Meshless Deformations Based on Shape Matching

Matthias Müller
NovodeX/AGEIA & ETH Zürich

Bruno Heidelberger
ETH Zürich

Matthias Teschner
University of Freiburg

Markus Gross
ETH Zürich

Figure 1: The presented technique is stable under all circumstances and allows to simulate hundreds of deformable objects in real-time.

Abstract

We present a new approach for simulating deformable objects. The underlying model is geometrically motivated. It handles point-based objects and does not need connectivity information. The approach does not require any pre-processing, is simple to compute, and provides unconditionally stable dynamic simulations.

The main idea of our deformable model is to replace energies by geometric constraints and forces by distances of current positions to goal positions. These goal positions are determined via a generalized shape matching of an undeformed rest state with the current deformed state of the point cloud. Since points are always drawn towards well-defined locations, the overshooting problem of explicit integration schemes is eliminated. The versatility of the approach in terms of object representations that can be handled, the efficiency in terms of memory and computational complexity, and the unconditional stability of the dynamic simulation make the approach particularly interesting for games.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

Keywords: deformable modeling, geometric deformation, shape matching, real-time simulation

1 Introduction

Since Terzopoulos' pioneering work on simulating deformable objects in the context of computer graphics [Terzopoulos et al. 1987], many deformable models have been proposed. In general, these approaches focus on an accurate material representation, on stability aspects of the dynamic simulation and on versatility in terms of advanced object characteristics that can be handled, e. g. plastic deformation or fracturing.

Despite the long history of deformable modeling in computer graphics, research results have rarely been applied in computer games. Nowadays, deformable cloth models with simple geometries can be found in a few games, but in general, games are dominated by rigid bodies. Although rigid bodies can be linked with joints to represent articulated structures, there exist no practical solution which allows to simulate elastically deformable three-dimensional objects in a stable and efficient way. There are several reasons that prevent current deformable models from being used in interactive applications.

Efficiency. Existing deformable models based on complex material laws in conjunction with stable, implicit integration schemes are computationally expensive. Such approaches do not allow for interactive simulations of objects with a reasonable geometrical complexity. Further, these approaches might require a specific object representation and the algorithms can be hard to implement and debug. In contrast, interactive applications such as games constitute hard constraints on the computational efficiency of a deformable modeling approach. The approach is only allowed to use a small fraction of the available computing resources. Further, specific volumetric representations of deformable objects are often not available since the geometries are typically represented by surfaces only.

Stability. In interactive applications, the simulation of deformable objects needs to remain stable under all circumstances. While sophisticated approaches allow for stable numerical integration of velocities and positions, additional error sources such as degenerated geometries, physically incorrect states, or problematic situations with large object interpenetrations are not addressed by many approaches. A first contribution to this research area has been presented in [Irving et al. 2004], where large deformations and the inversion of elements in FE approaches can be handled in a robust way. However, this approach is not intended to be used in interactive applications.

Controllability. Physically-based deformable models are intended to simulate the dynamic object behavior as realistically as possible. However, in games or entertainment technologies, the simulation, i. e. the magnitude and shape of a deformation need to be controllable by the developer, tolerating a degradation of realism as long as the result looks realistic. An early discussion of physically-plausible simulations compared to accurate approaches can be found in [Barzel et al. 1996] and [Barzel 1997].

Many available techniques for simulating deformable objects fail with respect to one or more of these aspects. To overcome the existing restrictions we propose a technique which addresses the aforementioned problems and contributes towards stable, interactive, and versatile deformable modeling. In particular, the contributions of our deformable modeling approach are:

- Elasticity is modeled by pulling a deformed geometry towards a well-defined goal configuration which is determined by an extended shape matching technique.
- The degree of representable deformation details can be varied using linear and quadratic deformation modes. Subdivisions into clusters introduce further degrees of freedom.
- A large variety of objects can be handled. Geometric deformations are just computed for points and connectivity information or specific representations are not required.
- Our technique does not require any pre-processing or large accompanying data structures. The configuration of parameters is simple and intuitive. Thus, the technique is as simple as "plug and simulate" in terms of object handling.
- The dynamic simulation is stable under all circumstances and for all deformed geometry configurations. The approach is not exposed to problems such as ill-shaped or inverted elements. Even non-manifold meshes with arbitrarily shaped triangles can be handled.
- All components of the approach are simple to implement and very efficient in terms of memory requirements and run-time performance.

2 Related Work

Many methods and models have been proposed in computer graphics to simulate deformable objects ranging from finite difference approaches [Terzopoulos et al. 1987], mass-spring systems [Baraff and Witkin 1998], [Desbrun et al. 1999], the Boundary Element Method (BEM) [James and Pai 1999], the Finite Element Method (FEM) [Debunne et al. 2001], [Müller et al. 2002], [Müller and Gross 2004], the Finite Volume Method (FVM) [Teran et al. 2003] to implicit surfaces [Desbrun and Cani 1995] and mesh-free particle systems [Desbrun and Cani 1996], [Tonnesen 1998], [Müller et al. 2004].

