Simulation of Non-penetrating Elastic Bodies Using Distance Fields

Gentaro Hirota **Susan Fisher** **Ming C. Lin**

Abstract: *We present an efficient algorithm for simulation of non-penetrating flexible bodies with nonlinear elasticity. We use finite element methods to discretize the continuum model of non-rigid objects and the fast marching level set method to precompute a distance field for each undeformed body. As the objects deform, the distance fields are deformed accordingly to estimate penetration depth, allowing enforcement of non-penetration constraints between two colliding elastic bodies. This approach can automatically handle self-penetration and inter-penetration in a uniform manner. We combine quasi-viscous Newton's iteration and adaptive-stepsize incremental loading with a predictor-corrector scheme. Our numerical method is able to achieve both numerical stability and efficiency for our simulation. We demonstrate its effectiveness on a moderately complex animated scene.*

Keywords: Deformation, physically-based animation, numerical analysis.

1 Introduction

Due to recent advancements in physically-based modeling, simulation techniques have been increasingly used to improve the quality and efficiency in the generation of computer animation for major film productions [FM96, DKT98], medical simulation [KGC+96, Gib98] and computer games. These techniques produce animation directly from input objects, simulating natural motions and shape deformations based on mathematical models that specify the physical behavior of characters and complex structures.

Modeling deformation is a key component of physically-based animation, since many real-world objects are not rigid. Some examples include realistic motion generation of articulated characters with passive objects (such as clothing, footwear and other accessories), deformation of soft tissues and organs, and interaction among soft or elastic objects. Automatic, predictable and robust simulation of realistic deformation is one of the many challenges in computer animation and medical simulation.

When two flexible objects collide, they exert reaction forces on each other resulting in the deformation of both objects. Similarly when one flexible body self collides, it may deform and result in self-intersection. The reaction force is called contact force, and where the two surfaces touch is often called the contact surface. Simulating such events is non-trivial. It is known as the *contact problem* in computational mechanics, and has been actively investigated for decades [CL94]. The difficulty of this problem arises from unclear boundary conditions; neither the contact force nor the position of the contact surface is known a priori.

Previous work on deformable object animation uses physically-based methods [TPBF87, TF88, PW89, GTT89, WW90, BW92, CZ92, MT92], variational techniques [WW92, TQ94], as well as local and global deformations applied directly to the geometric models [Bar84, SP86, TQ94, RE99]. Earlier work in computer animation has focused mostly on the active simulation of primary characters [vFV90, RH91, Coh92, BPW93, HWBO95]. *Secondary motions*, i.e. motions of passive objects generated in response to environmental forces or to the movements of characters and other objects, add visual complexity to an animated scene and bring it to life [OHZ97]. Recently passive simulations of fire, gas, sand, fluids and cloth [SF95, CVT95, FM96, FM97, BW98, SOH99] have been gaining attention, as they add distinct realism to the animated scene.

Figure 1: *A snapshot of the animation automatically generated by our algorithm: Bulging effect around the neck of the snake due to its swallowing of an apple*

In this paper, we address the problem of simulation of nonlinear elastic bodies, such as soft tissue or synthetic materials like rubber or plastics. Linear approximations are often used to generate deformation of elastic bodies for computational efficiency. This is acceptable for some applications. However, large deformation is essential to create exaggeration effects in computer animation, and often bending of joints causes global deformation of tissues in medical simulation. In these situations, linear approximation can generate undesirable and unrealistic behavior. Methods such as spring-network systems are somewhat limited and are not applicable to modeling material properties like incompressibility.

In many cases where the movement of animated characters and articulated figures is relatively slow, simulating *static* behavior of passive elements is fast and sufficient. Each keyframe or motion sequence, generated by kinematics, inverse dynamics or other means, can provide positional constraints. Simulation can be used to generate the deformation of passive elements, such as the folding of skin or the bulging of muscles, to greatly enhance the realism of animation.

1.1 Main Results

We present an efficient approach for passive simulation of non-penetrating elastic bodies. The underlying geometric models are composed of polygonal meshes. Models consisting of implicit representations or parametric surfaces, such as NURBS, can be tessellated into polygonal meshes with bounded error.

Our technique is based on the nonlinear elasticity theory [Ogd84, Cia88, Tal94b] of continuum mechanics. We use finite element methods (FEM) [Tal94a] to model the non-rigid bodies. We employ the fast marching level set meth-

ods [OS88, Set99] to precompute the internal distance field of each *undeformed* model. When two flexible bodies come into contact and deform, the distance fields are likewise deformed to compute the estimated penetration depth between two deforming objects. This penetration measure is incorporated into a penalty-based formulation to enforce the non-penetration constraint between two elastic bodies. This enables efficient computation of the contact force and yields a versatile and robust contact resolution algorithm. Our algorithm efficiently computes the minimum energy equilibrium state of deformation by combining quasi-viscous Newton's iteration and adaptive-stepsize incremental loading with a predictor-corrector scheme [CL94]. We have successfully integrated these techniques to simulate collision response between two elastic bodies efficiently.

Specifically, our algorithm has the following characteristics:

- Both **self-collisions** and **contacts between soft objects** are handled in a uniform manner with our robust and efficient contact processing algorithm. This is achieved by first deforming the pre-computed distance fields to quickly estimate a penetration depth and thereby calculating penalty forces to resolve contacts.
- Material properties, including **nonlinear** elasticity and resistance to volume change, are taken into account to generate realistic behavior. Using finite element analysis, our numerical method can efficiently minimize the total energy in the system due to **large deformation** subject to the desired physical constraints.
- Due to our choice of discretization methods based on FEM, the simulated objects can be represented as polyhedra of **arbitrary shapes and topology**.
- **No prior assumption** or knowledge about the locations of contacts is required when using the resulting algorithm.

We demonstrate the effectiveness and potential of our approach on a moderately complex animated scene. Figure 1 shows a snapshot of the simulation results. The natural deformation of the snake's skin due to swallowing of a large apple has been automatically generated by our algorithm.

1.2 Organization

The rest of the paper is organized in the following manner. We briefly survey the state of the art on simulating deformation in section 2. In section 3, we give an overview of our algorithm. In section 4, we present our approach based on finite element methods for modeling elastic bodies. Section 5 describes the numerical methods used in our simulation. Section 6 presents our new contact determination method for deformable objects based on linear interpolation of precomputed distance fields and the resulting collision response. Section 7 describes the system implementation and demonstrates the effectiveness of our approach.

2 Previous Work

Modeling deformation has been studied in several fields, including computer graphics, geometric modeling, computer vision, computational mathematics, biomechanics, engineering and many others. An excellent survey of literature on deformable modeling in computer graphics can be found in [GM97]. In this section, we focus on physically-based modeling techniques and mechanics for modeling deformation of soft tissues or other highly nonlinear materials.

2.1 Modeling Deformation

Linear elastic models have been successfully used in real-time applications where only small deformations and strains occur [BNC96, JP99]. Mass-spring systems have been widely used to model deformation (e.g. [Mil88, LTW95, BW98]), due to their simplicity. However, they are notorious for their difficulty in determining the stiffness parameter and modeling incompressibility. Finite element methods (FEM) have been regarded as a versatile, effective and more accurate technique for discretization of continuum models. They are especially suitable for modeling nonlinear elasticity. Although they have been frequently employed in engineering applications, their use has been rather limited in computer graphics and animation (e.g. [CZ92, OH99]) due to their mathematical complexity and associated computational costs.

