Stable Real-Time Deformations

Matthias Müller* Julie Dorsey Leonard McMillan Robert Jagnow Barbara Cutler

ETH Zürich Massachusetts Institute of Technology, Laboratory for Computer Science

Abstract

The linear strain measures that are commonly used in real-time animations of deformable objects yield fast and stable simulations. However, they are not suitable for large deformations. Recently, more realistic results have been achieved in computer graphics by using Green's non-linear strain tensor, but the non-linearity makes the simulation more costly and introduces numerical problems.

In this paper, we present a new simulation technique that is stable and fast like linear models, but without the disturbing artifacts that occur with large deformations. As a precomputation step, a linear stiffness matrix is computed for the system. At every time step of the simulation, we compute a tensor field that describes the local rotations of all the vertices in the mesh. This field allows us to compute the elastic forces in a non-rotated reference frame while using the precomputed stiffness matrix. The method can be applied to both finite element models and mass-spring systems. Our approach provides robustness, speed, and a realistic appearance in the simulation of large deformations.

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

Keywords: Physically Based Animation, Finite Elements, Large Deformations, Elasticity, Stiffness Warping

1 Introduction

Mathematical and physical modeling of deformable objects has a long history in mechanical engineering and materials science. In those disciplines, the main objective is to model the physical world as accurately as possible. In graphics applications, the primary concern is usually the computational efficiency of generating plausible behaviors, rather than the accurate prediction of exact results. The most widely used technique to model deformable objects is to view material as a continuum. In this case, the constitutive laws yield partial differential equations that describe the static and dynamic behavior of the material. These equations are usually solved numerically using the Finite Element Method (FEM) [Bathe 1982] or finite differences [Terzopoulos et al. 1987]. Such simulations are typically done offline – that is, computers spend minutes or hours to arrive at a single answer or a simulation of a few seconds.

Real-time simulation of deformable objects is a younger field. The performance of modern computers and graphics hardware has made physically-based animation possible in real time. But even with today's best hardware and most sophisticated techniques [Debunne et al. 2001; Wu et al. 2001; Zhuang 2000], only a few hundred elements with small deformations have been simulated in real-time to date. Since simulating the dynamic behavior of deformable objects in real time is an important and challenging task, a great deal of work has been done in the field and a large variety of techniques and methods have been proposed in the last two decades [Gibson and Mitrich 1997]. Typical applications for real-time deformable objects include virtual surgery [Debunne et al. 2001; Wu et al. 2001], virtual sculpting, games or any application requiring an interactive virtual environment. A real-time simulator could offer artists the option to design and test animations interactively before rendering their work offline in higher quality.

An interactive simulation system needs to meet two main requirements. It certainly needs to be fast enough to generate 15 to 20 frames per second. Speed, however, is not the only requirement. Ideally, we want to give the user of the system complete freedom of action. Thus, stability and robustness are just as important as the frame rate.

With the availability of fast computers, there has been a trend in real-time animation away from simple models such as mass-spring systems toward the more sophisticated Finite Element approach. FEM is computationally more expensive, but it is physically more accurate, and the object's deformation behavior can be specified using a few material properties instead of adjusting a large number of spring constants. However, because of its computational cost, only the simplest variant of FEM has been used so far – namely tetrahedral elements with linear shape functions. While not suitable for engineering analysis, such models are sufficient to obtain visually plausible results.

There is an additional option when choosing an FEM model – namely how strain is measured with respect to the deformation of an object. Linear elasticity only models small deformations accurately, but its computational cost is much lower than the cost of a non-linear strain measure. One important feature of the linear approach is that the stiffness matrix of the system is constant and numerically well-conditioned, yielding a fast and stable simulation. Under large rotational deformation, however, objects increase unnaturally in volume because the linear model is only a first order approximation at the undeformed state (see Fig. 7).

Non-linear elasticity, on the other hand, models large rotational deformations accurately [Picinbono et al. 2000]. With a non-linear strain measure, the stiffness matrix is no longer constant. For implicit integration it must be reevaluated at every time step as the Jacobian of the non-linear function that describes the internal elastic forces. This process slows down the simulation and introduces numerical instabilities when the Jacobian is evaluated far from the equilibrium state. This is why models have usually only been subjected to small displacements in demonstrations of real-time systems thus far. Dramatic deformations are not possible without ei-

*e-mail: muellerm@inf.ethz.ch

ther slowing the simulator down or risking numerical divergence.

In this paper, we propose a new technique that is as fast and stable as linear elasticity while avoiding the artifacts associated with large deformations. We do this by warping the stiffness matrix of the system according to a tensor field that describes local rotations of the deformed material. In this way, we can use a precomputed stiffness matrix. The evaluation of the tensor field is much cheaper than the cost of a single time step. Our technique is easy to understand and implement, making it practical for a wide variety of applications.

1.1 Related Work

Many methods have been proposed to simulate deformable objects in real time. We will discuss just a few recent publications and papers that describe those techniques similar to ours.

