ArtDefo
Accurate Real Time Deformable Objects

Doug L. James[1] and Dinesh K. Pai[1,2]

University of British Columbia

Abstract

We present an algorithm for fast, physically accurate simulation of deformable objects suitable for real time animation and virtual environment interaction. We describe the boundary integral equation formulation of static linear elasticity as well as the related Boundary Element Method (BEM) discretization technique. In addition, we show how to exploit the coherence of typical interactions to achieve low latency; the boundary formulation lends itself well to a fast update method when a few boundary conditions change. The algorithms are described in detail with examples from ArtDefo, our implementation.

1 Introduction

Simulation of deformable objects is a key challenge, with a long history in computer graphics. Deformable object models are important because so much of our physical world can not be modeled as rigid bodies. The most important examples are, of course, humans, human tissue, and human-like characters. We conjecture that one reason for the preponderance of robotic, wooden, and chitinous characters in recent computer animation is the difficulty of realistic deformable object simulation.

Simulating deformable objects for graphics is difficult because of the conflicting demands of interactivity and accuracy. In animation, the simulation must be fast enough to provide useful feedback to the animator. In virtual environments, the objects must deform in real time, in response to user input. In surgical training and games with haptic force feedback, forces due to the deformation must also be computed in real time.

On the other hand, physical accuracy is important since it allows users to treat deformable objects as "black boxes" and focus on the their primary objectives. In animation, it allows an animator to focus on the action rather than on surface tweaking. In surgical simulation, the contact forces felt by the surgeon must be accurate — this increases the likelihood that the surgeon has acquired useful manual skills for real surgeries, rather than skill at a new video game.

As a result of these conflicting demands, work to date can be broadly classified into two categories. (1) Interactive models: in these models, speed and low latency are paramount and physical accuracy is secondary. Typical examples include mass-spring models and spline surfaces used as deformable models. (2) Accurate models: in these models, physical accuracy is paramount; the models start with the constitutive laws of the material and solve the resulting boundary value PDE's numerically using, for instance, the Finite Element Method (FEM). The models are computationally expensive and are typically simulated off-line.

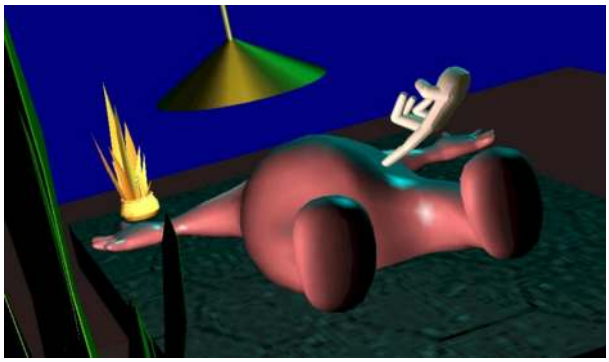


Figure 1: Example of interactive simulation with ArtDefo. Here, a human-like character is being tickled by its alien abductors. The body is rendered as a Loop subdivision surface.

In this paper we describe a physically based algorithm for deformable object simulations which makes this tradeoff unnecessary for many applications in which linear elastic models are sufficient. Interactive speeds as well as accuracy are attained by exploiting the fact that a linear model allows many system responses (Green's functions) to be precomputed and then combined later in real time using an incremental low-rank update solution reconstruction approach. Our linear elastic model is based on boundary integrals and the Boundary Element Method (BEM). While the low rank update approach could in fact be applied to any discrete linear system, it is particularly convenient and instructive using a BEM discretization. The solver's low latency success is due in part to the limited amount of nonzero boundary data associated with typical interactions.

This model has been implemented in our system, ArtDefo, using Java. We chose Java for integration into our software environment and for easy accessibility from the Internet. Despite the performance penalty imposed by current Java virtual machines for numerical array-based computations, our system provides good interactive performance for the examples described in this paper. Visit `www.cs.ubc.ca/~djames/deformable` for online demos.

Figure 1 shows an interactive simulation in which the torso of the character is deformable. The deformation and resulting contact forces due to a virtual finger or medical instrument touching a location on the torso were computed in approximately a millisecond after an initial cost of several milliseconds on an ordinary 350MHz PC. The global deformation is accurately computed and important phenomena such as the upward bulging of the torso due to the preservation of volume are automatically produced.

An important feature is that since our model is physically

[1]Institute of Applied Mathematics

[2]Dept. of Computer Science, 2366 Main Mall, UBC, Vancouver, Canada. {djames|pai}@cs.ubc.ca. This work was supported in part by IRIS NCE and NSERC.

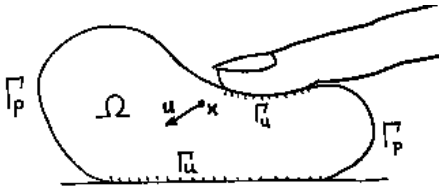


Figure 2: Notation

based, deformable objects are also easier to create and modify; the object's deformation behavior is specified using a few material properties rather than by adjusting a large number of spring constants and other parameters. Material properties for standard objects can be looked up in a handbook, and properties can be modified in intuitive ways by making it "more compressible (sponge-like)" "less stiff," "more dense" etc. We can easily represent physical constraints, such as incompressibility, which are difficult for deformable models based on spring networks and splines.

Finally, even though the technique has somewhat complex mathematical foundations, it is easy to implement. We describe the basic algorithms in sufficient detail to allow implementation. We bring out the connections between deformation computations and radiosity methods in global illumination. This will allow rendering professionals to easily understand and implement fast simulations of deformation.

The paper is organized as follows. In §2 we present an overview of all the important techniques in this paper. Connections to related work are discussed in §3. In subsequent sections §4–7 we provide details of the algorithms, with specific integrals needed for the constant element case collected in the Appendix.

2 Overview

We describe the key ideas in our system in this section, at a high level. Details are provided in the sections that follow.