In addition to approaches which mainly focus on the accurate simulation of elasto-mechanical properties, there exist several acceleration strategies. Robust integration schemes for large time steps have been investigated [Baraff and Witkin 1998] and multi-resolution models have been proposed [Debunne et al. 2001], [Capell et al. 2002], [Grinspun et al. 2002]. To further improve the performance, modal analysis approaches have been employed which can trade accuracy for efficiency [Pentland and Williams 1989], [Shen et al. 2002], [James and Pai 2002]. Further, data-driven methods have been presented where the pre-computed state space dynamics and pre-computed impulse response functions are incorporated to improve the run-time performance [James and Pai 1999]. In [Metaxas and Terzopoulos 1992], dynamic models have been derived from global geometric deformations of solid primitives such as spheres, cylinders, cones, or superquadrics.

Our approach uses deformation modes which are related to modal analysis approaches [Pentland and Williams 1989], [Hauser et al. 2003], [James and Pai 2004]. However, we do not approximate the deformation modes from elasto-mechanical object properties which commonly requires the incorporation of additional auxiliary object representations, such as tetrahedral meshes [James and Pai 2002]. Instead, our deformation modes are geometrically motivated which significantly simplifies the pre-processing stage. Although the model is not physically motivated, we show that the approach has similar capabilities in representing deformations compared to existing modal analysis models. In contrast to existing models, no additional data structures for computing the deformation modes, no expensive modal decompositions, and no explicit representation and storage of modal vectors are required in our technique. Thus, our approach is efficient in terms of computational complexity and memory.

Our approach draws from previous work in the field of shape matching [Shoemake and Duff 1992], [Alexa et al. 2000], [Kent et al. 1992]. While standard shape matching approaches are primarily concerned with establishing the correct correspondences between two shape representations [Faugeras and Hebert 1983], [Horn 1987], [Besl and McKay 1992], [Kazhdan et al. 2004], all correspondences are a priori known in our case. The remaining problem is that of finding least squares optimal rigid transformations in 3D between the two point clouds with a priori known correspondences [Kanatani 1994], [Umeyama 1991] and [Lorusso et al. 1995]. In contrast to finding optimal rigid transformations, we extend these methods to compute optimal linear and quadratic transformations.

3 Meshless Animation

Newton's second law of motion is the common basis for many physically-based simulation techniques including rigid body simulation, deformable modeling, and fluid simulation. Deviations from an equilibrium cause forces which accelerate the material back to an equilibrium configuration. In order to compute object locations, the accelerations and velocities are numerically integrated over time.

Stability and efficiency are major issues in numerical integration. Implicit integration schemes guarantee stability independent of the chosen time step. However, they require the solution of a system of equations which is computationally expensive and potentially interferes with the constraints of interactive applications. In contrast, explicit integration schemes are much faster to compute. Unfortunately, this advantage comes with the loss of unconditional stability. A prominent example is the explicit Euler scheme which is unconditionally unstable if applied to linear undamped mass-spring systems [Eberly 2003].

In this section, we illustrate instability aspects of explicit integration schemes and we show how to obtain unconditional stability using a purely geometric scheme. We point out, how the proposed geometric scheme combines computational efficiency and unconditional stability which makes the approach especially appropriate for interactive deformable modeling applications. Although there exist many different explicit integration schemes, they all have the aforementioned stability problem in common. Therefore, we exemplify stability aspects using a modified Euler method from which we derive our unconditionally stable geometric scheme.

3.1 Explicit Numerical Integration

We consider a linear undamped mass-spring system to illustrate the instability of explicit methods. Fig. 2 shows a linear spring with resting length l_0 and spring constant k. The spring connects two points. One point is fixed at the origin while the other point with mass m is free and located at $x(t)$. The free point is pulled towards the equilibrium $x(t) = l_0$ by the force $f = -k(x(t) - l_0)$.

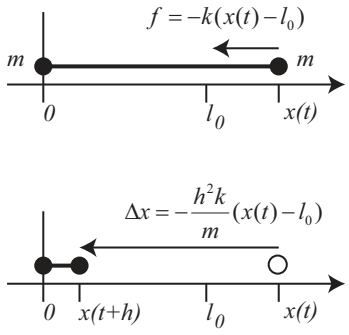

Figure 2: A linear spring integrated with the explicit scheme as shown in Eq. (1). If the time step is too large, internal forces erroneously increase the energy of the system.

With the modified Euler integration scheme

$$\begin{aligned} v(t+h) &= v(t) + h\frac{-k(x(t)-l_0)}{m} \\ x(t+h) &= x(t) + hv(t+h), \end{aligned} \tag{1}$$

the point velocity v is integrated with an explicit Euler step while the point position x is integrated with an implicit Euler step using the predicted velocity. Considering the state vector $[v\ x]^T$, Eq. (1) results in the system matrix

$$A = \begin{bmatrix} 1 & -\frac{kh}{m} \\ h & 1 - \frac{h^2 k}{m} \end{bmatrix} \tag{2}$$

with eigenvalues

$$e_0 = 1 - \frac{1}{2m}(h^2 k - \sqrt{-4mh^2 k + h^4 k^2}) \tag{3}$$

and

$$e_1 = 1 - \frac{1}{2m}(h^2 k + \sqrt{-4mh^2 k + h^4 k^2}) \tag{4}$$

Since A represents a discrete system, the spectral radius of A , i. e. the maximum magnitude of the eigenvalues e_0 and e_1, must not be larger than one to ensure stability of the system. The magnitude of eigenvalue e_0 converges to 1 with $|e_0| < 1$ for $h^2 k \to \infty$. However, it can be easily shown that the magnitude of e_1 is only smaller than one if h is smaller than $2\sqrt{\frac{m}{k}}$. If a larger time step h is chosen the system is unstable. Thus, the integration scheme is only conditionally stable.

