the corotational formulation. However, in this rare case the simulation still stays stable since the combination of QR and polar decomposition always produces a valid corotation. An adoption of the diagonalisation used by Irving et al. [Irving et al. 2004] could offer additional potential to handle the problem, which we leave as an option for future work.

Integration with Other Simulators A bi-directional integration with other physics based animation techniques, namely a grid-based fluid simulation and a cloth simulation based on finite elements (Fig. 12), is possible in a straightforward manner. This provides the animator with the opportunity to model complex environments which would be impossible to animate in the traditional way. We employ a simple method which treats each simulator as a black box and couples them by updating the boundary conditions in every time step. Wind forces from the fluid simulation are applied to surface triangles in the vicinity of the grid node. Vice versa, the particles are reflected at the surface. Repulsion forces between deformable solids and clothes are transferred in response to detected proximities or intersections of the surfaces. Both techniques work

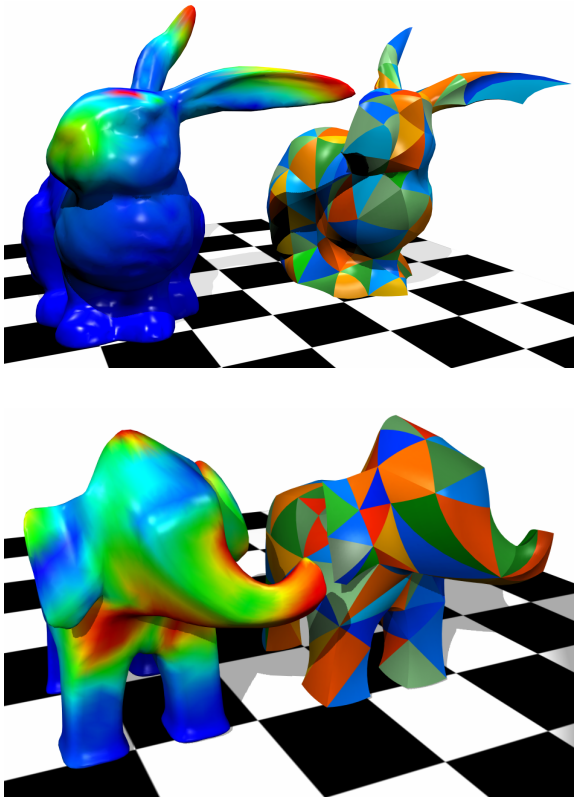

Figure 9: *Physically correct shape editing with at least 40 fps. Bunny: 1,379 surface vertices / 101 quadratic tetrahedra. Elephant: 2,578 / 234, respectively.*

sufficiently well for animation purposes, but of course would be less suited for applications where e.g. momentum transfer and energy conservation are important.

6.2 Benchmarks

Table 1 shows the computation times for two exemplary deformation tasks: repeatedly dragging one arm of the armadillo (Fig. 1) up and down, and stretching and bending the upper part of the dragon model (Fig. 10). For the armadillo model, surface meshes with 20,000 and 60,000 vertices, and volume meshes with 146 and 1,051 quadratic tetrahedra were tested. The dragon model is discretised with up to 1,005 quadratic tetrahedra and 100,000 vertices. Computation times per frame (not considering lazy corotation) are separated into corotation with stiffness matrix update (T_{mat}), matrix re-factorisation (T_{fac}), solution of the linear system (T_{solve}) and update time of the interpolated surface mesh (T_{def}). From the total time (T_{tot}) it is evident that time-step sizes from 146ms for the largest model down to 19ms for the smallest model can be used to achieve real-time. Additionally, in more than half of the frames lazy corotation was active, leaving only T_{solve} and T_{def} for the respective frames, and therefore significantly reducing the average computation time per frame. Actually, in our experiments the frame rate was limited mostly by the modelling environment, which acted as a bottleneck when huge surface meshes had to be updated for rendering.