2.1 Linear Elasticity

Our model is physically based and starts with the physics of static linear elastic deformation. Linear elasticity accurately models small deformations of objects and is commonly used in engineering analysis. Biological tissues and large deformations will involve additional non-linear effects; nevertheless, linear models can play an important role in a hybrid simulation given that they can be significantly cheaper to evaluate.

We use the following notation (see Figure 2) which is standard (e.g., [5]). The domain of the deformable object is denoted $\Omega \subset \mathbb{R}^3$ with boundary Γ. A point $\mathbf{x} \in \Omega$ undergoes a displacement $\mathbf{u}$ due to deformation. The deformation is governed locally by Navier's equation (described in §4) which is a generalization of Hooke's law from high school physics. It is written

$$\mathbf{N}\mathbf{u} + \mathbf{b} = 0, \tag{1}$$

where $\mathbf{N}$ is a linear second-order differential operator, and $\mathbf{b}$ is a term due to "body forces" like gravity which act everywhere in the body. Navier's equation must be satisfied at all points $\mathbf{x} \in \Omega$; $\mathbf{N}$, $\mathbf{u}$, and $\mathbf{b}$ are functions of $\mathbf{x}$. $\mathbf{N}$ encodes all the elastic material properties of the object. For simplicity, we will assume $\mathbf{b}=0$ in the remainder of this section.

2.2 Boundary Value Problem

To specify the deformation, we must also describe how the object interacts with the environment at its boundary Γ. We may want to specify *displacement boundary conditions* on some parts of the boundary (denoted Γ_u). For instance, we can indicate that the boundary is in contact with a rigid table or is being poked by a virtual finger whose motion is known (e.g., through a motion tracker attached to the user's hand). In other parts of the boundary (denoted Γ_p), it may be more natural to specify the force or rather the *traction*, $\mathbf{p}$, which is defined as the force per unit area (and has the same units as pressure). These *traction boundary conditions* are useful for specifying that the boundary is free to move (zero traction).

The boundary conditions, along with Navier's equation, constitute the boundary value problem (BVP). Changing the type of a boundary condition changes the BVP significantly, which is an important consideration we will return to in §2.5.

2.3 Boundary Integral Formulation

In graphics applications, we are primarily interested in determining the displacement $\mathbf{u}$ and traction $\mathbf{p}$ on the boundary of the body. The boundary displacement $\mathbf{u}$ is needed for rendering while the boundary traction $\mathbf{p}$ is needed for force feedback user interfaces. Therefore it is possible to use a boundary integral formulation of Navier's equation,

$$\mathbf{c}\mathbf{u} + \int_\Gamma \mathbf{p}^*\mathbf{u}\, d\Gamma = \int_\Gamma \mathbf{u}^*\mathbf{p}\, d\Gamma. \tag{2}$$

Here $\mathbf{c}$ is a (known) function that depends only on the geometry of the boundary, and $\mathbf{u}^*$ and $\mathbf{p}^*$ are *fundamental solutions* which depend only on known elasticity properties.

A key advantage of the boundary integral formulation is that all the unknowns are on the boundary Γ, and are exactly the quantities we are interested in. In contrast, the FEM has unknowns in the interior as well.

For an excellent survey of boundary integral equation issues we refer the reader to [2].

We remark that boundary integral equations also arise in global illumination in the form of the Rendering Equation [17], where it is derived from the physics of radiative energy transport. Therefore many of the same issues arise in solving such integral equations. There are also important differences; equation 2 is a vector integral equation instead of a scalar equation, and far from being a Fredholm integral equation of the second kind, the vector equation involves both single and double layer potential integral operators [2].

2.4 Numerical Discretization

To solve equation 2 numerically, it is discretized by approximating $\mathbf{u}$, $\mathbf{u}^*$, etc., in finite dimensional function spaces. We discuss the choice of these spaces in §6 when we discuss the BEM, but in the end the integral equation is reduced to a matrix equation of the form

$$\mathsf{H}\mathsf{u} = \mathsf{G}\mathsf{p}, \tag{3}$$

where u is a vector of displacements $\mathsf{u}_i = \mathbf{u}(\mathbf{x}_i)$, sampled at a finite set of nodes $\mathbf{x}_i, i = 1, \ldots, n$, on the boundary, and p are the corresponding tractions at these nodes. For the constant boundary element we describe in §6, $\mathbf{x}_i$ are the centroids of boundary triangles.

The $3n \times 3n$ dense matrices H and G are discretized versions of the operators on the left and right and sides of (2),

respectively. They are analogous to the form factors computed in radiosity methods.

Finally, specifying boundary conditions, either $\mathbf{u}$ or $\mathbf{p}$, at each of the n nodes, yields a system of equations for the unknown nodal quantities:

$$\mathsf{A}\mathsf{v} = \mathsf{z}. \tag{4}$$

Since A depends on the BVP specified, it is not possible to simply invert A ahead of time. Nor is it realistic to invert and store all possible matrices ahead of time.

2.5 Interactive Deformation

Equation 4 is still prohibitively expensive to solve at interactive rates for even moderate size problems consisting of a few hundred surface nodes. This is so even though fast transform and iterative methods are available for solving (4). Therefore solving the discrete equations directly in real time is not necessarily suitable for an interactive method. This is especially true when it is possible to do much of the work ahead of time.

In a typical situation a user may, say, poke the deformable object with a virtual finger, and then move the finger along its surface. As mentioned, this process is accompanied by a sequence of different matrix problems to be solved.

We address this issue by observing that, for example, as the user makes or breaks contact with a surface node only 3 columns of A change. This implies that there is a great degree of similarity between these matrix BVP problems. We can then compute the solution to the new problem efficiently using results obtained directly from using the Sherman-Morrison-Woodbury formula [12]. We describe this procedure in detail in §7 — we mention here that if s boundary conditions change, we spend $O(s^3)$ time updating a set of weights, after which correct solutions can be computed, with each using an additional $O(ns)$ time. While complexity analysis isn't indicative of interactivity, we find that these solution methods are very fast in practice. Also, since this is a *direct* solution method, latency bounds are strictly related to s. Hence the real-time performance can be easily predicted.