To further exemplify the instability of the system, we perform one integration step assuming $v(t) = 0$. The step moves the free point by a distance $\Delta x = -\frac{h^2 k}{m}(x(t) - l_0)$. If either the time step h or the stiffness k are too large or the mass m is too small, the point not only overshoots the equilibrium position at l_0 but is moved to a position where the spring deformation $|x(t) - l_0|$ is larger than in the previous time step (see Fig. 2 bottom). The potential energy of the system has increased. Since we started with zero kinetic energy, the overall energy has erroneously increased. In subsequent steps the problem gets worse because the restoring force will be even larger than in earlier time steps.

In general, the stability problem of explicit integration schemes can be stated as follows: Elastic forces are the negative gradients of the elastic energy. As such, they always point towards an equilibrium configuration. However, explicit schemes scale these forces blindly to compute displacements of points. Therefore, displacements can overshoot the equilibrium by an amount which increases the deformation and the energy of the system instead of preserving or decreasing it which is required for stability.

One way to solve this problem would be to clamp the displacements such that points do not overshoot the equilibrium or goal position. In the simple one-dimensional spring example, the goal position of the moving point is simply $x(t) = l_0$. Unfortunately, in more general cases such as solid finite elements or geometrically complex meshes, goal positions of individual points cannot be defined easily. Our new approach solves this problem.

3.2 The Algorithm

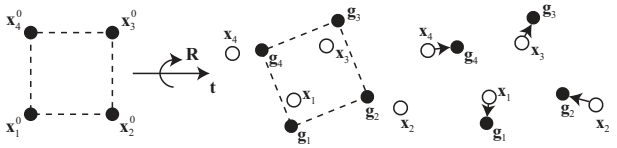

Figure 3: First, the original shape $\mathbf{x}_i^0$ is matched to the deformed shape $\mathbf{x}_i$. Then, the deformed points $\mathbf{x}_i$ are pulled towards the matched shape $\mathbf{g}_i$.

The basic idea behind our geometric algorithm is simple. All we need as input is a set of particles with masses m_i and an initial configuration (i. e. the initial positions $\mathbf{x}_i^0$ of the particles). No connectivity information or mesh is needed. The particles are simulated as a simple particle system without particle-particle interactions, but including response to collisions with the environment and including external forces such as gravity. After each time step, each particle is pulled towards its goal position $\mathbf{g_i}$. To compute the individual goal positions, we match the original configuration (or shape) defined by the $\mathbf{x}_i^0$ with the actual configuration defined by the actual positions of the particles $\mathbf{x}_i$ (see Fig. 3). The next two sections describe the shape matching process and how points are pulled towards goal positions.

3.3 Shape Matching

In our approach, the shape matching problem with a priori known correspondences can be stated as follows: Given two sets of points $\mathbf{x}_i^0$ and $\mathbf{x}_i$. Find the rotation matrix $\mathbf{R}$ and the translation vectors $\mathbf{t}$ and $\mathbf{t}_0$ which minimize

$$\sum_i w_i(\mathbf{R}(\mathbf{x}_i^0 - \mathbf{t}_0) + \mathbf{t} - \mathbf{x}_i)^2, \tag{5}$$

where the w_i are weights of individual points. In our case, the natural choice for the weights is $w_i = m_i$. The optimal translation vectors turn out to be the center of mass of the initial shape and the center of mass of the actual shape, i. e.

$$\mathbf{t}_0 = \mathbf{x}_{\text{cm}}^0 = \frac{\sum_i m_i \mathbf{x}_i^0}{\sum_i m_i}, \quad \mathbf{t} = \mathbf{x}_{\text{cm}} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i}, \tag{6}$$

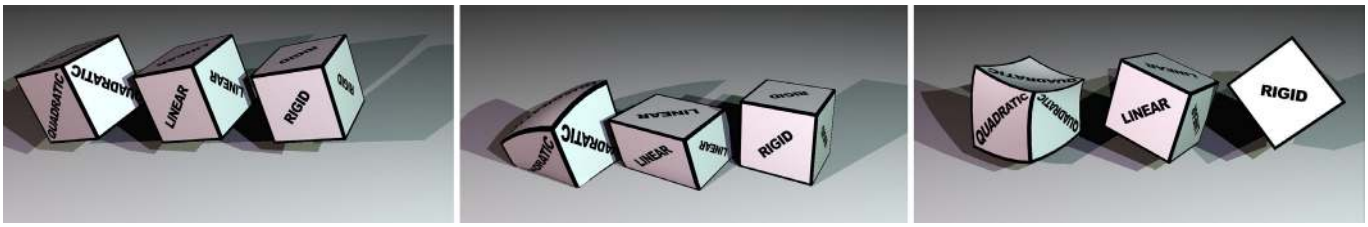

Figure 4: The basic shape matching algorithm is well suited for nearly rigid objects (third cube). The linear and quadratic extensions (middle and first cube) allow large deviations form the rest shape.

which is physically plausible. Finding the optimal rotation is slightly more involved. Let us define the relative locations $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{x}_{\text{cm}}^0$ and $\mathbf{p}_i = \mathbf{x}_i - \mathbf{x}_{\text{cm}}$ of points with respect to their center of mass and let us relax the problem of finding the optimal rotation matrix $\mathbf{R}$ to finding the optimal linear transformation $\mathbf{A}$. Now, the term to be minimized is $\sum_i m_i(\mathbf{A}\mathbf{q}_i - \mathbf{p}_i)^2$. Setting the derivatives with respect to all coefficients of $\mathbf{A}$ to zero yields the optimal transformation

$$\mathbf{A} = (\sum_i m_i \mathbf{p}_i \mathbf{q}_i^T)(\sum_i m_i \mathbf{q}_i \mathbf{q}_i^T)^{-1} = \mathbf{A}_{pq}\mathbf{A}_{qq}. \tag{7}$$