2.2 Contact Surfaces and Contact Forces

In some applications, the contact relationship can be predetermined. Donzelli [Don95] analyzed articular cartilage in 2D, where the "contactor" and "target" are pre-determined. The method is not applicable to general geometric configurations. Similar "master-slave" relationships have also been defined in other approaches [HGB87]. If deformation can be expressed as a function of time a priori, geometric algorithms for finding collision points between time-dependent surfaces can be found [VBZ90, SWF+93].

Once collisions have been detected, repulsive forces between particles are often used to prevent inter-penetration (e.g. [BHW94]). In this type of method, many particles on the boundary surfaces repulse each other to keep the surfaces from colliding. In a situation where the boundary surface significantly stretches, the gaps between particles widen with increasing chance of interpenetration. A high density of particles is required to prevent penetration. The "pinball" method [BN91, BY92] uses sphere trees to quickly reject the particles not intersecting the surface, but also requires a large number of spheres to achieve desired accuracy.

Precise contact modeling between flexible solids using implicit functions has been described in [Gas93, DG95]. Its applications, however, are limited to scenarios where objects can be represented as specific types of implicit functions.

2.3 Numerical Methods

Explicit integration techniques (e.g. [ZC99]) are often used in dynamic simulation of deformable objects. They are usually fast, but suffer from numerical stability problems. In addition, they are unsuitable for static analysis of highly nonlinear systems. Recently the advantages of implicit integration have been extensively discussed in [BW98, DSB99], which are kindred to the numerical method chosen by our algorithm. A more detailed discussion about our numerical method will be presented in section 5. We refer the readers to [KO88] for a complete, formal treatment of the numerical solutions of contact problems using FEM.

3 Algorithm Overview

In this section, we describe the problem statement and give an overview of our approach.

3.1 Problem Definition

Suppose we are given a set of elastic objects represented as polyhedra. Let $\mathbf{V}$ be the set of the vertices of these poly-

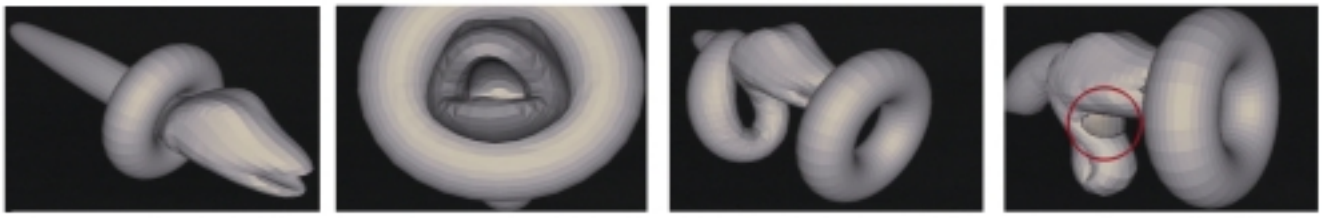

Figure 2: *(a)-(c) show a simulation sequence generated by our algorithm: (a) A snake retreating through a flexible torus. (b) A close-up look at the torus deforming as the snake slides through. (c) The snake coiling up without self-penetration. (d) Visually displeasing self-penetration of the snake without penetration avoidance. Its neck penetrates its body as the snake bends backward (highlighted by the red circle).*

hedra. Given the new positions for a subset of vertices $\mathbf{V}^c$ as positional constraints, the problem is to deform each object by computing the new positions of the remaining vertices $\{\mathbf{V} - \mathbf{V}^c\}$, taking into consideration material properties and external forces. Interpenetration between objects and self-intersections need to be avoided. Given the above constraints, the total energy stored in all objects must be minimized.

Examples of this problem can be found in computer animation and medical simulation. For instance, assume that the motions of the skeleton of an articulated figure are given (by dynamics controller or inverse kinematics) as positional constraints. We would like to automatically generate the static behavior of the deformable soft tissues, skin and muscles, as the skeleton figure moves.

Another example is shown in Figure 2(a)-(c). Our algorithm simulates a snake retreating through a flexible torus ring and coiling itself up with the non-penetration constraints, given only a few simple positional constraints provided by the user. In comparison, Figure 2(d) shows visually disturbing self-penetration of the snake in the same simulation without the penetration avoidance mechanism.

3.2 Outline of Our Approach

We solve the problem described above by using the following steps:

1. Given the input models, construct a tetrahedral element mesh as a discretized representation for each object (section 4).
2. Generate an internal distance field for each input object using the fast marching level set method (section 6.2).
3. Apply finite element analysis (section 4):
   (a) Estimate the penetration depth based on the deformed distance fields for penetration avoidance (section 6.1).
   (b) Minimize the total energy due to deformation, taking into account all material properties and external forces, using our synthesized numerical method (section 5).

Figure 3 shows the flow of our system.

3.3 Notation Convention

In the next few sections, we will describe each step of our algorithm in greater detail. Here we define some conventions for the notations that we will use throughout the paper. A lower-case, bold-face letter, such as $\mathbf{p}$, denotes a position in the 3D Euclidean space. A upper-case, bold-face letter, such as $\mathbf{A}$, denotes a matrix or a displacement vector in R^3.

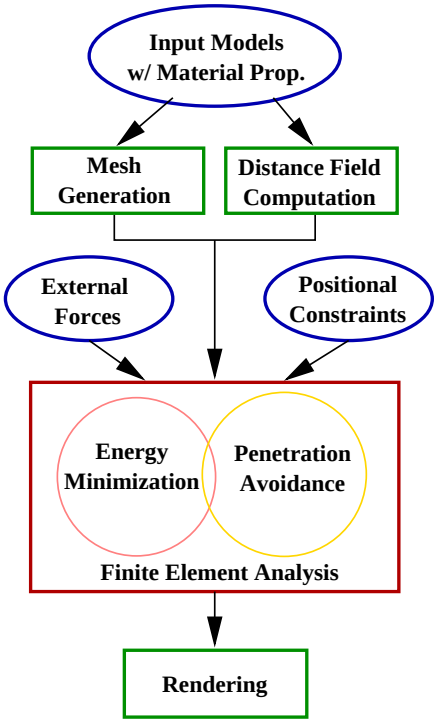

Figure 3: *A system overview showing various components of our algorithm*

4 Finite Element Methods

In this section, we describe the discretization method we use, namely finite element analysis, to model deformation based on the nonlinear elasticity theory. We reformulate the problem of simulating deformable objects as a constrained minimization problem using Constitutive Law.