With the boundary integral formulation and our optimizations, we are able to accurately simulate deformation problems on ordinary PCs at interactive rates, even with changing boundary conditions. The high cost usually associated with deformation computations is relegated to precomputation and storage of the database used by the fast run-time solver.

3 Related Work

There has been considerable work in the area of deformable models in graphics. We refer the reader to a recent, thorough survey [10] which describes much of this work. We mention some relevant physically based models here, which include cloth models (e.g., [4]), linked volumetric objects [11], modal models [23] and the pioneering work of Terzopoulos and colleagues [26, 24, 25, 22]. In the following we focus mainly on models specifically designed to simulate physical deformation of objects, rather than purely geometric deformable models.

Among physically-based models, the most popular are mass-spring models (e.g., [4, 19, 28]). They divide the domain Ω into set of mass points connected by springs. The technique is effective for several applications, can be used for static as well as dynamic deformations, and has been extended to model non-linear materials and even liquids [27]. However, despite the simplicity of formulation, the resulting systems can be expensive to solve; the inaccuracy of the method implies that a larger number of nodes are required for realistic simulations (compared to FEM and BEM techniques), and these result in larger system matrices. Numerical stiffness is also a problem [3]. The model parameters are difficult to tune and can easily produce undesired Jello-like or putty-like behavior. Worse, common properties of materials like incompressibility are difficult to model. In contrast our model (and others explicitly based on Navier's equation) can be tuned using a few, well-known material properties; incompressibility is trivial to specify using Poisson's ratio $\nu = \frac{1}{2}$.

The Finite Element Method (FEM), a widely used, flexible and accurate method for solving Navier's equations, has seen application in graphics [13, 7, 8, 6]. However, its use has been limited by the complexity of the method and the cost of solving the resulting linear system.

While FEM techniques are effective general tools for (especially nonlinear) elasticity, we believe that the BEM, on which our technique is based, has certain features which will be of interest to the graphics community. These methods also have a long history in engineering analysis (e.g., [5, 9]). Radiosity methods for global illumination are closely related and are essentially boundary element methods [1, 20]. To our knowledge, BEM techniques have not been used in computer graphics for the simulation of deformable objects.

In BEM, the unknowns are only the boundary displacements and forces — exactly the quantities we need in graphics and haptics. This is in contrast with many engineering applications, in which stresses and deformations inside the body are important for determining properties of a structure, making FEM more natural. The BEM leads to a small but dense set of well–conditioned equations which are simple to solve; the FEM leads to a larger, but sparse, set of equations which can be solved efficiently using sparse matrix techniques. Fast solution methods exist for FEM, e.g., multigrid; numerous fast integral transform methods exist for BEM [2]. The BEM is also more accurate for computing contact forces than the FEM since forces are solved for just like displacements, instead of being derived from displacements using difference formulas [5]. As a result, the BEM easily handles mixed boundary conditions. And, perhaps of greatest interest, the BEM can use the same boundary discretization as used for rendering — no separate meshing is required; in FEM the interior has to be meshed as well. However, while the boundary-only nature of the BEM is a selling point, it also restricts the class of materials which it can handle to those with homogeneous material properties. A few regions with different material properties can be handled using BEM by introducing internal boundaries, but for complicated inhomogeneities a domain method, such as the FEM, is better.

Our use of the Sherman-Morrison-Woodbury formula to exploit coherence between deformable is related to the technique used in the FEM community called "structural reanalysis" [14, 18]. Unfortunately, due to the overhead introduced by internal domain nodes, these methods are not necessarily suitable for interactive applications. A notable exception is the work of [6] and colleagues; they perform various linear system optimizations, similar to those presented here, and eliminate interior nodes, using a technique called condensation, to achieve a boundary-only representation.

4 Navier's Equation

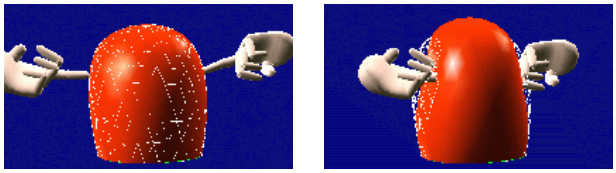
Linear elastostatic objects with isotropic and homogeneous material properties, have displacements satisfying the well-known Navier's equation on Ω,

$$G\sum_{k=1}^{3}\left(\frac{\partial^2 u_i}{\partial x_k^2}+\frac{1}{1-2\nu}\frac{\partial^2 u_k}{\partial x_k \partial x_i}\right)+b_i=0, \qquad (5)$$

which is conveniently written in a vector operator form as

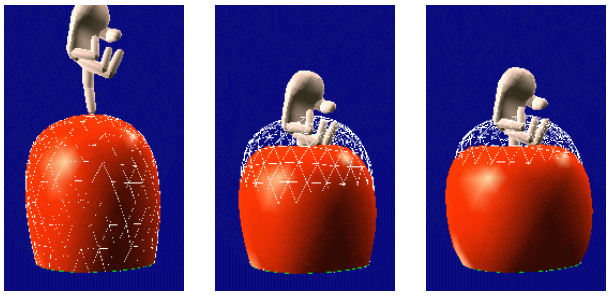
$$(\mathbf{Nu})(\mathbf{x})+\mathbf{b}(\mathbf{x})=0, \qquad \mathbf{x}\in\Omega. \qquad (6)$$

Here ν is Poisson's ratio and G is the shear modulus. These are material properties which can be found in handbooks for many materials. Suitable values for Poisson's ratio are $0<\nu\leq\frac{1}{2}$, with $\nu=\frac{1}{2}$ corresponding to an incompressible material. The shear modulus G is positive, with larger values resulting in larger forces accompanying a given deformation. Figure 4 shows the effect of changing ν; see also Figure 3.