The second term $\mathbf{A}_{qq}$ is a symmetric matrix and, thus, contains only scaling but no rotation. Therefore, the optimal rotation $\mathbf{R}$ is the rotational part of $\mathbf{A}_{pq}$ which can be found via a polar decomposition $\mathbf{A}_{pq} = \mathbf{R}\mathbf{S}$, where the symmetric part is $\mathbf{S} = \sqrt{\mathbf{A}_{pq}^T\mathbf{A}_{pq}}$ and the rotational part is $\mathbf{R} = \mathbf{A}_{pq}\mathbf{S}^{-1}$. Finally, the goal positions can be computed as

$$\mathbf{g}_i = \mathbf{R}(\mathbf{x}_i^0 - \mathbf{x}_{\text{cm}}^0) + \mathbf{x}_{\text{cm}}. \tag{8}$$

3.4 Integration

The knowledge of the goal positions $\mathbf{g}_i$ can now be used to construct an integration scheme which does not overshoot:

$$\mathbf{v}_i(t+h) = \mathbf{v}_i(t) + \alpha\frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{h} + h\mathbf{f}_{\text{ext}}(t)/m_i \tag{9}$$

$$\mathbf{x}_i(t+h) = \mathbf{x}_i(t) + h\mathbf{v}_i(t+h), \tag{10}$$

where $\alpha = [0\ldots1]$ is a parameter which simulates stiffness. The only difference to the modified Euler scheme in Eq. (1) is the way the internal elastic forces are treated. For $\alpha = 1$, the term $(\mathbf{g}_i(t) - \mathbf{x}_i(t))/h$ is added to the velocity and, thus, $\mathbf{g}_i(t) - \mathbf{x}_i(t)$ is added to the position in the second line moving the point directly to the goal position, or towards the goal position for $\alpha < 1$.

For the one-dimensional spring example shown in Fig. 2, this iterative scheme yields the following update rule:

$$\begin{bmatrix} v(t+h) \\ x(t+h) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix} \begin{bmatrix} v(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} \alpha l_0/h \\ \alpha l_0 \end{bmatrix} \tag{11}$$

The first term on the right hand side is the system matrix. Its eigenvalues are $(1-\alpha/2) \pm i\sqrt{4\alpha - \alpha^2}/2$. The magnitude of both eigenvalues is 1, independent of α and the time step h. This shows that the scheme is unconditionally stable and does not introduce damping. This is also true for the general 3D case without external forces. As long as the external forces are independent of the locations of the points (such as gravitational forces) or applied only instantaneous (such as collision response forces), the system matrix does not change and the system remains stable.

3.5 Discussion

The described algorithm can be implemented efficiently. Both, the center of mass of the initial configuration $\mathbf{x}_{\text{cm}}^0$ as well as all $\mathbf{q}_i$ can be pre-computed. At each time step, the 3×3 matrix $\mathbf{A}_{pq} = \sum_i m_i \mathbf{p}_i \mathbf{q}_i^T$ has to be assembled. To evaluate $(\sqrt{\mathbf{A}_{pq}^T\mathbf{A}_{pq}})^{-1}$, we diagonalize the symmetric matrix $\mathbf{A}_{pq}^T\mathbf{A}_{pq}$ using 5-10 Jacobi rotations, the complexity of which is constant, i. e. independent of the number of points. In contrast to most methods for simulating deformable objects, the approach described so far is well-suited for stiff or almost rigid objects. In this basic form, it is less suited for objects undergoing large deformations, a limitation that is eliminated in Sections 4.2 and 4.3.

The fact that the shapes are matched using their centers of mass ensures that all impulses applied in Eq. (9) sum up to zero and conserve linear momentum. By using the m_i as the weights in shape matching and the computation of $\mathbf{A}_{pq}$ conservation or angular momentum is enforced as well.

A problem with the velocity update in Eq. (9) is that the behavior of the system depends on the time step, i. e. the velocity update is the same for varying time steps. This problem can be solved by setting $\alpha = h/\tau$, where $\tau \le h$ is a time constant.

4 Extensions

4.1 Rigid Body Dynamics

The method can be used to imitate a rigid body simulator by setting $\alpha = 1$. In this case, the points are moved to the goal positions $\mathbf{g}_i$ exactly at each time step. These positions represent a rotated and translated version of the initial shape. Given an arbitrary surface mesh, only a small subset of the vertices need to be animated as particles. The remaining vertices can be transformed at each time step using the computed transformation given by $\mathbf{R}$ and $\mathbf{t}$. The same holds for the methods described in the subsequent sections.