4.1 Discretization Methods

Deformation induces movement of every particle within an object. It can be modeled as a mapping of the positions of all particles in the original object to those in the deformed body. Each point $\mathbf{p}$ is moved by the deformation function $\phi(\cdot)$:

$$\mathbf{p} \rightarrow \phi(t, \mathbf{p})$$

where $\mathbf{p}$ represents the original position, and $\phi(t, \mathbf{p})$ represents the position at time t. We limit the discussion to the

static analysis, hence t is omitted:

$$\mathbf{p} \rightarrow \phi(\mathbf{p})$$

Simulating deformation is in fact finding the $\phi(\cdot)$ that satisfies the laws of physics. Since there are an infinite number of particles, $\phi(\cdot)$ has infinite degrees of freedom. In order to model a material's behavior using computer simulation, some type of *discretization* method must be used. For simulation of deformable bodies, spring networks, the finite difference method (FDM), the boundary element method (BEM), and the finite element method (FEM) have all been used for discretization.

A spring network approximates the laws of physics by pairwise relationship between node points. It is very difficult to map continuum mechanics to such pairwise relationships correctly. Namely, it is impossible to describe a potential energy associated with volume change.

In the FDM, the independent approximations happen at a finite number of sampled points. A FDM usually requires "regular" structures for mesh topology, which constrains our choices of geometric representations.

The BEM uses node points sampled only on boundary surfaces. The use of the BEM is limited to linear differential equations, hence cannot be used to simulate nonlinear elastic bodies.

The FEM uses a piecewise approximation of the deformation function $\phi(\cdot)$. Each "piece" is called an element, which is defined by several node points. The elements constitute a mesh. Since the FEMs pose relatively small restrictions on the mesh topology, they are suitable for representing a variety of shapes and topology.

4.2 Tetrahedral Elements

Our algorithm uses a FEM with 4-node tetrahedral elements with linear shape functions. We have chosen this element because:

- In a mesh composed of 4-node tetrahedral elements, each node has a relatively small number of neighbors. This results in fewer non-zero elements in the stiffness matrix and less expensive computation. Higher order elements such as 6-node hexahedral elements, on the other hand, produce denser stiffness matrices.
- 4-node tetrahedral elements simplify the integration of the derivatives of the potential energy. This integration is essential for computing the stiffness matrix. Precise integrations for higher order elements are expensive, and usually require numerical integration techniques such as Gauss quadrature.
- A tetrahedron does not self-penetrate. As a result the penetration problem is reduced to pairwise element-element problem. A higher order element can deform and result in self-intersection, which makes the penetration problem much more complicated.

$\phi(\cdot)$ maps a point in a tetrahedral element at $\mathbf{p} = [x, y, z]^T$ to a new position $\phi(\mathbf{p})$. As shown in Fig. 4, by definition, $\phi(\cdot)$ moves four nodes of an element from their original positions

$$\mathbf{n_i} = [n_{ix}, n_{iy}, n_{iz}]^T, 1 \leq i \leq 4,$$

to the new positions

$$\tilde{\mathbf{n}}_\mathbf{i} = [\tilde{n}_{ix}, \tilde{n}_{iy}, \tilde{n}_{iz}]^T, 1 \leq i \leq 4.$$

The displacements of the four nodes due to the deformation is

$$\begin{aligned} \mathbf{U_i} &= [U_{ix}, U_{iy}, U_{iz}]^T \\ &= [\tilde{n}_{ix} - n_{ix}, \tilde{n}_{iy} - n_{iy}, \tilde{n}_{iz} - n_{iz}]^T, 1 \leq i \leq 4. \end{aligned}$$

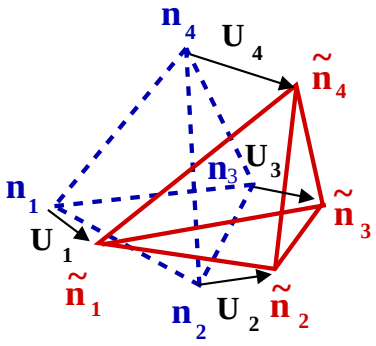

Figure 4: *$\phi(\cdot)$ maps four nodes of a tetrahedral element, $\mathbf{n_1}, ..., \mathbf{n_4}$ to their new position at $\tilde{\mathbf{n}}_1, ..., \tilde{\mathbf{n}}_4$. $\mathbf{U_1}, ..., \mathbf{U_4}$ are the corresponding displacement vectors.*

Clearly this deformation is an affine transformation of the form:

$$\phi(\mathbf{p}) = \mathbf{F}\mathbf{p} + \mathbf{T}$$

where $\mathbf{F}$ is a 3×3 matrix and $\mathbf{T}$ is 3×1 column vector representing translational displacement. $\tilde{\mathbf{n}}_\mathbf{i}$ and $\mathbf{n_i}$ satisfy the linear system:

$$\begin{aligned} \tilde{\mathbf{n}}_1 &= \mathbf{F}\mathbf{n}_1 + \mathbf{T} \\ \tilde{\mathbf{n}}_2 &= \mathbf{F}\mathbf{n}_2 + \mathbf{T} \\ \tilde{\mathbf{n}}_3 &= \mathbf{F}\mathbf{n}_3 + \mathbf{T} \\ \tilde{\mathbf{n}}_4 &= \mathbf{F}\mathbf{n}_4 + \mathbf{T} \end{aligned}$$

Since $\mathbf{T}$, representing the translational components, has no effect on the elastic energy, it is omitted from the rest of derivation. By solving the linear system, we have

$$\mathbf{F} = \mathbf{B}\mathbf{A}^{-1} = (\mathbf{A} + \mathbf{H})\mathbf{A}^{-1} = \mathbf{H}\mathbf{A}^{-1} + \mathbf{I}$$

where $\mathbf{I}$ is a 3×3 identity matrix and

$$\begin{aligned} \mathbf{A} &= [\mathbf{n}_2 - \mathbf{n}_1, \mathbf{n}_3 - \mathbf{n}_1, \mathbf{n}_4 - \mathbf{n}_1] \\ \mathbf{B} &= [\tilde{\mathbf{n}}_2 - \tilde{\mathbf{n}}_1, \tilde{\mathbf{n}}_3 - \tilde{\mathbf{n}}_1, \tilde{\mathbf{n}}_4 - \tilde{\mathbf{n}}_1] \\ \mathbf{H} &= [\mathbf{U}_2 - \mathbf{U}_1, \mathbf{U}_3 - \mathbf{U}_1, \mathbf{U}_4 - \mathbf{U}_1] \end{aligned}$$

with $\mathbf{n_i} - \mathbf{n_j}$ representing vector differences. $\mathbf{F}$ is known as the deformation gradient [CL94]. The right Cauchy-Green tensor, $\mathbf{C} = \mathbf{F}^\mathbf{T}\mathbf{F}$, is often used to characterize deformation, and is insensitive to rigid motions.

4.3 Constitutive Law & Energy Minimization

Given the basics of FEM, we reformulate the problem of simulating deformable objects as a constrained minimization problem using Constitutive Law in this section.

Since the use of 4-node tetrahedral elements simplifies the energy integration, we use an energy minimization scheme to derive the equilibrium equations. This greatly simplifies the derivation. The same result is obtained by the Galerkin method, which is commonly used in standard FEM textbooks [CL94].

A constitutive law defines the relationship between the deformation gradient $\mathbf{F}$ and the elastic energy stored per unit volume for a specific material. The simplest and most commonly used is the Saint-Venant-Kirchhoff material [CL94]. It is characterized by the energy

$$W_{svk}(\mathbf{F}) = \frac{\lambda}{2}\ (tr(\mathbf{E}))^2 + \mu\ tr(\mathbf{E}^2).$$

where $\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$, λ and μ are *Lamè* constants, and $tr(\mathbf{E})$ is the trace of the matrix $\mathbf{E}$.