(a) (b)

Figure 3: A test nodule is pinched between two fingers. The nodule is a Loop subdivision surface, whose control polyhedron is an octahedron. The boundary of one face of the octahedron is tagged as "sharp" [16] and leads to a sharp edge around the bottom face of the object. We impose a zero-displacement boundary condition on this face, and zero traction everywhere else, except for the two finger contacts.



(a) (b) (c)

Figure 4: Poisson's ratio, ν, provides an easy way to describe the compressibility of a material. Figure (a) shows the example nodule in its rest state. A coarse reference mesh is also shown in white on the surface. In figure (b), $\nu=0.01$, making the material very compressible; the nodule exhibits a sponge-like behavior, deforming mainly in the vicinity of the contact. In Figure (c) $\nu=0.5$, making the material incompressible; the sides of the nodule bulge to conserve volume.

The traction at a point on the surface is

$$p_i=p_i(\mathbf{x})=G\sum_{j=1}^{3}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)n_j+\frac{2G\nu}{1-2\nu}n_i\left(\sum_{k=1}^{3}\frac{\partial u_k}{\partial x_k}\right)$$

where n_i are the direction cosines of the outward normal. In a vector operator notation this becomes

$$\mathbf{p}(\mathbf{x})=(\mathbf{Pu})(\mathbf{x}), \qquad \mathbf{x}\in\Gamma. \qquad (7)$$

5 Boundary Integral Formulation

Similar to Laplace's equation, Navier's equation on a domain may be converted to an integral equation defined on the boundary of that domain. At the heart of the derivation is integration by parts, which produces boundary integrals from volume integrals. The end result is that Navier's equations (5) on, for example, a bounded domain may be converted into a set of integral equations. The direct boundary integral equation formulation yields the vector integral equation

$$\mathbf{c}(\mathbf{x})\mathbf{u}(\mathbf{x})+\int_\Gamma \mathbf{p}^*(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y})\,d\Gamma(\mathbf{y}) \qquad (8)$$
$$=\int_\Gamma \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{p}(\mathbf{y})\,d\Gamma(\mathbf{y})+\int_\Omega \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{b}(\mathbf{y})\,d\Omega(\mathbf{y})$$

valid at a point $\mathbf{x}$ on the boundary Γ [5]. Three matrix functions occur in this equation:

$$\begin{aligned}\mathbf{c} &= \mathbf{c}(\mathbf{x}) &&= [c_{ij}],\\ \mathbf{u}^* &= \mathbf{u}^*(\mathbf{x},\mathbf{y}) &&= [u^*_{ij}],\\ \mathbf{p}^* &= \mathbf{p}^*(\mathbf{x},\mathbf{y}) &&= [p^*_{ij}].\end{aligned}$$

The integral kernel functions $u^*_{ij}(\mathbf{x},\mathbf{y})$ and $p^*_{ij}(\mathbf{x},\mathbf{y})$ are known fundamental solutions and tractions, respectively, and are provided in the next section. The coefficient $c_{ij}(\mathbf{x})$ depends on the smoothness properties of the boundary at $\mathbf{x}$, but is not needed explicitly (see Appendix A.2.1).

5.1 Fundamental Solutions

The fundamental solutions of Navier's equation, $u^*_{ij}(\mathbf{x},\mathbf{y})$, correspond to the displacement in the j^{th} direction at a field point, $\mathbf{y}$, as produced by a unit point load applied in each of the i directions at a specified load point, $\mathbf{x}$, in an infinite linear elastic medium. This corresponds to the fundamental solution due to Kelvin [21]. Conceptually, this point load fundamental solution plays an analogous role in elasticity as the familiar $\frac{1}{r}$ Coulomb potential solution accompanying a point charge in electrostatics. In both cases, the fundamental solutions are highly localized and decay very quickly, e.g., the fundamental displacements have a typical $\frac{1}{r}$ character while the fundamental tractions behave like $\frac{1}{r^2}$. Mathematically, $u^*_{ij}(\mathbf{x},\mathbf{y})$ is the j^{th} component of the displacement solution to

$$(\mathbf{Nu})(\mathbf{y})+\delta(\mathbf{x}-\mathbf{y})\hat{\mathbf{e}}_i=0,$$

where the vector operator notation from (6) has been used. The fundamental tractions are related to the fundamental displacements via (7), that is

$$\mathbf{p}^*=\mathbf{Pu}^*.$$

Expressions for the fundamental solutions are [5]

$$u^*_{ij}(\mathbf{x},\mathbf{y})=\frac{1}{16\pi(1-\nu)G}\left\{\frac{(3-4\nu)\,\delta_{ij}}{r}+\frac{r_i r_j}{r^3}\right\}$$
$$p^*_{ij}(\mathbf{x},\mathbf{y})=\frac{(1-2\nu)}{8\pi(1-\nu)}\left[\frac{(r_i n_j-r_j n_i)}{r^3}-\left\{\frac{\delta_{ij}}{r^2}+\frac{3r_i r_j}{(1-2\nu)\,r^4}\right\}\frac{\partial r}{\partial \mathbf{n}(\mathbf{y})}\right]$$

where

$$\begin{aligned} \mathbf{r} &= \mathbf{y}-\mathbf{x} & r &= |\mathbf{r}| \\ r_i &= (\mathbf{r})_i & \frac{\partial r}{\partial \mathbf{n}(\mathbf{y})} &= \frac{\mathbf{r}\cdot\mathbf{n}(\mathbf{y})}{r} \end{aligned}$$

and $\mathbf{n}(\mathbf{y})$ is the outward unit normal at $\mathbf{y} \in \Gamma$.

5.2 Internal Body Forces

Any user-specified body forces mildly complicate the boundary-only character of the integral equations, as they introduce a volume integral term in (8). However, for certain classes of functions, e.g., polynomials, it is possible to analytically convert the volume integral into a boundary integral using essentially repeated integration by parts via the Multiple Reciprocity Method [5]. For example, a constant gravitational force, $\mathbf{b}=\rho\mathbf{g}$, may be evaluated as a boundary integral. More simply, concentrated force loads, $\mathbf{b}=\mathbf{b_0}\delta(\mathbf{x}-\mathbf{x}_0)$, are trivial to integrate, and are useful for introducing internal body articulation. Due to space limitations, the body force term will not be mentioned hereafter.