4.2 Linear Deformations

As mentioned before, the method described so far can only simulate small deviations from the rigid shape. To extend the range of motion, we use the linear transformation matrix $\mathbf{A}$ computed in Eq. (7). This matrix describes the best linear transformation of the initial shape to match the actual shape in the least squares sense. Instead of using $\mathbf{R}$ in Eq. (8) to compute the $\mathbf{g}_i$, we use the combination $\beta\mathbf{A} + (1-\beta)\mathbf{R}$, where β is an additional control parameter. This way, the goal shape is allowed to undergo a linear transformation. The presence of $\mathbf{R}$ in the sum ensures that there is still a

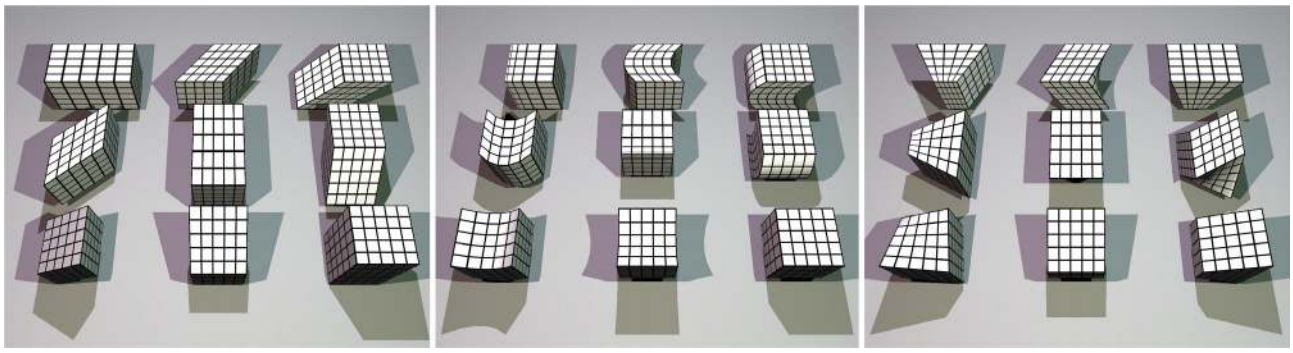

Figure 5: Visualization of all 3×9 modes defined by the coefficients of $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{Q}\ \mathbf{M}]$ defined in Eq. (12).

tendency towards the undeformed shape. To make sure that volume is conserved, we divide $\mathbf{A}$ by $\sqrt[3]{det(\mathbf{A})}$ ensuring that $det(\mathbf{A}) = 1$. For the standard approach we only need to compute $\mathbf{A}_{pq}$. Here, we also need the matrix $\mathbf{A}_{qq} = (\sum_i m_i \mathbf{q}_i \mathbf{q}_i^T)^{-1}$. Fortunately, this symmetric 3×3 matrix can be pre-computed.

4.3 Quadratic Deformations

Linear transformations can only represent shear and stretch. To extend the range of motion by twist and bending modes, we move from linear to quadratic transformations (see Fig. 4). We define a quadratic transformation as follows:

$$\mathbf{g}_i = [\mathbf{A}\ \mathbf{Q}\ \mathbf{M}]\tilde{\mathbf{q}}_i, \tag{12}$$

where $\mathbf{g}_i \in \mathbb{R}^3$, $\tilde{\mathbf{q}} = [q_x, q_y, q_z,\ q_x^2, q_y^2, q_z^2,\ q_x q_y, q_y q_z, q_z q_x]^T \in \mathbb{R}^9$, $\mathbf{A} \in \mathbb{R}^{3\times3}$ contains the coefficients for the linear terms, $\mathbf{Q} \in \mathbb{R}^{3\times3}$ the coefficients for the purely quadratic terms and $\mathbf{M} \in \mathbb{R}^{3\times3}$ the coefficients for the mixed terms. With $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{Q}\ \mathbf{M}] \in \mathbb{R}^{3\times9}$ we now have to minimize $\sum_i m_i(\tilde{\mathbf{A}}\tilde{\mathbf{q}}_i - \mathbf{p}_i)^2$. The optimal quadratic transformation turns out to be

$$\tilde{\mathbf{A}} = (\sum_i m_i \mathbf{p}_i \tilde{\mathbf{q}}_i^T)(\sum_i m_i \tilde{\mathbf{q}}_i \tilde{\mathbf{q}}_i^T)^{-1} = \tilde{\mathbf{A}}_{pq}\tilde{\mathbf{A}}_{qq}. \tag{13}$$

Again, the symmetric $\tilde{\mathbf{A}}_{qq} \in \mathbb{R}^{9\times9}$ as well as the $\tilde{\mathbf{q}}_i$ can be pre-computed. Analogous to the linear case, we use $\beta\tilde{\mathbf{A}} + (1-\beta)\tilde{\mathbf{R}}$ to compute the goal shape, where $\tilde{\mathbf{R}} \in \mathbb{R}^{3\times9} = [\mathbf{R}\ \mathbf{0}\ \mathbf{0}]$. The algorithm based on quadratic deformations is a computationally cheap imitation of methods using modal analysis. The linear shear and stretch modes and the additional bend and twist modes are shown in Fig. 5.

4.4 Cluster Based Deformation

To extend the range of motion even further, the set of particles can be divided into overlapping clusters. One way of doing this would be to use a volumetric mesh and interpret the vertices as particles and groups of vertices that are adjacent to the same element (e. g. tetrahedron) as a cluster. To generate the results in this paper, we used an alternative method. We regularly subdivide the space around a given surface mesh into overlapping cubical regions. For each region, we generate one cluster with all the vertices contained in this region.

At each integration step, the original shape of each cluster is matched with its actual shape. Then each cluster adds the term

$$\Delta\mathbf{v}_i = \alpha \frac{\mathbf{g}_i^c(t) - \mathbf{x}_i(t)}{h} \tag{14}$$

to all the particle it contains, where $\mathbf{g}_i^c(t)$ is the goal position of particle i with respect to cluster c.