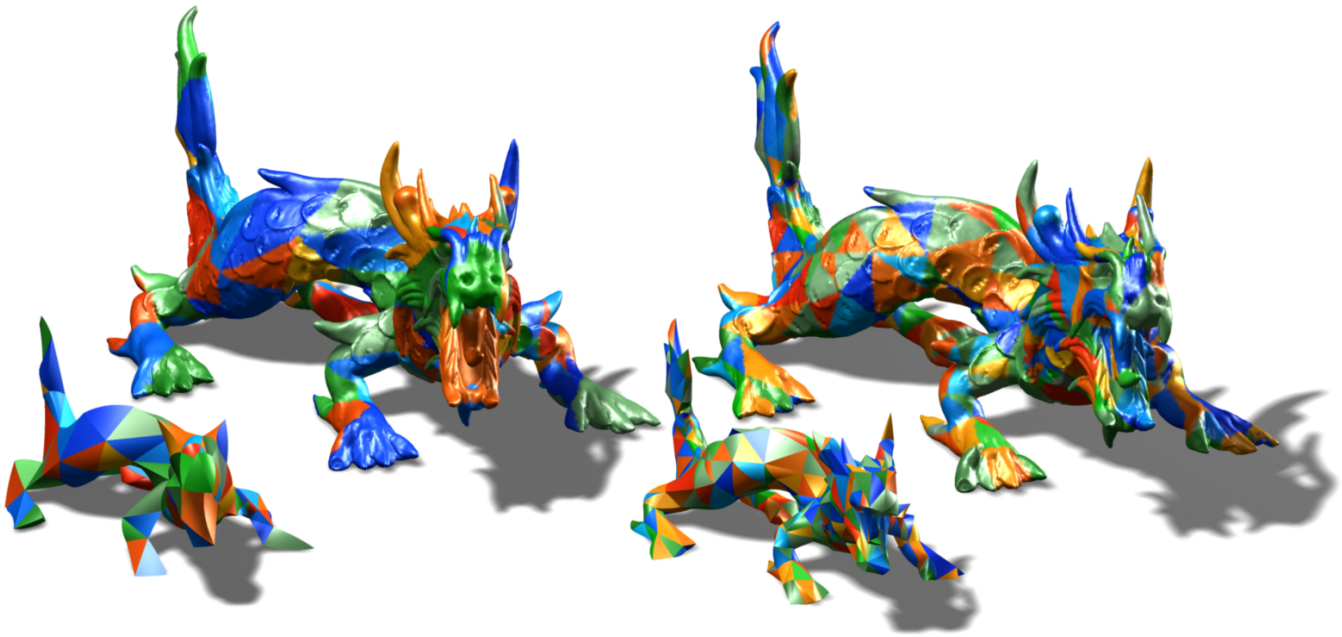

Figure 10: *Dragon: 100,000 surface vertices with 173 quadratic tetrahedra (left) and 1,005 quadratic tetrahedra (right). The surface colours visualise the mapping of the boundary tetrahedra to the surface points.*

Surface mesh	Tetra-hedra	T_{mat}	T_{fac}	T_{solve}	T_{def}	T_{tot}
Arm 20k	146	10	3.9	2.5	1.9	18
Arm 60k	”	”	”	”	5.3	22
Arm 20k	1,051	73	43	23	2.0	141
Arm 60k	”	”	”	”	5.4	146
Dragon 20k	173	13	3.3	2.7	1.9	21
Dragon 100k	”	”	”	”	8.7	28
Dragon 20k	1,005	74	29	15	2.3	120
Dragon 100k	”	”	”	”	8.8	127

Table 1: *Computation times in milliseconds for one implicit time-step with the armadillo and the dragon model in different resolutions up to 100,000 vertices. The surface mesh resolution only marginally affects the computation time (not considering the rendering).*

6.3 Conclusions and Future Work

We introduced a new approach to shape editing, which is applicable to complex surface meshes, but still takes the elastic properties of volumetric bodies into account. The main advantages of our approach are intuitive deformations and the potential to easily combine plastic shape modelling with other simulations or animations. From the continuum mechanics formulation we automatically obtain physically accurate results. The quadratic shape functions allow for smooth mesh interpolation even with a small number of tetrahedra. Together with a dedicated linearisation this provides the necessary performance for interactive modelling.