This material has a weak characteristic to resist volume change, but can be compressed infinitely (i.e. zero-volume) without raising its energy to infinity. This also means that negative compression rates will be allowed under high stress. Such materials exhibit unrealistic deformations. Avoiding infinite and negative compression is also important to proper computation of penetration penalty forces.

Therefore, we use the following constitutive law [Cia88] with a slightly more complex energy function:

$$W_{cg}(\mathbf{F}) = C_1(I_1-3)+C_2(I_2-3)+a(I_3-1)-(C_1+2C_2+a)ln(I_3)$$

where

$$\begin{aligned} I_1 &= tr(\mathbf{C}) \\ I_2 &= \frac{1}{2}tr(\mathbf{C})^2 - tr(\mathbf{C}^2) \\ I_3 &= det(\mathbf{C}) \end{aligned}$$

and, C_1, C_2, C_3 and a are material dependent constants. The energy goes to infinity as the material is compressed infinitely.

$\mathbf{F}$ and W_{cg} are constant in a tetrahedral element. The integration of the energy stored in an element is expressed as:

$$W_{tetra}(\mathbf{U_{tet}}) = \int_V W_{cg}\ d\mathbf{p} = V\ W_{cg}$$

where $\mathbf{U_{tet}}$ is a vector that contains all the displacements of nodes and V is the volume of the tetrahedron.

$\partial W_{tetra}(\mathbf{U_{tet}})/\partial \mathbf{U_{tet}}$ represents the elastic forces acting on nodes. Note that $\partial W_{tetra}(\mathbf{U_{tet}})/\partial \mathbf{U_{tet}}$ is a nonlinear function of $\mathbf{U_{tet}}$. This implies that the force acting on each node is a nonlinear function of the displacements. If it were a linear function, the solution of the following minimization problem could be solved by using a linear system solution. But it is well known that such a linear function cannot retain essential properties of finite elasticity [Ogd84, Tal94b], which is crucial for simulating realistic behavior of deformable objects.

The total elastic energy $W_{elastic}$ is a function of the displacements of all nodes. It is obtained as the summation of W_{tetra} for all elements. The displacements of nodes can be decomposed into two parts, $\mathbf{U_{fix}}$ and $\mathbf{U_{free}}$. $\mathbf{U_{fix}}$ represents constant displacements corresponding to predetermined vertex coordinates. $\mathbf{U_{free}}$ represents the displacements of "free" nodes. Thus we denote the total elastic energy as $W_{elastic}(\mathbf{U_{fix}}; \mathbf{U_{free}})$. $\mathbf{U_{fix}}$ serves as the boundary condition. If $\mathbf{U_{fix}}$ is the only boundary condition and the object is in a stationary and stable configuration, $\mathbf{U_{free}}$ locally minimizes $W_{elastic}(\mathbf{U_{fix}}; \mathbf{U_{free}})$.

To incorporate the penetration avoidance constraint into the same energy minimization scheme, an energy function W_{penet} representing the amount of "penetration" between polyhedra is added. Together with W_{ext}, the energy due to external forces (e.g. the potential energy for gravity), the total energy W in the system is:

$$\begin{aligned} W(\mathbf{U_{fix}}; \mathbf{U_{free}}) &= W_{elastic}(\mathbf{U_{fix}}; \mathbf{U_{free}}) + \\ & \quad W_{penet}(\mathbf{U_{fix}}; \mathbf{U_{free}}) + W_{ext}(\mathbf{U_{fix}}; \mathbf{U_{free}}) \end{aligned}$$

In the discretized domain, all forces can be considered as point forces acting on nodes. The total forces acting on free nodes are $\partial W(\mathbf{U_{fix}}; \mathbf{U_{free}})/\partial \mathbf{U_{free}}$. At the equilibrium, this force must vanish. That is,

$$\frac{\partial W(\mathbf{U_{fix}}; \mathbf{U_{free}})}{\partial \mathbf{U_{free}}} = \mathbf{0}$$

The solution of the minimization problem below satisfies this equilibrium condition:

$$\text{Find } \mathbf{U_{free}} \text{ s.t. it minimizes } W(\mathbf{U_{fix}}; \mathbf{U_{free}})$$

The details of our numerical method are presented in the next section.

5 Numerical Methods for Minimization

In addition to the non-linear nature of elastic materials, penetration introduces discontinuity to the deformation energy function. Therefore, a relatively robust numerical method must be used. In this section, we present a relatively stable numerical method by combining quasi-viscous Newton's iteration and adaptive-stepsize incremental loading with Euler and two-point predictors.

5.1 Basic Newton's Iteration

Newton's method finds the minimum of the deformation energy function by repeatedly approximating the total energy W with quadratic functions. We use the following approximation:

$$\begin{aligned} &W(\mathbf{U_{fix}}; \mathbf{U_{free}} + \Delta\mathbf{U_{free}}) \approx W(\mathbf{U_{fix}}; \mathbf{U_{free}}) + \\ &\quad \Delta\mathbf{U_{free}}^T\ \mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}}) + \Delta\mathbf{U_{free}}^T\ \mathbf{K}(\mathbf{U_{fix}}; \mathbf{U_{free}})\ \Delta\mathbf{U_{free}} \end{aligned}$$

where

$$\begin{aligned} \mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}}) &= \frac{\partial W}{\partial \mathbf{U_{free}}}(\mathbf{U_{fix}}; \mathbf{U_{free}}) \\ \mathbf{K}(\mathbf{U_{fix}}; \mathbf{U_{free}}) &= \frac{\partial^2 W}{\partial \mathbf{U_{free}}^2}(\mathbf{U_{fix}}; \mathbf{U_{free}}) \end{aligned}$$

Newton's iteration is given as

initialize
  $\mathbf{U_{free}} = \mathbf{U_{free}}$'s initial guess
repeat
  $\Delta\mathbf{U_{free}} = \mathbf{K}(\mathbf{U_{fix}}; \mathbf{U_{free}})^{-1}(-\mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}}))$
  $\mathbf{U_{free}} = \mathbf{U_{free}} + \Delta\mathbf{U_{free}}$
until $R_i < \epsilon, \forall i$, where R_i are entries of $\mathbf{R}$

This is the same as finding an solution of the equilibrium equation:

$$-\mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}}) = \mathbf{0}$$

Using linear approximation, we have

$$\begin{aligned} &-\mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}} + \Delta\mathbf{U_{free}}) \approx \\ &\qquad -\mathbf{R}(\mathbf{U_{fix}}; \mathbf{U_{free}}) - \mathbf{K}(\mathbf{U_{fix}}; \mathbf{U_{free}})\ \Delta\mathbf{U_{free}} \end{aligned}$$

If the above iteration converges and $\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ is positive definite, the algorithm finds a local minimum. The treatment of indefinite matrices are discussed in section 5.3.