6 The Boundary Element Method

The Boundary Element Method (BEM) is a straight-forward approach to discretizing integral equations defined on the boundary via a collocation method. There are three main steps when implementing the BEM in 3D:

1. Discretize the boundary Γ into a set of N non-overlapping elements which represent the displacements and tractions by functions which are piecewise interpolated between the element's nodal points.

2. Apply the integral equation(s) at each of the n boundary nodes, and perform the resulting integrals over each boundary element in order to generate an undetermined system of $3n$ equations involving the $3n$ nodal displacements and $3n$ nodal tractions.

3. Apply the boundary conditions of the desired boundary value problem, fixing n nodal values (either displacement or traction) per direction. The remaining linear system of $3n$ equations is determined and may be solved to obtain the unknown nodal boundary values.

Drawing on the notation from [5], the discretization of (8), dropping the body force, may be summarized as follows. The piecewise interpolated displacement and traction functions evaluated at the point $\mathbf{x}$ may be written as

$$\begin{aligned} \mathbf{u} &= \mathbf{u}(\mathbf{x}) = (u_1, u_2, u_3)^T = \mathbf{\Phi}(\mathbf{x})\mathsf{u} \\ \mathbf{p} &= \mathbf{p}(\mathbf{x}) = (p_1, p_2, p_3)^T = \mathbf{\Phi}(\mathbf{x})\mathsf{p} \end{aligned} \tag{9}$$

where $\mathbf{\Phi}(\mathbf{x})$ is an interpolation matrix and u and p are n-vectors of the nodal displacement and traction 3-vectors, respectively, e.g., $\mathsf{u} = [\mathsf{u}_1, \ldots, \mathsf{u}_n]^T$.

The displacement, traction and c vectors at the i^{th} node, $\mathbf{x}_i$, will be denoted by

$$\mathsf{u}_i = \mathbf{u}(\mathbf{x}_i), \qquad \mathsf{p}_i = \mathbf{p}(\mathbf{x}_i), \qquad \mathsf{c}_i = \mathbf{c}(\mathbf{x}_i),$$

respectively. Substituting (9) into the elasticity integral equation (8) applied at $\mathbf{x}_i$ and converting the surface integrals into sums of integrals over each boundary element, one obtains

$$\begin{aligned} \mathsf{c}_i\mathbf{u}_i + \sum_{j=1}^{N}\left(\int_{\Gamma_j} \mathbf{p}^*(\mathbf{x}_i,\mathbf{y})\,\mathbf{\Phi}(\mathbf{y})\,d\Gamma(\mathbf{y})\right)\mathsf{u} \\ = \sum_{j=1}^{N}\left(\int_{\Gamma_j} \mathbf{u}^*(\mathbf{x}_i,\mathbf{y})\,\mathbf{\Phi}(\mathbf{y})\,d\Gamma(\mathbf{y})\right)\mathsf{p} \end{aligned}$$

which, in an obvious notation, may be written as

$$\mathsf{c}_i\mathsf{u}_i + \sum_{j=1}^{n}\hat{\mathsf{h}}_{ij}\mathsf{u}_j = \sum_{j=1}^{n}\mathsf{g}_{ij}\mathsf{p}_j. \tag{10}$$

For convenience, define off-diagonal h_{ij} as $\hat{\mathsf{h}}_{ij}$, but let

$$\mathsf{h}_{ii} = \mathsf{c}_i + \hat{\mathsf{h}}_{ii}.$$

Assembling the equations at all nodes into a block matrix system yields

$$\sum_{j=1}^{n}\mathsf{h}_{ij}\mathsf{u}_j = \sum_{j=1}^{n}\mathsf{g}_{ij}\mathsf{p}_j \quad \text{or} \quad \mathsf{Hu} = \mathsf{Gp}. \tag{11}$$

The final step is to specify the boundary conditions at each of the n nodes, then bring the unknowns to the left-hand side, and the knowns to the right-hand side to obtain the final linear system

$$\mathsf{Av} = \mathsf{z}, \tag{12}$$

which may be solved for the unknown nodal quantities, v.

All that remains is to determine the integrals for the matrix entries of H and G. Indeed, this is the part of the BEM which takes the majority of a computation. Complete formulae for constant boundary elements, are provided in the Appendix for those who are interested in constructing their own elasticity solver. It is the simplest element for the reader to implement and understand. Formulae for linear elements may be found in [15].

6.1 Constant Element Case

Analogous to the midpoint rule for integrating a univariate function, integration of a triangular constant element is accomplished using data located at the centroid. This corresponds to a centroid collocation scheme, as the j^{th} node, $\mathbf{x}_j$, is identified as the centroid of the j^{th} element, and is where the elastic state is represented accurately. In this case $n = N$. Since the collocation node lies in the element's interior, it is called a *nonconforming* element. This happens to make the element particular easy to implement, as connectivity is not required. It also has the convenient casual property that special care need not be taken to accommodate corners or sharp edges [5].

7 Interactive simulation

To solve equation 12 at interactive rates, we exploit the coherence of typical deformable object interactions. This allows us to achieve low latency at the expense of memory and precomputation.

7.1 Precomputation

Suppose A_0 is the matrix in (12) corresponding to a particular reference BVP. In the following, we assume that we have precomputed $\mathsf{A}_0^{-1}\mathsf{H}$ and $\mathsf{A}_0^{-1}\mathsf{G}$. Note that only half of the columns from the last two matrices require storage since A_0 consists of columns of G and H, and $\mathsf{A}_0^{-1}\mathsf{A}_0 = \mathsf{I}$. In general, several such reference BVPs can be precomputed and it is possible to perform this computation adaptively in a low priority thread.