To improve the numerical stability of the simulation, Terzopoulos *et al.* [Terzopoulos and Witkin 1988] proposed a hybrid model that breaks a deformable object into a rigid and a deformable component. The rigid reference body captures the rigid-body motion while a discretized displacement function gives the location of mesh nodes relative to their position within the rigid body. As in their approach, we handle the rotational component of the deformation separately. However, they use one single rotation matrix for the entire model – namely the one associated with the underlying rigid body frame – even if regions of the deformable object undergo large rotations while other regions don't rotate at all.

In ArtDefo (Accurate Real Time Deformable Objects) [James and Pai 1999], James *et al.* used linear elasticity in connection with the Boundary Element Method (BEM) to deform objects in real time. Because of the linearity of the model, many system responses can be precomputed and then combined later in real time. However, the linear model is not accurate for large deformations, as we already mentioned.

Desbrun *et al.* [Desbrun et al. 1999] split the forces in mass-spring networks into linear and non-linear (rotational) parts. The rotational part is first neglected to compute a rapid approximation of the implicit integration. Then they correct the estimate to preserve momentum.

To guarantee a real-time frame rate, Debunne *et al.* [Debunne et al. 2001] use an automatic space and time adaptive level of detail technique. The body is partitioned in a multi-resolution hierarchy of tetrahedral meshes. The high resolution meshes are only used in regions of high stress. This reduces the number of active elements, thus increasing the speed of the simulation. We also use this method in our system to further increase the speed of our simulation.

Wu's approach [Wu et al. 2001] is very similar to Debunne's technique. They use progressive meshes to adapt the number of elements according to the internal stresses.

1.2 Overview

In the next section, we introduce linear and non-linear models of static and dynamic deformation and discuss their advantages and disadvantages for real-time simulation. This motivates the need for our technique called *Stiffness Warping*, which we describe in Section 3. We propose two ways of computing the rotation field of a deformed mesh along which the stiffness matrix is warped. A comparison of our technique with linear and non-linear approaches shows the advantages of the method. In the last section, we present a collection of our results.

2 Modeling Deformation

There are a variety of ways to model the behavior of deformable objects. Mass-spring networks are popular in real-time simulators because they are simple to implement. However, models that treat objects as a continuum have several advantages over simple mass-spring networks. The physical material properties can be described using a few parameters, which can be looked up in textbooks, and the force coupling between mass elements is defined throughout the volume rather than according to the spring network. As a result, continuous models yield more accurate results. The deformation of an object in such a model is described by a boundary value partial differential equation. For realistic objects, this equation cannot be solved analytically. A standard technique to solve it numerically is the Finite Element Method [Bathe 1982]. Using FEM, an object is subdivided into elements of finite size – typically polyhedra – and a continuous deformation field within each element is interpolated from the deformation vectors at the vertices. Once the interpolation functions for all the elements are chosen, the deformation vectors at all the vertices describe a piecewise continuous deformation field. This field incorporated into the partial differential equation yields a set of simultaneous algebraic equations for the deformation vectors at the vertices.

Regardless of the choice of element type and shape functions, the Finite Element Method yields an algebraic function F that relates the deformed positions of all the nodes in the object to the internal elastic forces at all the nodes:

$$\mathbf{f}_{elastic} = F(\mathbf{x} - \mathbf{x}_0), \tag{1}$$

where $\mathbf{f}_{elastic} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^T$ contains the internal force vectors of all n nodes, and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T$ and $\mathbf{x}_0 = (\mathbf{x}_{01}, \mathbf{x}_{02}, \ldots, \mathbf{x}_{0n})^T$ represent their deformed and original positions, respectively. This even holds for mass-spring networks. The function $F : \mathbf{R}^{3n} \rightarrow \mathbf{R}^{3n}$ is, in general, non-linear and encapsulates the material properties as well as the type of mesh and discretization used.

2.1 Dynamic Deformation

In a dynamic system, the coordinate vector $\mathbf{x}$ is a function of time, $\mathbf{x}(t)$. The dynamic equilibrium equation has the following form:

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + F(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_{\text{ext}}, \tag{2}$$

where $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivatives of $\mathbf{x}$ with respect to time, M is the mass matrix and C the damping matrix [Cook 1981]. Eqn. (2) defines a coupled system of $3n$ ordinary differential equations for the n position vectors contained in $\mathbf{x}$. To solve them, the continuous $3n$-dimensional function $\mathbf{x}(t)$ is approximated by a series of vectors $\mathbf{x}^0, \mathbf{x}^1, \ldots \mathbf{x}^i, \ldots$, where $\mathbf{x}^i$ approximates $\mathbf{x}(i \cdot \Delta t)$. In a first step, (2) is transformed into a system of $2 \times 3n$ equations of first derivatives:

$$\begin{array}{ll} \dot{\mathbf{x}} & = \mathbf{v} \\ M\dot{\mathbf{v}} & = -C\mathbf{v} - F(\mathbf{x} - \mathbf{x}_0) + \mathbf{f}_{\text{ext}}, \end{array} \tag{3}$$

where $\mathbf{v}$ is an additional vector of $3n$ velocities. Although there are mathematically more accurate integration methods (see [Pozrikidis 1998]), Euler's first order method is known to better handle discontinuities (caused, for instance, by collisions) than higher order methods [Desbrun et al. 1999]. The implicit form of Euler's method approximates (3) as follows:

$$\begin{array}{ll} \mathbf{x}^{i+1} & = \mathbf{x}^i + \Delta t \mathbf{v}^{i+1} \\ M\mathbf{v}^{i+1} & = M\mathbf{v}^i + \Delta t(-C\mathbf{v}^{i+1} - F(\mathbf{x}^{i+1} - \mathbf{x}_0) + \mathbf{f}_{\text{ext}}^{i+1}). \end{array} \tag{4}$$