Although our approach does not directly support skeleton driven deformations, similar effects can be achieved by changing local material properties in an appropriate way: assigning a stiff material to the limbs and a comparably soft one to the joints yields the desired behaviour. However, in future work we hope to extend our system to account for totally rigid regions and with a more convenient user interface for specifying skeleton constraints like in the work of [Zhou et al. 2005].

Acknowledgements

The third author was supported by DFG grant STR 465/21-1.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-Rigid-As-Possible Shape Interpolation. In *Proc. ACM SIGGRAPH*, 157–164.

ALEXA, M. 2003. Differential coordinates for mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

AURICCHIO, F., AND DA VEIGA, L. B. 2003. On a new integration scheme for von-Mises plasticity with linear hardening. *International Journal for Numerical Methods in Engineering 56*, 10, 1375–1396.

BARGTEIL, A. W., WOJTÁN, C., HODGINS, J. K., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3, article no. 16.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOTSCH, M., AND SORKINE, O. 2008. On Linear Variational Surface Deformation Methods. *IEEE Trans. Vis. Graph. 14*, 1, 213–230.

BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. 2007. Adaptive Space Deformations Based on Rigid Cells. In *Proc. Eurographics*.

Figure 11: *Offline rendering of the interactively edited dragon with 173 quadratic tetrahedra and 250,000 surface vertices. Neither artefacts nor loss of volume are observed in the bent area.*

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

DEWAELE, G., AND CANI, M.-P. 2003. Interactive Global and Local Deformations for Virtual Clay. In *Proc. Pacific Graphics*.

HAUTH, M., AND STRASSER, W. 2004. Corotational Simulation of Deformable Solids. In *Proc. WSCG*, 137–145.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3, 1126–1134.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. ACM Symp. Comp. Animation (SCA)*, 131–140.

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

KILIAN, M., MITRA, N. J., AND POTTMANN, H. 2007. Geometric Modeling in Shape Space. *ACM Trans. Graph. 26*, 3, article no. 64.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive Multi-Resolution Modeling on Arbitrary Meshes. *Computer Graphics (Proc. ACM SIGGRAPH) 32*, 105–114.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

LIPMAN, Y., COHEN-OR, D., GAL, R., AND LEVIN, D. 2007. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph. 26*, 1, article no. 5.

MEZGER, J., AND STRASSER, W. 2006. Interactive Soft Object Simulation with Quadratic Finite Elements. In *Articulated Motion and Deformable Objects (AMDO)*, vol. 4069 of *LNCS*, 434–443.

MÜLLER, M., AND GROSS, M. 2004. Interactive Virtual Materials. In *Proc. Graphics Interface*, 239–246.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 3, 291–294.

PICINBONO, G., DELINGETTE, H., AND AYACHE, N. 2000. Real-Time Large Displacement Elasticity for Surgery Simulation: Non-linear Tensor-Mass Model. In *MICCAI*, 643–652.

RAPPAPORT, A., SHEFFER, A., AND BERCOVIER, M. 1996. Volume-preserving free-form solids. *IEEE Trans. Vis. Comp. Graph. 2*, 1, 19–27.