Newton's method requires the expensive computation of the tangent or stiffness matrix, $\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$. But once it is computed, it provides a better global idea about the shape of $W(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ than just the gradient (i.e. the residual $\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$). Gradient descent methods that rely only on gradients require far more iterations than Newton's methods. The computations of $\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ and $\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ both require collision checking to perform penetration avoidance. Since collision detection between deformable bodies is an expensive process, the computational costs for $\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ and $\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$ are nearly the same. Hence, the expensive computation of the stiffness matrix pays off in the long run.

In dynamic simulation, implicit integration methods take advantage of stiffness matrices, whereas explicit integration methods use only residuals. We can obtain the minimum energy point by explicitly integrating a viscous system to the point where the system loses all kinetic energy and converges to a stationary point. But, the stiffness of common materials such as human tissue or a block of rubber may be more a dominating factor than inertia in determining the stepsize. Thus, for many realistic applications, a very small time step must be chosen for integration.

5.2 Incremental Loading with Euler and 2-Point Predictors

To handle large values of $\mathbf{U}_{\text{fix}}$, incremental loading is introduced. This method successively solves the problem:

$$\text{Find } \mathbf{U}_{\text{free}} \text{ s.t. it minimizes } W(\lambda\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$$

where λ is a scalar incremented from 0 to 1. A solution curve is obtained by plotting $\mathbf{U}_{\text{free}}$ against λ, shown in blue in Figure 5.

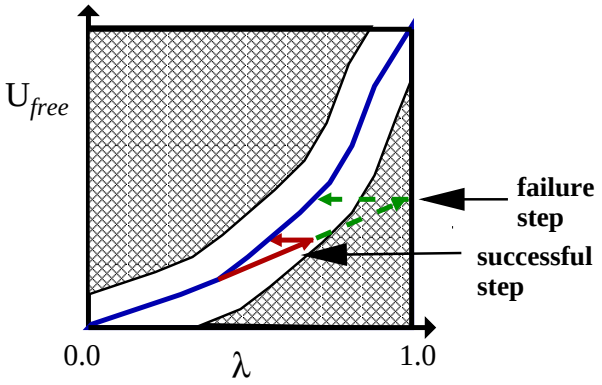

Figure 5: *Tracing solution curve*

For each incremented λ, a new initial guess of $\mathbf{U}_{\text{free}}$ is estimated (predictor) and Newton's iteration finds the solution (corrector).

Since $W(\cdot)$ is highly nonlinear, Newton's iteration converges only if the initial guess is close to the solution. If an initial guess lies in the shaded area as shown in Figure 5, convergence cannot be achieved.

Euler and two-point predictors are used to compute the initial guess of Newton's iteration. An Euler predictor estimates the tangent vector of the solution curve and extrapolates the new initial guess on the tangent line. This involves solving a linear system. If the linear system cannot be solved, then a two-point predictor is used. The two-point predictor extrapolates the two previous solutions to make an initial guess.

The predictors determine the direction in $\mathbf{U}_{\text{free}}$ space. The magnitude of a step is determined by the stepsize of λ. If the stepsize is too large, Newton's iteration may not converge. On the other hand, too small a stepsize degrades the performance. Therefore, the stepsize must be adjusted according to the current situation. If Newton's iteration fails to converge, it backtracks to the previous step and the stepsize of λ is decreased. If the iteration was successful, the stepsize is slightly increased. Thus, the stepsize is automatically adjusted for optimized performance.

We also implemented the arc-length continuation method [Tal94b], which is known to be a more sophisticated technique than incremental loading. In this method, Newton's iteration searches for the solution on the arc that is at a certain distance from the previous solution. This method is superior in the sense that it can overcome limit points. However, in our experience, the performance of this method is limited, because of its ability to decrement λ. It may be trapped in a loop in the solution curve, or return back to the starting point.

5.3 Newton's Iteration with Viscosity

The methods described above including arc-length continuation are all vulnerable to ill-conditioned and indefinite stiffness matrices. Viscosity is introduced to alleviate this problem.

Imagine the entire mesh immersed in viscous fluid. The tetrahedral elements can move through the fluid without any resistance, but the nodes encounter drag forces due to the friction between the nodes and the fluid. The drag force is:

$$-\mathbf{D}\ \dot{\mathbf{U}}_{\text{free}}$$

where

$$\mathbf{D} = diag[d1, d1, d1, d2, d2, d2, \ldots, dn, dn, dn]$$

$\dot{\mathbf{U}}_{\text{free}}$ is the "velocity" vector of nodes and $\mathbf{D}$ is the diagonal matrix that contains drag force scalars. d_i is the drag force scalar for node i, and proportional to the total volume of the tetrahedra surrounding the node. This ensures that the drag forces have similar magnitude as the elastic forces. This scalar value can be viewed as the radius of the node. The size of the node is set to the average size of elements to which the node belongs. Larger nodes experience more drag forces from the surrounding fluid.

The resulting equilibrium equation in Newton's iteration is

$$-\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}+\Delta\mathbf{U}_{\text{free}}) - \mathbf{D}\ \dot{\mathbf{U}}_{\text{free}} = \mathbf{0}$$

Our goal is mainly to stabilize Newtons' iteration. Hence the "current" velocity is set to zero, and the "next" velocity, $\dot{\mathbf{U}}_{\text{free}}$, is set to be $\Delta\mathbf{U}_{\text{free}}/\Delta t$ where Δt is the time step, which can be set to 1. Thus

$$-\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}+\Delta\mathbf{U}_{\text{free}}) - \mathbf{D}\ \Delta\mathbf{U}_{\text{free}} = \mathbf{0}$$

Using linear approximation, it simplifies to

$$\begin{aligned}
-\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}) - \mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})\Delta\mathbf{U}_{\text{free}} - \mathbf{D}\ \Delta\mathbf{U}_{\text{free}} &= 0 \\
-\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}) - (\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}) + \mathbf{D})\Delta\mathbf{U}_{\text{free}} &= 0 \\
\Delta\mathbf{U}_{\text{free}} = (\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}) + \mathbf{D})^{-1}\ (-\mathbf{R}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}}))&
\end{aligned}$$

As a result, we simply add positive values to the diagonal elements of the stiffness matrix $\mathbf{K}(\mathbf{U}_{\text{fix}};\mathbf{U}_{\text{free}})$. Since the

diagonal elements of $\mathbf{K}(\mathbf{U_{fix}};\mathbf{U_{free}})$ are usually the largest positive values in a row, with a large enough value of $\mathbf{D}$, $\mathbf{K}(\mathbf{U_{fix}};\mathbf{U_{free}})$ remains positive definite, preventing the algorithm from climbing energy hills. This also decreases the condition number of $\mathbf{K}(\mathbf{U_{fix}};\mathbf{U_{free}})$ and helps in solving linear systems.

6 Resolving Penetration

In section 4, we described the finite element analysis framework and how the static analysis of deformation can be solved using minimization techniques with the numerical methods described in section 5. In this section, we address the issue of formulating the non-penetration constraint, so to include it in the optimization process. We present an efficient contact processing technique that can handle both self-collision and inter-penetration between objects in a uniform manner.