In order to find the positions and velocities at time $(i+1)\Delta t$, a coupled system of algebraic equations needs to be solved, because the unknown values $\mathbf{x}^{i+1}$ and $\mathbf{v}^{i+1}$ appear on both sides of Euler's implicit equation. To compute positions and velocities at time

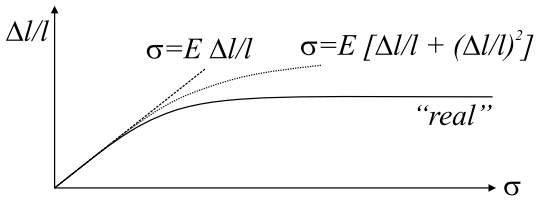


Figure 1: Quadratic stress approximates the real stress-deformation curve better than linear stress. It is not a full second order approximation of the real curve though.

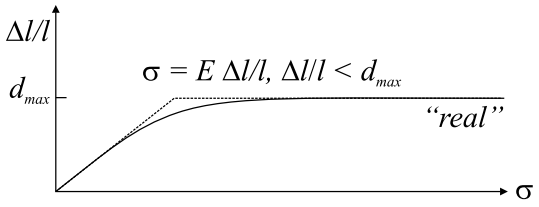


Figure 2: With a method to prevent material from over-stretching, the linear model can fit a real stress curve appropriately.

$(i+1)\Delta t$ we use implicit integration because it is stable for much larger time steps than explicit integration [Witkin and Baraff 1998], which only uses quantities at time $i\Delta t$. (For a detailed discussion of implicit and explicit methods see [Witkin and Baraff 1997].)

2.2 Non-Linear Elasticity

In order for a strain measure to be accurate for large deformations, it should not include the rigid body motions of the simulation elements. This can be achieved by defining strain as the change in length of an infinitesimal material vector going from the original configuration to the deformed configuration. In 3D, it is more practical to measure the change of the *squared* length of a vector, because the squared length is merely the dot product of a vector with itself. This is why the Green-Lagrange strain tensor [Bathe 1982] is defined via the expression

$$\frac{1}{2}\frac{(ds)^2-(ds_0)^2}{(ds_0)^2}, \tag{5}$$

where ds_0 and ds are corresponding infinitesimal vectors in the undeformed and deformed configuration respectively. The omission of the square root when measuring the length of a vector yields quadratic strain-displacement and stress-displacement relationships. This is a nice side effect because for some materials, quadratic stress approximates the real displacement-stress curve better than linear stress (Fig. 1). Green-Lagrange strain is not a full second order approximation of the real stress curve though, because, as with the linear model, there is only one coefficient (i.e. Young's modulus $\mathbf{E}$) to fit the curve.

Desbrun [1999] describes a method to prevent material from over-stretching. He approximates the real stress curve with a piecewise linear function (see Fig. 2). When combined with a linear stress measure, this method yields realistic results. Thus, the reason why one would use a quadratic strain tensor in computer graphics and real-time simulations is not because a linear deformation-stress relationship would not yield plausible results, but because linear strain tensors are not invariant under rigid body transformations, and therefore are inappropriate for rendering *rotational* deformations correctly.

With a quadratic stain tensor, the function F describing the internal elastic forces becomes non-linear. Thus, in both static (1) and dynamic (2) simulations, a non-linear algebraic system of equations has to be solved. This generally involves the computation of the Jacobian J of F. Since F is $3n$-dimensional, J is a matrix of dimension $3n \times 3n$. Even though J is usually sparse, its evaluation is computationally expensive. Moreover, the numerical conditioning of J deteriorates when evaluated far from the equilibrium state.

2.3 Linear Elasticity

In linear elasticity, F is replaced by a first order approximation:

$$F(\mathbf{x}-\mathbf{x}_0) = K\cdot(\mathbf{x}-\mathbf{x}_0) + O(||\mathbf{x}-\mathbf{x}_0||^2), \tag{6}$$

where K is the Jacobian $\partial F/\partial \mathbf{x}$ of F evaluated at $\mathbf{x}_0$, usually called the stiffness matrix of the system. The stiffness matrix is computed only once before the simulation is run. At every time step, a linear system (usually well conditioned) has to be solved. This is why a linear simulation is faster and more stable than a simulation based on non-linear elasticity. The drawback of this approach, however, is that large deformations are not rendered correctly. More precisely, linear elastic forces are invariant under translations but not under rotations. This raises the question of whether it is possible to work with a constant linear stiffness matrix and extract the rotational part of the deformation. The next section describes our new technique called *Stiffness Warping*, which is based on this idea.