We remind the reader that to simplify indexing, A_0, H, etc., are 3×3 block matrices; similarly v, z, etc., are block column matrices whose elements are 3D vectors.

7.2 Changing boundary values

The simplest case of coherence occurs for the reference BVP when there is a change in some boundary values of the reference BVP. The new solution can be computed in time $O(n(|S_u| + |S_p|))$ as

$$\mathsf{v}^{(0)} = \mathsf{v}_{\mathsf{unchanged}} + \sum_{j \in S_u} (-\mathsf{A}_0^{-1}\mathsf{H}_j)\, \delta\bar{\mathsf{u}}_j + \sum_{j \in S_p} (\mathsf{A}_0^{-1}\mathsf{G}_j)\, \delta\bar{\mathsf{p}}_j \quad (13)$$

where $\delta\bar{\mathsf{u}}_j$ and $\delta\bar{\mathsf{p}}_j$ are changes in the specified boundary values, G_j (respectively H_j) represents the j^{th} column of G (resp. H), and S_u and S_p are mutually exclusive sets indicating where specified displacements and tractions, respectively, are changing. The term $\mathsf{v}_{\mathsf{unchanged}}$ represents already computed contributions from unchanged or previous boundary values, and provides a way to avoid somewhat redundant vector summations. If present, (un)changing parameterized body forces are handled in a similar manner.

7.3 Changing boundary conditions

The more difficult case occurs if the type of a boundary condition changes; for instance, a new contact between a rigid virtual finger and deformable object will change a traction condition to a displacement one. The system matrix A_0 itself changes and can introduce unacceptable latencies if we have to solve the linear system afresh.

The key to fast computation is to realize: (1) Typically only a few columns of A_0 change; and (2) the columns change in very specific ways: a column of A_0 which originated in $(-\mathsf{H})$ is replaced by the corresponding column of G, or vice-versa. We exploit (1) by using the Sherman-Morrison-Woodbury formula; we exploit (2) by extracting large parts of the formula from the precomputed matrices listed in §7.1

7.3.1 The Sherman-Morrison-Woodbury Formula

Expressions relating the inverse of a matrix to the inverse of the matrix after it has undergone some minor changes are well-known from linear algebra and have enjoyed numerous applications [14]. Essentially, given our square block matrix A_0 and its inverse A_0^{-1}, as well as a second square matrix A_S which is related to the first matrix by a rank $3s$ change,

$$\mathsf{A}_\mathsf{S} = \mathsf{A}_0 + \mathsf{R}\mathsf{S}^\mathsf{T},$$

where R and S have s block columns, the Sherman-Morrison-Woodbury (SMW) formula [12], provides an expression for the inverse of the second matrix as

$$\mathsf{A}_\mathsf{S}^{-1} = \mathsf{A}_0^{-1} - \mathsf{A}_0^{-1}\mathsf{R}\left[\mathsf{I} + \mathsf{S}^\mathsf{T}\mathsf{A}_0^{-1}\mathsf{R}\right]^{-1}\mathsf{S}^\mathsf{T}\mathsf{A}_0^{-1}. \quad (14)$$

However, explicit calculation of the inverse is not our intent. We can do better by using the formula to efficiently evaluate $\mathsf{A}_\mathsf{S}^{-1}\mathsf{z}$ matrix-vector multiplications as shown below.

7.3.2 The Fast Update Process

If the type of boundary condition on node j changes, only the j^{th} column of A_0 changes; let L_j denote the column that is replaced, and let M_j be its replacement. As described in §6, these columns are either $(-\mathsf{H}_j)$ or G_j. Now suppose s boundary conditions change (corresponding to nodes $j \in S$, where S is an index set). Let L be the $n \times s$ matrix whose columns are replaced by those of M, and let $\delta\mathsf{A}_\mathsf{S} = \mathsf{M} - \mathsf{L}$. Formally, we can write the change in A_0 as

$$\mathsf{A}_\mathsf{S} = \mathsf{A}_0 + \delta\mathsf{A}_\mathsf{S}\,\mathsf{E}^\mathsf{T}$$

where E is the $n \times s$ submatrix of the $n \times n$ identity matrix, consisting of columns $j \in S$.

Applying the SMW formula to $\mathsf{A}_\mathsf{S}^{-1}\mathsf{z}$ and grouping terms,

$$\mathsf{A}_\mathsf{S}^{-1}\mathsf{z} = (\mathsf{A}_0^{-1}\mathsf{z}) - (\mathsf{A}_0^{-1}\delta\mathsf{A}_\mathsf{S})\left[\mathsf{I} + \mathsf{E}^\mathsf{T}(\mathsf{A}_0^{-1}\delta\mathsf{A}_\mathsf{S})\right]^{-1}\mathsf{E}^\mathsf{T}(\mathsf{A}_0^{-1}\mathsf{z}), \quad (15)$$

we obtain a formula that can be transcribed into an efficient algorithm for computing the new solution v for general boundary data when s boundary conditions have changed.

The first step is to construct the portion independent of z that may be reused for other BVPs with $\mathsf{A} = \mathsf{A}_\mathsf{S}$:

1. "Compute" $\mathsf{W} = \mathsf{A}_0^{-1}\delta\mathsf{A}_\mathsf{S}$. Since $\delta\mathsf{A}_\mathsf{S} = \mathsf{M} - \mathsf{L}$ and $\mathsf{A}_0^{-1}\mathsf{L} = \mathsf{E}$, obtaining W only consists of looking up $\mathsf{A}_0^{-1}\mathsf{M}$ in the precomputed $\mathsf{A}_0^{-1}\mathsf{G}$ and $\mathsf{A}_0^{-1}\mathsf{H}$ matrices.
2. Construct the s-by-s matrix
$$\mathsf{C}_\mathsf{S} = \mathsf{I} + \mathsf{E}^\mathsf{T}\mathsf{W}$$
by extracting rows from W. The matrix C_S is traditionally called the *capacitance matrix* [14].
3. Compute the LU factors of C_S [12] and store them for future use. Denote the LU decomposition by $\mathsf{C}_\mathsf{S}^{-1}$.