Ideally, no two tetrahedral elements should share the same space. This is the non-penetration constraint. The non-penetration constraint can be imposed using techniques such as constrained optimization techniques or penalty-based methods. For example, in an augmented Lagrangian method, each Lagrangian multiplier corresponding to a contact relationship is updated in an iterative manner [Pow69]. The difficulty lies in the integration of the update loop with Newton's iteration that changes node positions and hence contact relationships. Instead of treating it as a hard constraint, we used penalty methods. Thus, a small amount of penetration is allowed. In practice, sufficiently large penalty constant is good enough to prevent visually displeasing penetration, unless the simulated objects are very thin.

6.1 Penetration Potential Energy

When using the penalty based method, we need to first define a penetration potential energy $W_{penet}(\cdot)$ that measures the amount of intersection between two polyhedra or the degree of self-intersection of a polyhedron, and thereby finds an efficient method to compute it, its first and second derivatives.

6.1.1 Defining the Extent of Intersection

There are several known methods to define the extent of intersection. The node-to-node method is the simplest way to compute $W_{penet}(\cdot)$. This method computes $W_{penet}(\cdot)$ as a function of the distances between sampled points on the boundary of each objects. The repulsive particle method mentioned in section 2 belongs to this category. The drawback of this method is that, once a node penetrates boundary polygons, the repulsive force flips its direction, and induces further penetration. Such penetration often occurs in intermediate steps of the aggressive Newton's iterations. Furthermore, once a node is inside a tetrahedral element, it is no longer clear which boundary polygon the node has actually penetrated.

A more accurate approach is to compute the penetration depth, commonly defined as the minimum translational distance required to separate two intersecting objects. No general and efficient algorithm for computing penetration depth between two non-convex objects is known. In fact, an $O(n^6)$ time bound can be obtained for computing the Minkowski sum of two non-convex polyhedra to find the minimum penetration depth in 3D [DHKS93].

The most complicated yet accurate method is to use the intersection volume. Using this method, $W_{penet}(\cdot)$ is defined based on the volume of intersection between two penetrating polyhedra. Since polyhedra deform as simulation steps proceed, it is difficult to create and reuse previous data from the original model. Furthermore, it is susceptible to accuracy problems and degenerate contact configurations. So, efficient computation of the intersection volume is rather difficult to achieve.

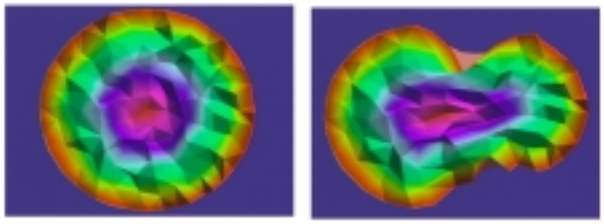

Figure 6: *(a) The distance field of a sphere. (b) The distance field of a deformed sphere computed using linear interpolation of the precomputed distance field.*

6.1.2 Estimating Penetration Depth

We have chosen a method that provides a balance between the two extremes by computing an *approximate* penetration depth between deformable objects. With our method, $W_{penet}(\cdot)$ is defined as a function of distances between boundary nodes and boundary polygons that the nodes penetrate. We define

$$W_{penet}(\cdot) = k * d^2 \tag{1}$$

where d is the minimum distance from a boundary node to the intruded boundary and k is a penalty constant.

The convergence rate of Newton's iteration depends on the smoothness of the energy function. Therefore, a quadratic function was chosen. Accurate computation of the penetration depth d is very expensive, and d needs to be evaluated at every Newton's iteration. Our algorithm estimates the computation of the penetration depth d by replacing it with the linear interpolation $\tilde{d}$ of pre-computed distance values:

$$\tilde{d} = u_1\ d_1 + u_2\ d_2 + u_3\ d_3 + (1 - u_1 - u_2 - u_3)\ d_4 \tag{2}$$

where d_1, d_2, d_3 and d_4 are distance values at the four nodes of each tetrahedral element. These distance values are sampled from the distance field generated by the fast marching level set method to be described in the next section. u_1, u_2 and u_3 are the interpolation parameters, and $0 \leq u_i \leq 1$, $1 \leq i \leq 3$.

Once an accurate value of distance is assigned to each node, no matter how the mesh is deformed, the value of $\tilde{d}$ is quickly computed at any point inside the object. Figure 6 shows an example where the distance field of a sphere is quickly re-computed as the sphere deforms.

This approximated distance field shares a few properties with the exact distance field. Some of these properties are essential for proper computation of penalty forces and their derivatives:

1. It vanishes on the boundary polygons.
2. It is twice differentiable inside the elements and C^0 continuous everywhere.
3. By following its negative gradient, an internal point can reach the boundary within short distance, which helps the Newton's iteration with high viscosity to quickly converge to the solution.

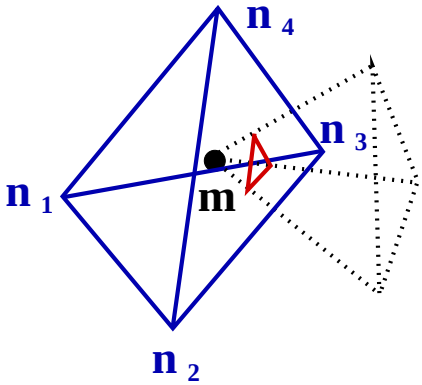

Figure 7: *A node* $\mathbf{m}$ *penetrates into another tetrahedral element. The distance between* $\mathbf{m}$ *and the red triangle is the penetration depth.*

A simple space subdivision method is used to find each instance where a boundary node from one element penetrates another element. Suppose a boundary node $\mathbf{m}$ is within an element with nodes $\mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}$ and $\mathbf{n_4}$ as shown in Figure 7. $\mathbf{m}$ can be written in terms of linear interpolation of $\mathbf{n_1}, \ldots, \mathbf{n_4}$:

$$\mathbf{m} = u_1\ \mathbf{n_1} + u_2\ \mathbf{n_2} + u_3\ \mathbf{n_3} + (1 - u_1 - u_2 - u_3)\ \mathbf{n_4} \quad (3)$$

$\tilde{d}$ at $\mathbf{m}$ is obtained by solving Eqn. 2 and Eqn. 3:

$$\tilde{d} = [d_1 - d_4, d_2 - d_4, d_3 - d_4]\ G^{-1}\ [\mathbf{m} - \mathbf{n_4}] + d_4 \quad (4)$$

where

$$\mathbf{G} = [\mathbf{n_1} - \mathbf{n_4}, \mathbf{n_2} - \mathbf{n_4}, \mathbf{n_3} - \mathbf{n_4}]$$

$W_{penet}(\cdot)$ is computed by using $\tilde{d}$ instead of d in Eqn. 1. The terms $\partial W_{penet}/\partial \mathbf{U_{free}}$ and $\partial^2 W_{penet}/\partial {\mathbf{U_{free}}}^2$ are obtained as the corresponding components of the first and second derivatives of $W_{penet}(\cdot)$ with respect to $\mathbf{m}$ and $\mathbf{n_i}$, $1 \leq i \leq 4$. These derivatives can be easily computed by applying chain rules to Eqn. 1 and Eqn. 4.