4.5 Plasticity

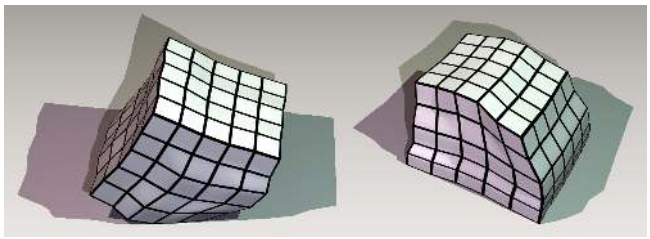

Figure 6: A cube defined by 8 clusters undergoes plastic deformations.

The algorithm for linear deformations described in Section 4.2 can easily be extended to simulate plasticity. In Section 3.3, we extracted the rotational part $\mathbf{R}$ of the linear transformation $\mathbf{A}$ using polar decomposition. Therefore, $\mathbf{A} = \mathbf{RS}$, where the matrix $\mathbf{S} = \mathbf{R}^T\mathbf{A}$ represents pure deformation. Because $\mathbf{S}$ is post-multiplied with $\mathbf{R}$, it represents a deformation in the initial, unrotated reference frame. Each cluster stores a plastic deformation state matrix $\mathbf{S}^p$. This state matrix is initialized with the identity matrix $\mathbf{I}$. At each time step, if the actual deformation $||\mathbf{S} - \mathbf{I}||_2$ exceeds a threshold c_{yield}, the state matrix is updated as

$$\mathbf{S}^p \leftarrow [\mathbf{I} + hc_{\text{creep}}(\mathbf{S} - \mathbf{I})]\mathbf{S}^p, \tag{15}$$

where h is the time step and c_{yield} and c_{creep} are parameters to control the plasticity [O'Brien et al. 2002]. The plasticity can be bound by testing whether $||\mathbf{S}^p - \mathbf{I}||_2$ exceeds a threshold $c_{\max}$. If this is the case, the plastic deformation is set to $\mathbf{I} + c_{\max}(\mathbf{S}^p - \mathbf{I})/||\mathbf{S}^p - \mathbf{I}||_2$. To make sure that volume is conserved by plasticity, we divide $\mathbf{S}^p$ by $\sqrt[3]{det(\mathbf{S}^p)}$ after it has been updated. Finally, the plastic state $\mathbf{S}^p$ is incorporated by replacing the definition of $\mathbf{q}_i = \mathbf{x}_i^0 - \mathbf{x}_{\text{cm}}^0$ in Eq. (7) with

$$\mathbf{q}_i = \mathbf{S}^p(\mathbf{x}_i^0 - \mathbf{x}_{\text{cm}}^0), \tag{16}$$

thereby deforming the original shape. Note that each time $\mathbf{S}^p$ is updated, $\mathbf{A}_{qq}$ respectively $\tilde{\mathbf{A}}_{qq}$ have to be updated too. Fig. 6 shows two different rest states after plastic deformation. To increase the level of detail, the cube is subdivided into 8 clusters.

5 Results

We have integrated our method into a game-like simulation environment for deformable objects and various experiments have been carried out to analyze the characteristics and the performance of the proposed method. All test scenarios presented in this section have been performed on a PC Pentium 4, 3.2 GHz.

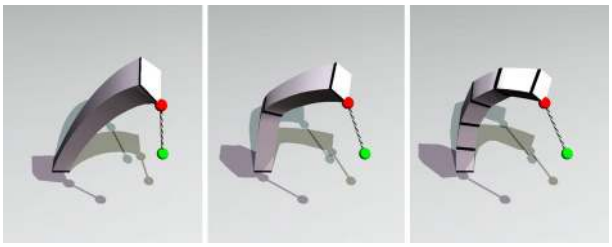

Figure 7: The flexibility of the object is influenced by the number of clusters. In this example, one, two, and five clusters are used from left to right, respectively.

Cluster based deformation. In Fig. 7, three quadratically deforming sticks with 60 points, but different numbers of clusters are shown. From left to right, the sticks are subdivided into one, two, and five clusters. In each figure, a spring forces is applied to the same surface point and originating from the same location in space. The experiment shows, that with an increasing number of clusters, the deformation gets more detailed and the physical plausibility is improved. In general, the number of clusters is user defined. For simple, sphere-like objects, one cluster might me sufficient while for more complex geometries a subdivision might be necessary.

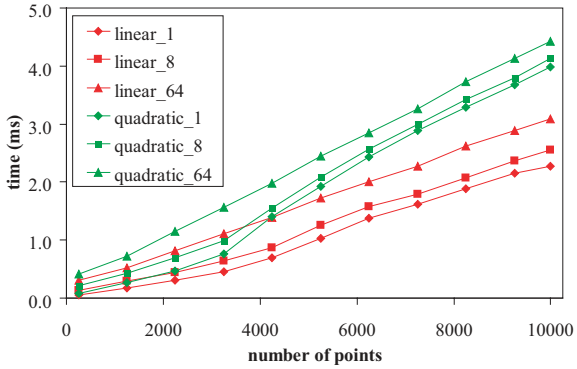

Figure 8: Time complexity of the presented methods in milliseconds per frame versus the number of points being simulated.