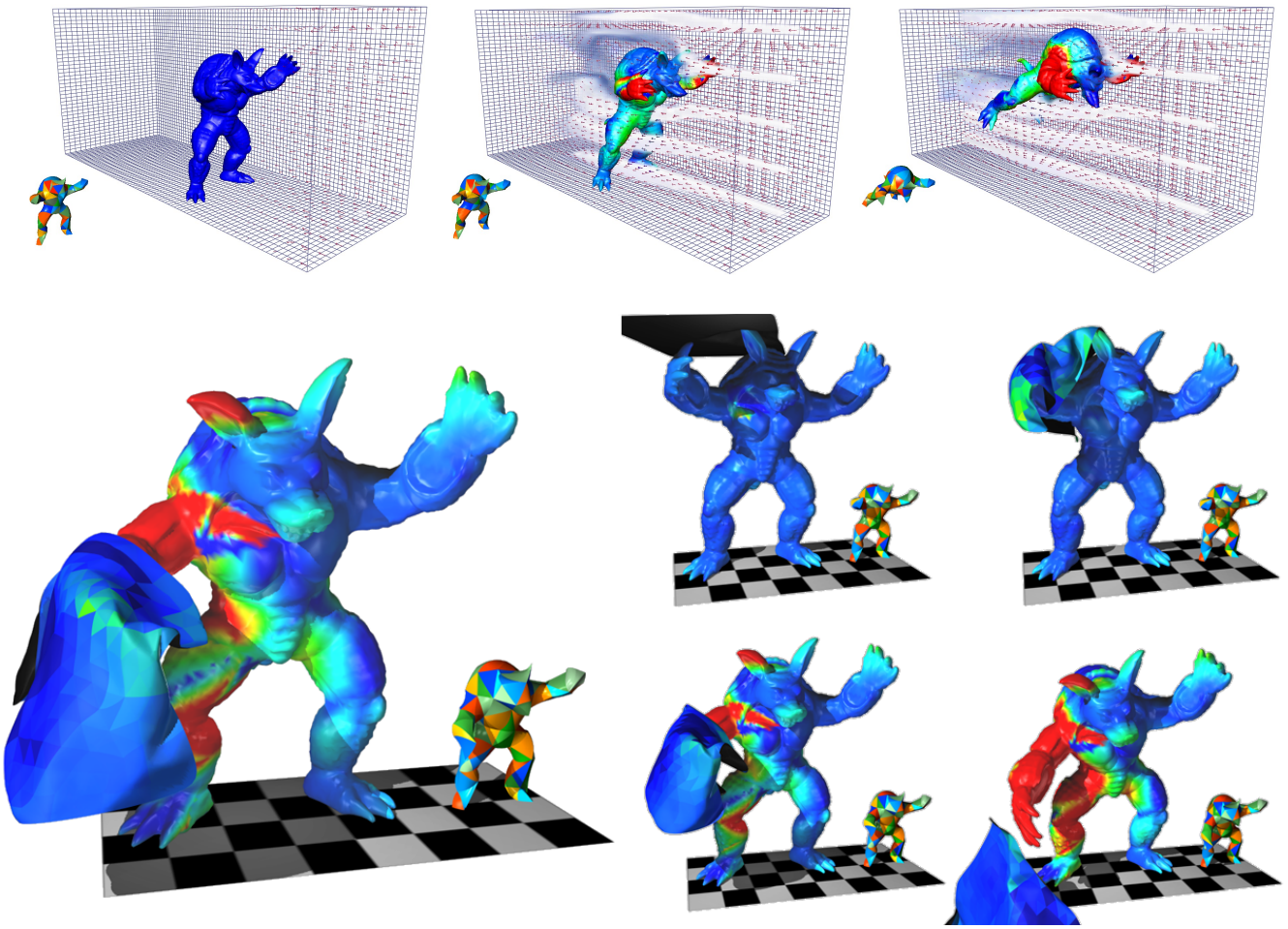

Figure 12: *Armadillo: 16,608 surface vertices, 146 quadratic tetrahedra. Interaction with wind (top), draped with an FE simulated piece of cloth (bottom).*

ROTH, S. H. M., GROSS, M. H., TURELLO, S., AND CARLS, F. R. 1998. A Bernstein-Bézier Based Approach to Soft Tissue Simulation. *CGF 17*, 3, 285–294.

SHEFFER, A., AND KRAEVOY, V. 2004. Pyramid Coordinates for Morphing and Deformation. In *Proc. 3D Data Processing, Visualization, and Transmission*, 68–75.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., ROSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Symp. Geometry processing (SGP)*, 179–188.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *ACM Symp. Comp. Animation (SCA)*, 63–72.

STROUD, A. H. 1971. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, Inc.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. *Computer Graphics (Proc. ACM SIGGRAPH) 22*, 4, 269–278.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3, 1118–1125.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2007. Explicit Control of Vector Field Based Shape Deformations. In *Proc. of Pacific Graphics*, 291–300.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3, 644–651.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph. 24*, 3, 496–503.

ZIENKIEWICZ, O. C., AND TAYLOR, R. L. 2000. *The Finite Element Method*, fifth ed., vol. 1 and 2. Butterworth-Heinemann.

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proc. ACM SIGGRAPH*, 256–268.