The estimated $\tilde{d}$ is not differentiable on the boundary of the elements, which causes abrupt changes in the residual and the stiffness matrix across the boundary and sometimes confuses Newton's iteration. But, with proper viscosity, the simulation progresses past the critical point.

This algorithm is insensitive to which object (or connected mesh) the nodes $\mathbf{m}$ and $\mathbf{n}$ belong to. Therefore, self-intersections and intersections between two objects are treated in a uniform manner. It is also robust enough to recover from deep penetrations. In the next section, we will briefly describe the method we use for pre-computing the distance field.

6.2 Distance Field Computation

Computing the minimum geodesic distance from a point to a surface is a well known complex problem [KAB95]. Osher and Sethian [OS88, Set99], introduced a new perspective on this problem by using a partial differential method to perform curve evolution. The fast marching level-set method avoids the problems that traditional methods incurred, such as the problems with corners and cusps, and changing topologies. The method has a basis in gas dynamics and hyperbolic conservation laws.

The algorithm first constructs a band of points around the initial surface. All of the initial points, with the exception of the innermost band, are labeled as "Alive". Once an Alive point is computed, it may not be recomputed. The remaining points of this initial band are labeled as "Narrow Band" points. These points form the surface that we will evolve. The remaining, uninitialized points are termed "Far Away".

At each step, the gridpoint with the minimum distance value is extracted from the Narrow Band. For efficiency reasons, the Narrow Band is implemented as a minimum heap structure. Each extraction of the minimum grid point from the Narrow Band is an $O(1)$ operation. This point is moved to the Alive points and recomputed by solving for T in the following equation:

$$M = (\max(D^{-x}T + \frac{\Delta x}{2}D^{-x-x}T, D^{+x}T + \frac{\Delta x}{2}D^{+x+x}T, 0))^2$$

where the finite differences are given by

$$\begin{aligned} D^{+x} &= \frac{T_{i+1} - T}{\Delta x} \\ D^{-x} &= \frac{T - T_{i-1}}{\Delta x} \\ D^{+x+x} &= \frac{T_{i+2} - 2T_{i+1} + T}{2\Delta x} \\ D^{-x-x} &= \frac{T - 2T_{i-1} + T_{i-2}}{2\Delta x} \end{aligned}$$

Similarly, we let

$$N = (\max(D^{-y}T + \frac{\Delta y}{2}D^{-y-y}T, D^{+y}T + \frac{\Delta y}{2}D^{+y+y}T, 0))^2$$
$$O = (\max(D^{-z}T + \frac{\Delta z}{2}D^{-z-z}T, D^{+z}T + \frac{\Delta z}{2}D^{+z+z}T, 0))^2$$

Let

$$\frac{1}{F_{ijk}} = \sqrt{M + N + O}$$

The term F_{ijk} represents the speed of the propagating front. Because we wish to find the distance from each point to the surface, this value is uniform (constant) in this implementation.

All Narrow Band and Far Away neighbors are also recomputed, and the Far Away neighbors are moved into the Narrow Band. The equations use a second order scheme whenever possible to produce higher accuracy. That is, both T_{i+2} and T_{i+1} must be Alive in order to compute D^{+x+x}, D^{+y+y} or D^{+z+z}, where $T_{i+2} > T_{i+1}$. The choice of when to use the second order scheme simply depends on whether two known (Alive), monotonically increasing values exist as neighbors of the test point. If not, then the first order scheme is used.

For details on the accuracy and robustness of this algorithm, see [OS88, Set99].

7 Results and Discussion

We have implemented the algorithm described in this paper and have successfully integrated it into a moderately complex animation (shown in the accompanying videotape). We used Maya developed by Alias|Wavefront to generate the models used in our animation sequences. We used a public domain mesh generation package, SolidMesh [MW95], to create tetrahedral elements used in our FEM simulation. Rendering of the simulation results was displayed using OpenGL on a 300MHZ R12000 SGI Infinite Reality.

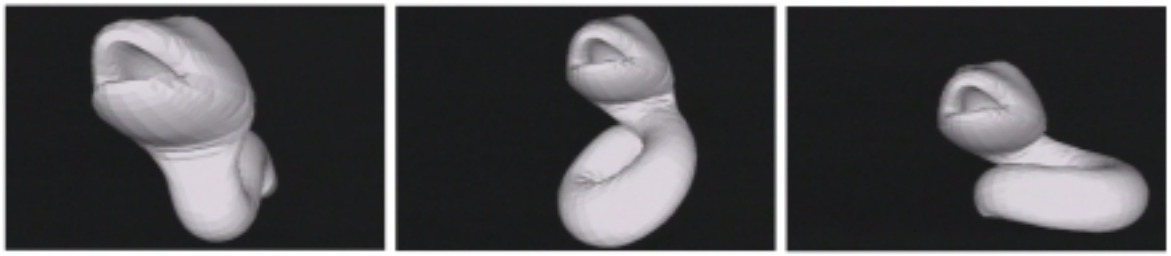

Figure 8: Large Deformation: A snake coiling up

Figure 9: A snake swallowing an apple from a bowl of fruits

7.1 System Demonstration

Figure 8 shows a large deformation simulated by our algorithm. Two sets of positional constraints were specified for internal nodes in the head part and the tail part. Given the positional constraints, the head of snake is forced to move toward its tail. The snake model has about 14,000 elements. The simulation automatically generates the natural coiling deformation. Simulating linear elasticity cannot generate such effects. It is not obvious from the images, but many small self-penetrations were resolved during the deformation. The total simulation took 57 minutes (wall-clock time).

Figure 9 are snapshots from an animation sequence where a snake swallows a deformable red apple from a bowl of fruits. The snake and the apple models have a total of 23,000 elements. Eight major keyframes were used to set the positional constraints. A few nodes inside the apple and on a part of snake's mouth follow linearly interpolated paths between these keyframes. The deformation of the apple and the snake was computed by our simulation. The entire sequence took approximately six hours to generate. As the snake swallows the apple, the snake tilts its head. Interesting effects such as the bulging around the neck and the wiggling tail motions are all automatically generated by our algorithm. It would have been difficult to create these movements using traditional keyframed animation or other deformation models with linear elasticity.

7.2 Limitations

Our method computes the distance fields within each object. Therefore, it is not best suited for handling self-penetration of thin objects, such as cloth and hair. The viscosity parameter is not automatically specified, and it needs to be adjusted for each application. If the viscosity is too small, Newton's iteration becomes unstable. If it is too large, the convergence becomes very slow. Some automatic mechanism to optimize the viscosity parameter should be investigated.

8 Summary and Conclusion

We have addressed the contact problem for passive simulation of nonlinear elastic bodies. We use finite element methods to model the elastic bodies undergoing deformation. Our technique precomputes internal distance fields for each object, which are later used for enforcing the non-penetration constraints using penalty methods. In conventional methods, the computation of penalty forces due to penetration takes a long time, and it is often a bottleneck of the contact problem in computational mechanics. By taking advantage of precomputed distance fields that deform as the finite element mesh deforms, our algorithm enables efficient computation of penalty forces and their derivatives, and yields a versatile and robust contact resolution algorithm.