Performance. The performance of our approach depends on the number of points that are used for the shape matching and on the number of clusters. In order to illustrate the scaling of the performance with respect to these two criterions, we have performed the following experiment. A number of points are placed at arbitrary positions and corresponding arbitrarily placed goal positions are defined, resulting in a random initial displacement for each point. Based on this scenario, measurements have been performed with varying numbers of points and clusters employing linear and quadratic deformation models. Note, that the computational complexity of the rigid body case is equal to the linear deformation model. Fig. 8 illustrates the results. It can be seen, that the performance scales linearly with the number of points. Further, a larger number of clusters requires more polar decompositions which reduces the performance. Also, the computation of the quadratic deformation model is more involved compared to the linear model. Still, 100 objects each containing 100 simulated points subdivided into 8 clusters can be animated with the quadratic approach at 50 frames per second.

It is difficult to compare our approach to the Finite Element Method or mass-spring models in terms of performance. As Fig. 8 shows, our technique depends on the deformation modes and the number of clusters chosen with no analogy in FEM or mass-spring systems.

Complex simulation scenarios. Fig. 9 depicts a scene of 384 objects, $2,448$ clusters and $55,200$ points. The quadratic shape matching in this case takes between 0.008 and 0.096 milliseconds per frame and object depending on object complexity. In the teaser sequence Fig. 1 145 objects, $1,728$ clusters and $24,618$ points are animated. The quadratic shape matching takes 0.12 milliseconds per frame and object.

Interactivity. Fig. 10 illustrates objects whose geometrical complexity can be considered typical for games: a head consisting of 8 clusters and 66 points, and some spheres consisting of 1 cluster and 13 points. To improve the visual impression, the objects are combined with surface meshes, consisting of $6,460$ and $2,000$ faces, respectively. This environment can be handled at interactive frame rates including user interaction with draggers, collision handling, and visualization.

Stability. Fig. 11 demonstrates the fact that our approach can handle degenerated geometries. In combination with the unconditional stability of the numerical integration scheme, this allows for stable animations independent of any user interaction.

6 Conclusions and Future Work

We have presented a new approach to geometric deformations. The underlying model is related to modal analysis approaches, but shows various differences. Our model does not require any preprocessing or auxiliary data structures, it is efficient to compute, and provides unconditionally stable simulations. However, in contrast to modal analysis approaches our model is not physically motivated. While the accuracy of modal analysis methods can be arbitrarily improved by just considering a larger number of deformation modes, our approach gets more involved if additional higher-order deformation modes would be considered. Further, higher-order modes do not necessarily improve the modeling accuracy in our approach, since they are not related to physical vibration modes.

Our approach can handle a reasonable number of deformable objects in real-time. However, adequate collision handling is required. In order to illustrate our results, we have used an existing penalty-based collision handling technique [Teschner et al. 2003], [Heidelberger et al. 2004], but found it to be a performance bottleneck. Alternatively, the Bounded Deformation Tree [James and Pai 2004] could be considered. Although the existing work is very promising, further research in deformable collision handling is necessary to match the advances in deformable modeling [Teschner et al. 2005].

The ”plug and simulate” handling of a large variety of objects, the efficiency in terms of memory and computational complexity, and the unconditional stability of the dynamic simulation make the approach particularly interesting for games. While we have investigated plasticity as one of the first model extensions, ongoing work focuses on further extensions such as fracture animation. Fracturing is particularly interesting since changing connectivity requires only minor updates of data structures.

Figure 9: In this massive scene consisting of 384 objects, the computation of the dynamics is possible in real-time while collision detection and collision response computations are not.

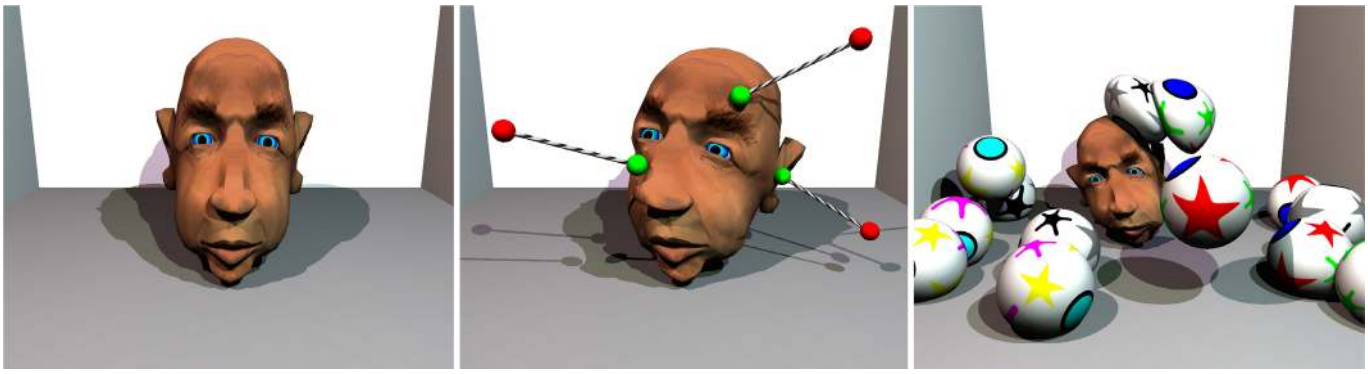

Figure 10: A head model simulated with 8 clusters reacts to user interaction or to collisions with other deformable objects.