3 Stiffness Warping

In linear elasticity, the elastic forces for a single tetrahedral element in 3D are evaluated as follows:

$$\mathbf{f}_{elastic} = K\cdot(\mathbf{x}-\mathbf{x}_0), \tag{7}$$

where $K \in \mathbf{R}^{12\times 12}$ is the element's stiffness matrix and $\mathbf{f}_{elastic}$, $\mathbf{x}$ and $\mathbf{x}_0 \in \mathbf{R}^{12}$ contain the elastic forces, the displaced positions and the original positions of the four vertices of the tetrahedron. As long as the deformed shape $\mathbf{x}$ is only stretched and translated with respect to the original shape $\mathbf{x}_0$, the linear approach yields plausible results. If the transformation from $\mathbf{x}_0$ to $\mathbf{x}$ contains a rotation, the artifacts associated with a linear model emerge.

Let us assume now that we know a global rotational component R_x of the rigid body transformation of the element, where $R_x \in \mathbf{R}^{3\times 3}$ is a 3D (orthogonal) rotation matrix. We can then construct $R_e \in \mathbf{R}^{12\times 12}$, which contains four copies of R_x along its diagonal and zeros everywhere else:

$$R_e = \begin{bmatrix} R_x & & & 0 \\ & R_x & & \\ & & R_x & \\ 0 & & & R_x \end{bmatrix} \tag{8}$$

This matrix rotates quantities of all four nodes of the tetrahedron by the same matrix R_x. If we compute the elastic forces as

$$\mathbf{f}_{elastic} = R_e K\cdot(R_e^{-1}\mathbf{x}-\mathbf{x}_0), \tag{9}$$

we get the same forces as if R_x was not present in $\mathbf{x}$ (Fig. 3). We first rotate the deformed positions $\mathbf{x}$ back to their original coordinate frame using the inverse R_e^{-1}. The forces are then computed in this coordinate frame as $K\cdot(R_e^{-1}\mathbf{x}-\mathbf{x}_0)$ and then rotated back using R_e.

Let k_{ij} be the 3×3 sub-matrix of K containing entries K_{vw}, with $3i-2 \le v \le 3i$ and $3j-2 \le w \le 3j$. Using (9), we get for the force $\mathbf{f}_i$ at vertex i:

$$\mathbf{f}_i = R_x\sum_{j=1}^{n} k_{ij}(R_x^{-1}\mathbf{x}_j - \mathbf{x}_{0j}), \tag{10}$$

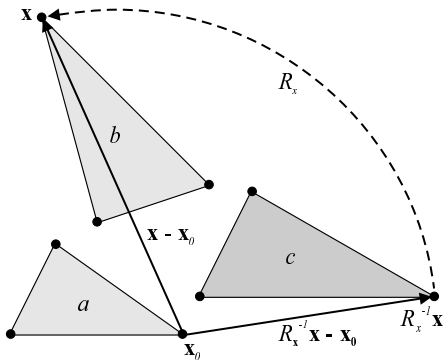


Figure 3: If the rotational part R_x of the deformation $\mathbf{x}$ is known, the forces can be computed with respect to a deformation $R_x^{-1}\mathbf{x}$ that only contains translation and stretching. Here, the original element (a) is deformed (b), and then rotated back into the original coordinate frame (c).

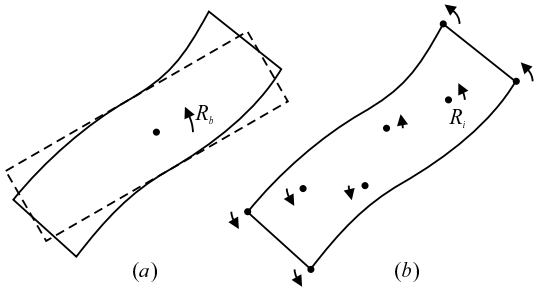


Figure 4: Instead of using a single rotation matrix R_b from an underlying rigid body frame (a), we compute local matrices R_i for every vertex (b).

where $i \in (1 \dots 4)$ and $\mathbf{x}_j$ and $\mathbf{x}_{0j}$ are the displaced and original positions of vertex j. If we use the same approach for the entire mesh, we get the following formula for the elastic force at vertex i:

$$\mathbf{f}_i = R_b \sum_{j=1}^{n} k_{ij}(R_b^{-1}\mathbf{x}_j - \mathbf{x}_{0j}), \tag{11}$$

where $i \in (1 \dots n)$. The k_{ij}'s are now sub-matrices of $K \in \mathbf{R}^{3n\times 3n}$, the stiffness matrix of the entire mesh. This raises the question of what R_b – the mesh's rotation – should be in this case. If we kept track of a global rigid body frame associated with the deformable body as in Terzopoulos' model [Terzopoulos and Witkin 1988], we could derive R_b from this rigid body rotation. For stiff materials with little deformation but arbitrary rigid body motion, this model would yield acceptable results. Large deformations other than the rigid body modes would still yield the typical artifacts of a linear model, such as growth in volume.