The total cost is $O(s^3)$. Once this has been done, particular BVPs may be solved efficiently as follows:

1. Construct $\mathsf{v}^{(0)} = \mathsf{A}_0^{-1}\mathsf{z}$ efficiently as in (13).
2. Extract rows of $\mathsf{v}^{(0)}$ to make $\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)}$.
3. Apply $\mathsf{C}_\mathsf{S}^{-1}$ to the s-vector $\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)}$.
4. Apply W to the s-vector $(\mathsf{C}_\mathsf{S}^{-1}\mathsf{E}^\mathsf{T}\mathsf{v}^{(0)})$ and subtract the result from $\mathsf{v}^{(0)}$ to obtain the final solution v.

After the initial construction of $\mathsf{v}^{(0)}$, whose cost is $O(n(|S_u| + |S_p|))$, the solve only costs $O(ns)$ operations.

8 Conclusions

We have described a fast method for simulating linear elastic deformable objects in interactive applications. We have used the boundary element method for linear elasticity, and have shown that it is a useful tool for graphics applications, due to its boundary-only nature, accuracy, and simplicity of implementation. It is particularly useful for interactive

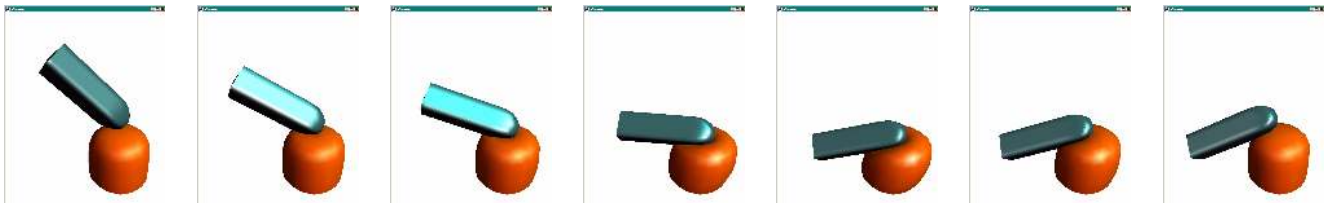


Figure 5: Simulation of a kinematically constrained rigid object contacting a deformable object. This is a difficult problem because the contact boundary conditions keep changing during the motion. ArtDefo solves such a sequence of problems quickly by combining precomputed system responses using a rank update procedure. It is possible to perform multiple solves each frame in order to guarantee unilateral contact constraints are satisfied.

simulation because of the ease of combination with low-rank update solution approaches derived from the Sherman-Morrison-Woodbury formula.

Specifically, for the small localized changes in boundary conditions which are typical of user interaction, the algorithm is linear in the complexity of the boundary discretization. It is also fast in practice; we have demonstrated these results in our system ArtDefo. By precomputing and updating the solver's component matrices, we can quickly compute the deformation and contact forces during user interaction, with low latency. In addition, it is a direct solution method — this allows accurate prediction of solution costs which is essential for real time applications. Using our method, we believe it is now possible to simulate more realistic deformable models with real-time user interaction.

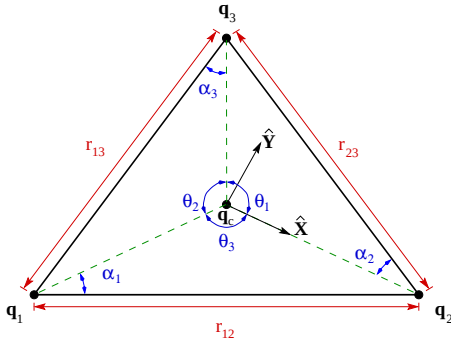


Figure 6: Notation

A Constant Element Influence Coefficients

When implementing boundary elements, there are always a number of singular integrals which the user must acquire or spend some time calculating. In the relatively simple case of triangular constant elements with centroid collocation, there are only a few integrals, and they are presented here for completeness.

A.1 Inter-element Effects

These are integrals corresponding to interactions where the load point lies at the centroid of a triangle other than the one being integrated over, i.e., $\mathbf{x}_i \notin \Gamma_j$. This includes the elements of the 3×3 matrices $\mathbf{g}_{ij}$ and $\hat{\mathbf{h}}_{ij}$, for $i \neq j$, and therefore corresponds to the majority of the integrals. Since $r = |\mathbf{x}_i - \mathbf{y}|$ is never zero, these are nonsingular integrals which may easily be calculated using standard numerical quadrature (see Brebbia [5]).

A.2 Self-effects

Self-effects correspond to the integrals in the diagonal terms of equation 11 such as $\mathbf{g}_{ii}$ and $\hat{\mathbf{h}}_{ii}$. Since the load point lies in the center of the triangle being integrated over, these are singular integrals, as the fundamental solutions are unbounded as $r \to 0$. The first integral is only weakly singular, while the second integral is strongly singular and only exists in a Cauchy principle value sense.

A.2.1 Calculation of $\mathbf{h}_{ii}$

Despite $\hat{\mathbf{h}}_{ii}$ being strongly singular, it is easy to calculate indirectly using rigid body translations by considering a bounded object with all nodes subjected to any arbitrary constant displacement boundary conditions, $\mathbf{u}_j = \bar{\mathbf{u}}, \forall j$. Since this body necessarily experiences no induced surface tractions, $\mathbf{p}_j = 0$, $\forall j$. It follows from (11) that

$$\mathbf{h}_{ii} = -\sum_{j \neq i} \mathbf{h}_{ij} \tag{16}$$

and therefore *neither* $\hat{\mathbf{h}}_{ii}$ *nor* $\mathbf{c}_i$ *need be calculated explicitly.* Note that (16) implies that H is a singular matrix.