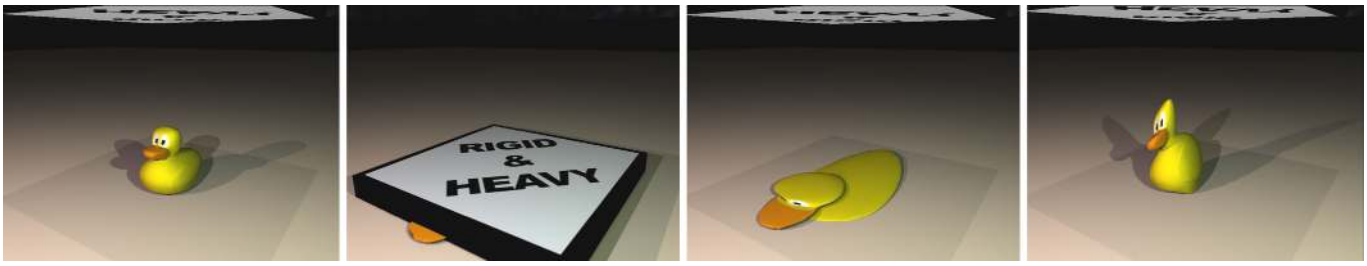

Figure 11: Squeezing a duck model demonstrates the stability of the approach and the ability to recover from highly deformed or inverted configurations.

7 Acknowledgements

The authors would like to thank Marco Heidelberger for the duck model. This project was funded by the Swiss National Commission for Technology and Innovation (CTI) project no. 6310.1 KTS-ET.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 2000, 157–164.

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

BARZEL, R., HUGHES, J. F., AND WOOD, D. N. 1996. Plausible motion simulation for computer graphics animation. *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, 183–197.

BARZEL, R. 1997. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications 17*, 31–39.

BESL, P., AND MCKAY, N. 1992. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2, 239–256.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIC, Z. 2002. Interactive skeleton-driven dynamic deformations. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 586–593.

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

DESBRUN, M., AND CANI, M.-P. 1995. Animating soft substances with implicit surfaces. In *Computer Graphics Proceedings*, ACM SIGGRAPH, 287–290.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96*, 61–76.

DESBRUN, M., SCHRODER, P., AND BARR, A. H. 1999. Interactive animation of structured deformable objects. In *Graphics Interface*, 1–8.

EBERLY, D. H. 2003. *Game Physics*. Morgan Kaufmann.

FAUGERAS, O. D., AND HEBERT, M. 1983. A 3-d recognition and positioning algorithm using geometric matching between primitive surfaces. In *Int. Joint Conference on Artificial Intelligence*, 996–1002.

GRINSPUN, E., KRYSL, P., AND SCHRODER, P. 2002. Charms: A simple framework for adaptive simulation. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 281–290.

HAUSER, K. K., SHEN, C., AND O'BRIEN, J. F. 2003. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, A K Peters, CIPS, Canadian Human-Computer Commnication Society, 247–256.

HEIDELBERGER, B., TESCHNER, M., KEISER, R., MÜLLER, M., AND GROSS, M. 2004. Consistent penetration depth estimation for deformable collision response. In *Proceedings of Vision, Modeling, Visualization VMV'04, Stanford, USA*, 339–346.

HORN, B. K. P. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A 4*, 4, 629–642.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. *Eurographics* (Sept.), 131–140.

JAMES, D., AND PAI, D. K. 1999. Artdefo, accurate real time deformable objects. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 99, 65–72.

JAMES, D. L., AND PAI, D. K. 2002. Dyrt: Dynamic response textures for real time deformation simulation with graphics hardware. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH, 582–585.

JAMES, D. L., AND PAI, D. K. 2004. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph. 23*, 3, 393–398.

KANATANI, K. 1994. Analysis of 3-d rotation fitting. *IEEE Trans. Pattern Anal. Mach. Intell. 16*, 5, 543–549.

KAZHDAN, M., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2004. Shape matching and anisotropy. *ACM Trans. Graph. 23*, 3, 623–629.

KENT, J. R., CARLSON, W. E., AND PARENT, R. E. 1992. Shape transformation for polyhedral objects. *Computer Graphics 26*, 47–54.

LORUSSO, A., EGGERT, D. W., AND FISHER, R. B. 1995. A comparison of four algorithms for estimating 3-d rigid transformations. In *BMVC '95: Proceedings of the 1995 British conference on Machine vision (Vol. 1)*, BMVA Press, 237–246.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 1992, 309–312.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. *Proceedings of Graphics Interface (GI 2004)*, 239–246.

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

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. *Proceedings of 2004 ACM SIGGRAPH Symposium on Computer Animation*, 141–151.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 291–294.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH, 215–222.

SHEN, C., HAUSER, K., GATCHALIAN, C., AND O'BRIEN, J. 2002. Modal analysis for real-time viscoelastic deformation. *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (july).

SHOEMAKE, K., AND DUFF, T. 1992. Matrix animation and polar decomposition. In *Graphics Interface*, 258–264.

TERAN, J., BLEMKER, S., HING, V. N. T., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. *Proceedings of 2003 ACM SIGGRAPH Symposium on Computer Animation*, 68–74.

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

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision, Modeling, Visualization VMV'03*, 47–54.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., AND VOLINO, P. 2005. Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (March), 61–81.

TONNESEN, D. 1998. *Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation*. PhD thesis, University of Toronto.

UMEYAMA, S. 1991. Least squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 4 (Apr.), 376–80.