A natural extension of the rigid body approach is to use individual rotation matrices R_i for every vertex i in the mesh (Fig. 4). Hence, instead of rotating the stiffness matrix K, we *warp* it along a rotation field described by the matrices $R_i, i = (1 \dots n)$. For $\mathbf{f}_i$, we now get:

$$\mathbf{f}_i = R_i \sum_{j=1}^{n} k_{ij}(R_i^{-1}\mathbf{x}_j - \mathbf{x}_{0j}). \tag{12}$$

The only nonzero k_{ij} in (12) are those for which there is an edge (i, j) in the mesh. Thus, the quantities used to compute $\mathbf{f}_i$ are all located at vertices immediately adjacent to vertex i. Therefore, the rotation matrix R_i is only used in the local neighborhood of vertex i. In this way, the force at vertex i is computed exactly as in linear FEM, but as if the local neighborhood of vertex i were rotated back by R_i^{-1}.

We also tried to use the individual R_j's to compute $\mathbf{f}_i$

$$\mathbf{f}_i = R_i \sum_{j=1}^{n} k_{ij}(R_j^{-1}\mathbf{x}_j - \mathbf{x}_{0j}), \tag{13}$$

but observed that instability may emerge when more than one rotation matrix is involved in the computation of $\mathbf{f}_i$ and that the stability depends on the method used to compute R_i.

Computing the elastic forces as in Eqn. (12) yields fast and robust simulations. However, the forces are not guaranteed to yield zero total momentum as elastic forces should. Errors come from the fact that the same rotation matrix is used in a finite size environment and also depend on the way the rotation matrices are computed (see next section). Even though the errors in the force vectors at individual vertices are tiny and don't show as long as objects are anchored, their sum – if non-zero – acts as a ghost force on free floating objects. In [Desbrun et al. 1999] Desbrun shows how to solve this problem by performing a simple and computationally cheap correction step after every time step. We used the same technique in our simulator.

3.1 Rotation Tensor Field

We now have to answer the question of how to estimate the local rotations of a deformed mesh. Extracting the rotational part of a mapping between two arbitrary sets of vectors is not straightforward and not unique if the two sets are not related via a pure 3D rotation. One approach to finding an optimal rotation matrix is to minimize an error function using a least squares method. This, however, requires the ability to take derivatives with respect to a matrix. Lasenby *et al.*[1998] describe an elegant alternative that uses geometric algebra [Hestenes and Sobczyk 1984]. In the geometric algebra notation, rotations can be represented by multivectors (rotors). Given two sets of vectors, the theory allows for minimizing with respect to such rotors and for finding optimal rotations.

For two given sets of vectors $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ with cardinality N a matrix $F \in \mathbf{R}^{3\times 3}$ is formed:

$$F_{ij} = \sum_{k=1}^{N} (\mathbf{n}_i \cdot \mathbf{u}_k)(\mathbf{n}_j \cdot \mathbf{v}_k). \tag{14}$$

where the vectors $\mathbf{n}_1, \mathbf{n}_2$ and $\mathbf{n}_3$ are orthonormal basis vectors of $\mathbf{R}^3$. In a second step, F is decomposed by SVD (singular value decomposition [Golub and Loan 1996]), which yields $F = USV^T$. The rotation matrix R is then simply given by the product

$$R = VU^T. \tag{15}$$

For our simulator, we have also used a simpler and faster technique to compute local rotations. We found that the stability of Eqn. (12) is not sensitive to the choice of the rotation field and that even a very simple approach can yield stable and fast simulations. Figure 5 illustrates our faster approximation procedure.

For corresponding vertices in the undeformed and deformed mesh, we compute orthonormal frames of vectors $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3)$ and $(\mathbf{n}'_1, \mathbf{n}'_2, \mathbf{n}'_3)$ based on a selection of outgoing edges $\mathbf{e}_i$ and $\mathbf{e}'_i$, respectively. More specifically, $\mathbf{n}_1$ is computed as the normalized average of three deterministically chosen edges. The second vector $\mathbf{n}_2$ is evaluated as the cross product of $\mathbf{n}_1$ and the direction of a chosen edge. The last vector $\mathbf{n}_3$ is the cross product of $\mathbf{n}_1$ and $\mathbf{n}_2$. These three vectors form a matrix $N = (\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3)$. The same

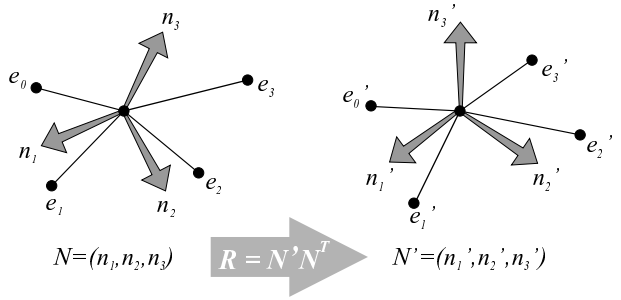


Figure 5: A fast way of estimating the rotational part of the deformation at a node is to compute the relative rotation between two orthonormal vector frames that are based on the directions of adjacent edges.

procedure applied to the deformed mesh yields a matrix N'. It is important that N' is computed using the exact same edges, but in their deformed directions $\mathbf{e}_i'$. The rotation matrix we are looking for can now be evaluated as

$$R = N'N^T. \tag{16}$$

In the case of a rigid body, where the two meshes are related via rotations and translations only, this simple approach yields the correct constant rotation matrix for all the vertices.