This algorithm can be useful for many applications, such as simulation of passive deformable tissues in computer animation. It can also be incorporated into medical simulation used for multi-modal image registration, surgical planning and instructional medical illustration. We have demonstrated its promising potential in the generation of secondary motions for an animated scene. In the future, we plan to explore other applications and to develop a parallel implementation of the algorithm for predicting tumor movement due to tissue deformation for image registration during an image-guided surgical procedure.

References

[Bar84] A. Barr. Global and local deformations of solid primitives. *ACM Computer Graphics*, 18(3):21–30, 1984.

[BHW94] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proc. of ACM SIGGRAPH*, pages 365–372, 1994.

[BN91] T. Belytschko and M. O. Neal. Contact-impact by the pinball algorithm with penalty and lagrangian methods. *Int. J. of Numerical Methods in Engineering*, 12, 1991.

[BNC96] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer Graphics Forum, 15(3) (Eurographics'96), 1996*, pages 57–66, 1996.

[BPW93] N. I. Balder, C. B. Phillips, and B. L. Webber. *Simulating Humans*. Oxford University Press, 1993.

[BW92] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. *ACM Computer Graphics*, 26(2):303–308, 1992.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proc. of ACM SIGGRAPH*, 1998.

[BY92] T. Belytschko and I.S. Yeh. The splitting pinball method for general contact. In R.Glowinksi, editor, *Computing Methods in Applied Science and Engineering*, New York, NY, 1992. Nova Science Publishers.

[Cia88] P.G. Ciarlet. *Mathematical Elasticity*. North-Holland, Amsterdam, 1988.

[CL94] P. G. Ciarlet and J. L. Lions, editors. *HANDBOOK OF NUMERICAL ANALYSIS*, volume I - VI. Elsevier Science B.V., 1994.

[Coh92] Michael F. Cohen. Interactive spacetime control for animation. *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 293–302, 1992.

[CVT95] Martin Courshesnes, Pascal Volino, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *SIGGRAPH 95 Conference Proceedings*, pages 137–144, 1995.

[CZ92] David T. Chen and David Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 89–98, 1992.

[DG95] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. *SIGGRAPH 95 Conference Proceedings*, pages 287–290, 1995.

[DHKS93] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9:518–533, 1993.

[DKT98] T. DeRose, M. Kass, and T. Troung. Subdivision surfaces in character animation. *Proc. of ACM SIGGRAPH*, 1998.

[Don95] Peter S. Donzelli. *A Mixed-Penalty Contact Finite Element Formulation for Biphasic Soft Tissues*. PhD thesis, Dept. Mech. Eng., Rensselaer Polytechnic Institute, 1995.

[DSB99] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. *Proc. of Graphics Interface*, 1999.

[FB88] D. Forsey and R.H. Bartels. Heirarchical b-spline refinement. In *Proc. of ACM Siggraph*, pages 205–212, 1988.

[FM96] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphics Interface*, 1996.

[FM97] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. *Proc. of ACM SIGGRAH*, 1997.

[Fun93] Y.C. Fung. *Biomechanics*. Springer-Verlag, 1993.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 313–320, 1993.

[Gib98] S. F. Gibson. Volumetric object modeling for surgical simulation. *Medical Image Analysis*, 2(2), 1998.

[GM97] S. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report Technical Report, Mitsubishi Electric Research Laboratory, 1997.

[GTT89] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 21–30, 1989.

[HGB87] J. O. Hallquist, G.L. Gourdreau, and D. J. Benson. Sliding interface with contact-impact in large scale lagrangian computation, 1987.

[HWBO95] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. *SIGGRAPH 95 Conference Proceedings*, pages 71–78, 1995.

[JP99] Doug L. James and Dinesh K. Pai. Accurate real time deformable objects. *Proc. of ACM SIGGRAPH*, 1999.

[KAB95] R. Kimmel, A. Amir, and A. M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6), 1995.

[KGC+96] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. Parish. Simulating facial surgery using finite element methods. *SIGGRAPH 96 Conference Proceedings*, pages 421–428, 1996.

[KO88] N. Kikuchi and J. T. Oden. Contact problems in elasticity: A study of variational inequalities and finite element methods. *SIAM*, 1988.

[LTW95] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic face modeling for animation. *SIGGRAPH 95 Conference Proceedings*, pages 55–62, 1995.

[Mil88] Gavin S. P. Miller. The motion dynamics of snakes and worms. *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 169–178, 1988.

[MT92] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 309–312, 1992.

[MW95] D.L. Marcum and N.P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9), September 1995.

[Ogd84] R. W. Ogden. *Nonlinear Elastic Deformation*. Dover Publications, Inc., 1984.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. *Proc. of ACM SIGGRAPH*, 1999.

[OHZ97] J. F. O'Brien, J. K. Hodgins, and V. B. Zordan. Combining active and passive simulations for secondary motion. *Technical Sketch, Visual Proceedings of ACM SIGGRAPH*, 1997.

[OS88] S. J. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. of Comp. Phys.*, 1988.

[Pow69] M. J. D. Powell. A method for nonlinear constraints in minimization problems. *Optimization*, 1969.

[PW89] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 215–222, 1989.

[RE99] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivaraite functions. *ACM Symposium on Solid Modeling and Applications*, 1999.

[RH91] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 349–358, 1991.

[Set99] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[SF95] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. *Proc. of ACM SIGGRAH*, 1995.

[SOH99] R. W. Sumner, J. F. O'Brien, and J. K. Hodgins. Animating sand, mud, and snow. *Computer Graphics Forum*, 18(1), March 1999. Preliminary version appeared in Graphics Interface 1998.

[SP86] T. W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. *ACM Computer Graphics*, 20(4):151–160, 1986.

[SWF+93] John M. Snyder, Adam R. Woodbury, Kurt Fleischer, Bena Currin, and Alan H. Barr. Interval method for multi-point collision between time-dependent curved surfaces. *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 321–334, 1993.

[Tal94a] Patrick Le Tallec. Equilibrium problems with frictionless contact. In P. G. Ciarlet and J. L. Lions, editors, *HANDBOOK OF NUMERICAL ANALYSIS*, volume 3, pages 565–578. Elsevier Science B.V., 1994.

[Tal94b] Patrick Le Tallec. Numerical methods for nonlinear three-dimensional elasticity. In P. G. Ciarlet and J. L. Lions, editors, *HANDBOOK OF NUMERICAL ANALYSIS*, volume 3, pages 465–622. Elsevier Science B.V., 1994.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 205–214, 1987.

[TQ94] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Trans. on Computer Graphics*, 13(2):103–136, 1994.

[VBZ90] Brian Von Herzen, Alan H. Barr, and Harold R. Zatz. Geometric collisions for time-dependent parametric surfaces. *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 39–48, 1990.

[vFV90] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 225–234, 1990.

[WW90] A. Witkin and W. Welch. Fast animation and control of non-rigid structures. *ACM Computer Graphics*, 24:243–252, 1990.

[WW92] W. Welch and A. Witkin. Variational surface modeling. *ACM Computer Graphics*, 26(2):157–166, 1992.

[ZC99] Yan Zhuang and John Canny. Real-time and physically realistic simulation of global deformation. *SIGGRAPH99 Sketches and Applications*, 1999. Also Appeared in IEEE Vis'99 Late Breaking Hot Topics.