3.2 The Algorithm

Let us summarize the entire simulation algorithm:

- $K \leftarrow \frac{\partial F}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x_0}}$ ($K \in \mathbf{R}^{3n\times 3n}$)
- $\mathbf{x}_i^0 \leftarrow \mathbf{x}_{0i}$; $\mathbf{v}_i^0 \leftarrow \mathbf{0}$ for all $i \in (1 \ldots n)$
- $t \leftarrow 0$
- loop
  - evaluate R_i for all $i \in (1 \ldots n)$
  - solve $\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \frac{\Delta t}{m_i}[-c_i\mathbf{v}_i^{t+1} - R_i \sum_{j=1}^n k_{ij}(R_i^{-1}(\mathbf{x}_j^t + \Delta t\mathbf{v}_j^{t+1}) - \mathbf{x}_{0j}) + \mathbf{f}_{ext}^{t+1}]$ for all unknown $\mathbf{v}_i^{t+1}$, $i \in (1 \ldots n)$
  - set $\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i^t + \Delta t\mathbf{v}_i^{t+1}$ for all $i \in (1 \ldots n)$
  - $t \leftarrow t + 1$
- end loop

The function $F(\mathbf{x}) : \mathbf{R}^{3n} \rightarrow \mathbf{R}^{3n}$ describes internal elastic forces given the deformed coordinates $\mathbf{x}$ of all n vertices of a mesh. This function does not necessarily need to stem from a Finite Element discretization – it can also be defined by a spring network. First, the Jacobian K of F is evaluated. Then the positions and velocities of all the vertices are initialized and the time is set to zero.

In the simulation loop, the rotation tensor field is evaluated based on the actual coordinates $\mathbf{x}_i^t$ as described in section 3.1. Then, the linear system for the unknown new velocities $\mathbf{v}_i^{t+1}$ is solved. This system is derived by substituting Eqn.12 into Eqn.4 for implicit integration. Note that the k_{ij} are 3×3 sub-matrices of K containing entries K_{vw} with $3i - 2 \leq v \leq 3i$ and $3j - 2 \leq w \leq 3j$. Note also that we lump the mass matrix M in Eqn.4 to the vertices, i.e. replace it by its diagonal entries m_i. The positions of the vertices are then updated using the new velocities before going to the next time step.

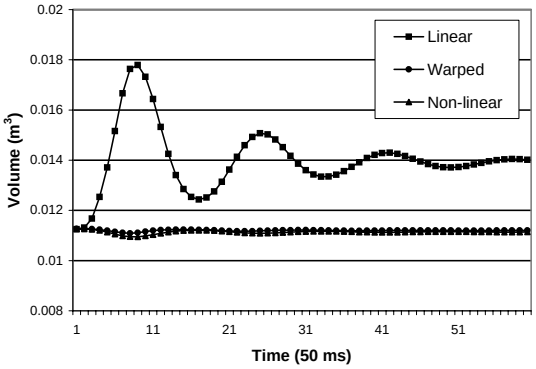


Figure 6: The volume of a bar that deformes under gravity simulated using linear, warped and non-linear stress measures.

4 Results

4.1 The Bars

To demonstrate the advantages of our approach, we compare it to a linear and a non-linear model. In all three cases, we use implicit Euler integration and lumped inertia and damping matrices. A Conjugate Gradients solver [Pozrikidis 1998] is used for Eqn. 4.

We animate a rectangular bar of $4 \times 4 \times 11$ vertices containing 450 tetrahedral elements. The block is fixed to a wall on one side and deforms under the influence of gravity (Fig. 7). In the linear case, the stiffness matrix of the object is evaluated once and used throughout the simulation to compute the internal elastic forces. In the warped stiffness case, we use the same constant stiffness matrix and warp it along a rotation field. This field is computed as shown in Fig. 5. In the non-linear case, a new stiffness matrix is computed at every time step as the Jacobian of the non-linear force function F based on Green's strain tensor. We use an elastic modulus E of $10^5 N/m^2$ and a Poisson ratio of 0.33, meaning the volume of the material should not change substantially during the simulation.

Fig. 6 depicts the volume of the block versus time. The linear model shows the typical growth artifact under deformation. As with the non-linear model, our method does not exhibit this problem. The time to compute one time step is 5 ms in the linear case, 6 ms for stiffness warping and 12 ms for the non-linear simulation. The simulated time step is 10 ms. The experiment shows that our approach is nearly as fast as the linear model but as accurate as the non-linear model in terms of volume conservation (try our applet at `graphics.lcs.mit.edu/simulation/warp/`).

To demonstrate the stability of stiffness warping, we repeated the simulation with a longer bar of $4 \times 4 \times 18$ vertices and 765 tetrahedra (Fig. 7). The linear and warped stiffness methods still yield stable simulations with a time step of 10 ms while the non-linear techniques diverges even with bars slightly longer than in the previous example.

4.2 A Simple Tube

The tube depicted in Fig. 8 is composed of a thousand tetrahedra and it is 50 cm x 13 cm in size. For its material we chose a density of 1 g/cm^3 and a Poisson Ratio of 0.33. The user interacts with the system by grabbing the tube at one vertex. This vertex is then attached to the mouse via a spring. In the first experiment, only the upper part of the tube is included in the simulation while the lower part remains fixed to the ground plane. When the entire model is animated, the user can pick it up. It bends due to inertial forces. The tube shows deformations and vibrations without the artifacts of

a linear model. When dropped from a diagonally oriented position 50 cm above the ground, the impact causes deformations that can lead to instabilities in the simulation. The following table shows the largest time step we were able to use before the system became instable. This value depends on the stiffness (Young's Modulus E) of the material.

$E\,[10^6 N/m^2]$	2.0	1.0	0.5	0.2	0.1
Warp	30 ms	20 ms	10 ms	10 ms	10 ms
Non-Linear	5 ms	5 ms	2 ms	1 ms	1 ms

As the results show, the simulation using the warped stiffness technique can be further accelerated by choosing larger time steps than in the non-linear case. Smaller elastic moduli cause larger deformations after the impact and smaller time steps need to be taken.

4.3 The Bunny

To generate the animation depicted in Fig. 9, we used a volumetric mesh of 5000 tetrahedra. The mesh is composed of a bone core and a layer of skin tetrahedra. Only the bunny's head, composed of 276 bone tetrahedra and 851 skin tetrahedra is animated. We treat all bone tetrahedra as one rigid body. This rigid skull can rotate about a fixed axis and is attached to the mouse via a spring. The skin tetrahedra follow the movement of the skull dynamically.

We use the deformation field of the tetrahedral mesh to animate a triangle surface mesh with higher resolution (5000 triangles). Every vertex in the surface mesh is associated with a tetrahedron in the volumetric mesh and uses its barycentric coordinate with respect to that tetrahedron to interpolate its position.

4.4 The Great Dane

As our last example, we animate the floppy skin of a Great Dane (Fig. 10). As in the bunny example, we simulate the bone core as a rigid body and let the skin layer follow its movements, but in this case, the entire model (i.e. 753 bone and 1244 skin tetrahedra) is animated. The elastic modulus E of the skin in the Dane's face is $10^3 N/m^2$ – much lower than in the previous examples – which makes the surface lag noticeably behind the skull movement. The visible surface mesh is formed with 5000 triangles, the vertices of which are interpolated using the underlying tetrahedral mesh.

5 Conclusions

In this paper, we have presented a new technique to animate deformable objects in real-time. By warping the constant stiffness matrix of the system used in linear approaches along a rotation field, we eliminate the visual artifacts while the simulator remains as stable and fast as a linear one, even for large rotational deformations. In contrast to a non-linear approach, the stiffness matrix needs only to be computed once. The same matrix can be used throughout the entire simulation for implicit integration, making the system fast and robust. We have also proposed a fast way of estimating a rotation field along which the stiffness matrix is warped at every time step.

Our examples show that stiffness warping makes possible real-time animation of detailed models in an interactive environment.

In the future, we would like to incorporate material fracture into our simulator. Stiffness warping works with a constant system matrix. This matrix changes when the structure of the underlying mesh changes. Fortunately, local changes in the mesh only cause local changes in the coefficients of the global stiffness matrix. Such updates can be done incrementally and will not slow down the simulation significantly.

References

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

COOK, R. D. 1981. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, NY.

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., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. *Graphics Interface*, 1–8.

GIBSON, S. F., AND MITRICH, B. 1997. *A survey of deformable models in computer graphics*. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA.

GOLUB, G. H., AND LOAN, C. F. V. 1996. *Matrix Computations, Third Edition*. The Johns Hopkins Univ. Pr., Baltimore and London.

HESTENES, D., AND SOBCZYK, G. 1984. *Cliffold Algebra to Geometric Calculus: A unified language for mathematics and physics*. D. Reidel, Dordrecht.

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.

LASENBY, J., FITZGERALD, W. J., DORAN, C. J. L., AND LASENBY, A. N. 1998. New geometric methods for computer vision. *Int. J. Comp. Vision 36(3)*, 191–213.

PICINBONO, G., DELINGETTE, H., AND AYACHE, N. 2000. Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery: MICCAI 2000* (Oct.), 643–652.

POZRIKIDIS, C. 1998. *Numerical Computation in Science and Engineering*. Oxford Univ. Press, NY.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically based models with rigid and deformable components. *IEEE Computer Graphics & Applications* (Nov.), 41–51.

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.

WITKIN, A., AND BARAFF, D. 1997. Physically based modeling: Principles and practice. *Siggraph Course Notes* (Aug.).

WITKIN, A., AND BARAFF, D. 1998. Large steps in cloth simulation. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 98, 43–54.

WU, X., DOWNES, M. S., GOKTEKIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Eurographics* (Sept.), 349–358.

ZHUANG, Y. 2000. *Real-time Simulation of Physically Realistic Global Deformation*. Ph. D. thesis of Univ. of California, CA.

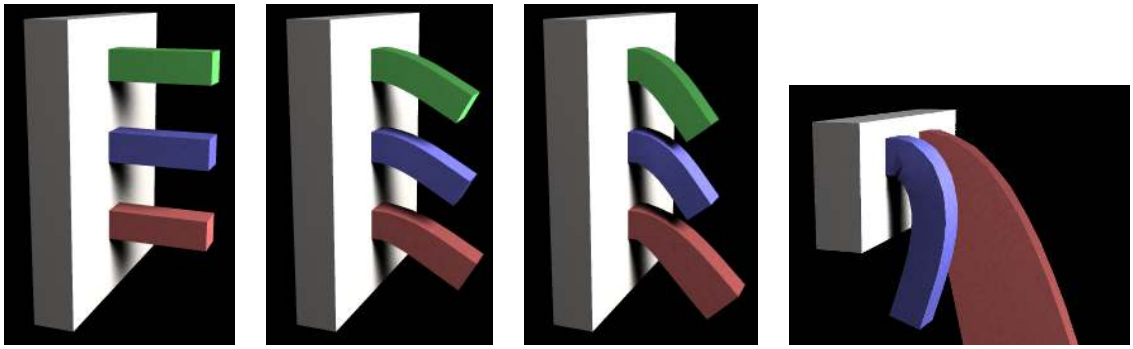
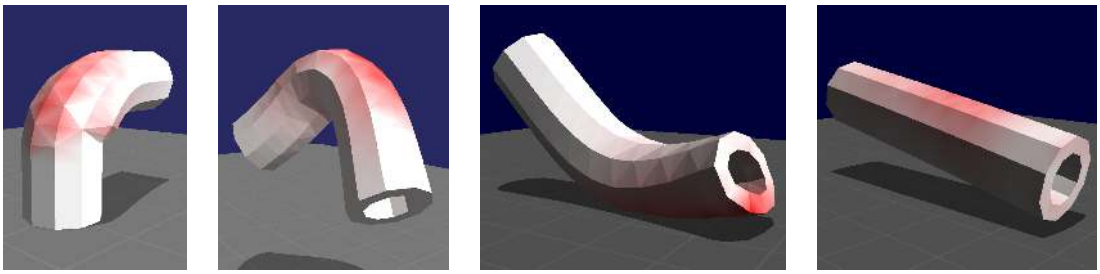


Figure 7: Three bars attached to a wall under the influence of gravity. They are simulated using non-linear (green), warped (blue) and linear (red) strain measures. Longer bars more noticeably show the artifacts with linear FEM.



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

Figure 8: A tube is bent under user-applied forces (a), inertial forces (b) and collision forces with low (c) and high (d) elasticity modulus.

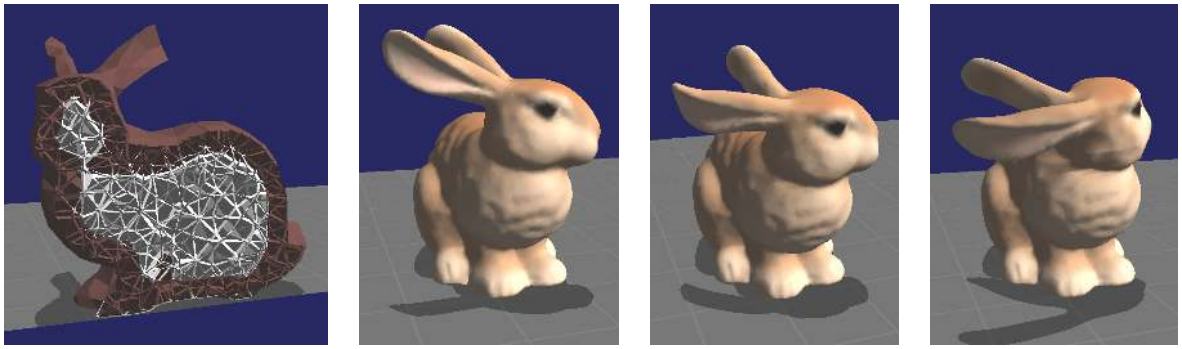


Figure 9: The bone core (white) is animated as a rigid body while the bunny's skin follows it dynamically.

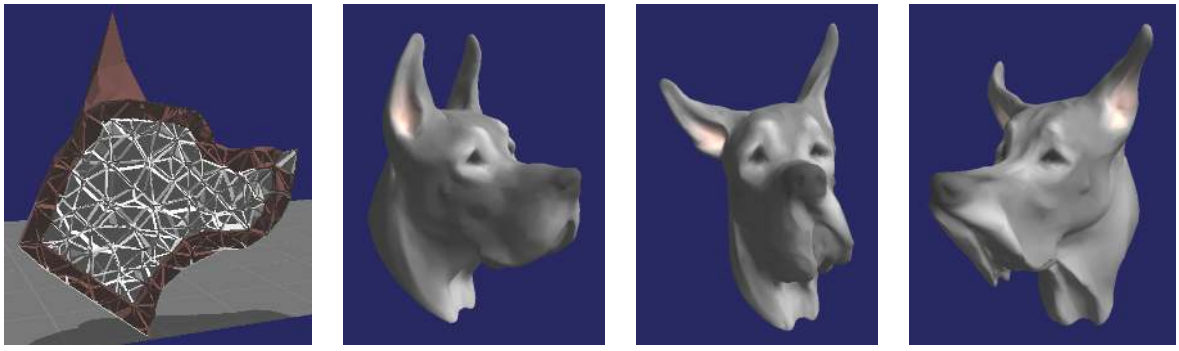


Figure 10: The great dane's skin has a low elastic modulus, which makes the surface lag noticeably behind the skull movement.