A.2.2 Calculation of $\mathbf{g}_{ii}$

The elements of

$$(\mathbf{g}_{ii})_{kl} = \frac{1}{16\pi(1-\nu)G} \int_{\Gamma_i} \left(\frac{(3-4\nu)\,\delta_{kl}}{r} + \frac{r_k r_l}{r^3} \right) d\Gamma(\mathbf{y})$$

will be expressed for a triangle Δ, using the notation in figure 6, i.e., with vertices at $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, centroid at $\mathbf{q}_c$, area A, and an outward unit normal $\hat{\mathbf{n}}$. Omitting any constant factors, the first integral is J_1^Δ, and the second is

$$\hat{\mathbf{X}}_k \hat{\mathbf{X}}_l J_1^\Delta + \frac{1}{2}\left(\hat{\mathbf{X}}_k \hat{\mathbf{Y}}_l + \hat{\mathbf{X}}_l \hat{\mathbf{Y}}_k\right) J_2^\Delta + \left(\hat{\mathbf{Y}}_k \hat{\mathbf{Y}}_l - \hat{\mathbf{X}}_k \hat{\mathbf{X}}_l\right) J_3^\Delta.$$

with

$$\hat{\mathbf{X}} = \frac{\mathbf{q}_2 - \mathbf{q}_c}{|\mathbf{q}_2 - \mathbf{q}_c|}, \quad \hat{\mathbf{Y}} = \hat{\mathbf{n}} \times \hat{\mathbf{X}},$$

and

$$J_m^\Delta = \frac{2A}{3}\left[\frac{J_m(\theta_1, \alpha_2, 0)}{r_{23}} + \frac{J_m(\theta_2, \alpha_3, 0)}{r_{31}} + \frac{J_m(\theta_3, \alpha_1, \theta_1 + \theta_2)}{r_{12}} \right]$$

where

$$J_1(\Delta\theta, \alpha, \theta_{\min}) = \ln\left[\frac{\tan\left(\frac{\Delta\theta + \alpha}{2}\right)}{\tan\left(\frac{\alpha}{2}\right)} \right]$$

$$J_2\left(\Delta\theta,\alpha,\theta_{\min}\right)=\sin\left(\frac{\theta_{\min}-\alpha}{2}\right)\ln\left[\frac{\tan\left(\frac{\Delta\theta+\alpha}{4}\right)}{\tan\left(\frac{\alpha}{4}\right)}\right]$$
$$+\cos\left(\frac{\theta_{\min}-\alpha}{2}\right)\ln\left[\frac{\left[1-\tan\left(\frac{\alpha}{4}\right)\right]\left[1+\tan\left(\frac{\Delta\theta+\alpha}{4}\right)\right]}{\left[1+\tan\left(\frac{\alpha}{4}\right)\right]\left[1-\tan\left(\frac{\Delta\theta+\alpha}{4}\right)\right]}\right]$$

$$J_3\left(\Delta\theta,\alpha,\theta_{\min}\right)=2\sin\left(2\theta_{\min}-\alpha+\frac{\Delta\theta}{2}\right)\sin\left(\frac{\Delta\theta}{2}\right)$$
$$-\sin^2\left(\theta_{\min}-\alpha\right)\ln\left[\frac{\tan\left(\frac{\alpha}{2}\right)}{\tan\left(\frac{\Delta\theta+\alpha}{2}\right)}\right].$$

References

[1] James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. In Andrew Glassner, editor, *Computer Graphics (SIGGRAPH 94 Conference Proceedings),* pages 75–84. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[2] Kendall Atkinson. The numerical solution of boundary integral equations. In I. Duff and G. Watson, editors, *The State of the Art in Numerical Analysis*, Clarendon Press, pages 223–259, 1997.

[3] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 303–308, July 1992.

[4] David Baraff and Andrew Witkin. Large steps in cloth simulation. In Michael Cohen, editor, *Computer Graphics (SIGGRAPH 98 Conference Proceedings)*, pages 43–54. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[5] C. A. Brebbia, J. C. F. Telles and L. C. Wrobel *Boundary Element Techniques: Theory and Applications in Engineering*, Springer-Verlag, New York, 1984.

[6] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, August 1996. Proceedings of Eurographics '96. ISSN 1067-7055.

[7] George Celniker and Dave Gossard. Deformable curve and surface finite elements for free-form shape design. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Conference Proceedings)*, volume 25, pages 257–266, July 1991.

[8] David T. Chen and David Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 89–98, July 1992.

[9] S. L. Crouch and A. M. Starfield *Boundary Element Methods in Solid Mechanics*, Unwin Hyman Inc., London, 1990.

[10] Sarah F. Gibson and Brian Mirtich. A survey of deformable models in computer graphics. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November 1997.

[11] Sarah F. F. Gibson. 3D chainmail: a fast algorithm for deforming volumetric objects. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 149–154. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3.

[12] G. H. Golub and C. F. Van Loan *Matrix Computations* . Second Edition, The John Hopkins University Press, Baltimore, 1989.

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

[14] W. W. Hager. Updating the Inverse of a Matrix. In *SIAM Review*, volume 31, no. 2, pages 221-239, 1989.

[15] Friedel Hartmann. *Introduction to Boundary Elements: Theory and Applications*. Springer-Verlag, Berlin, 1989.

[16] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J, McDonald and J. Schweitzer and W. Stuetzl. Piecewise Smooth Surface Reconstruction. *Computer Graphics (SIGGRAPH 94 Conference Proceedings)*, pages 24–29, July 1994.

[17] James T. Kajiya. The rendering equation. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH 86 Conference Proceedings)*, volume 20, pages 143–150, August 1986.

[18] J. H. Kane, B. L. Kumar and R. H. Gallagher Boundary Element Direct Reanalysis for Continuum Structures In *Journal of Engineering Mechanics*, volume 118, pages 1679-1691, 1992.

[19] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. In *Computer Graphics (SIGGRAPH 95 Conference Proceedings)*, pages 55–62, August 1995.

[20] Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and error estimates for radiosity. In Andrew Glassner, editor, *Computer Graphics (SIGGRAPH 94 Conference Proceedings),* pages 67–74. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[21] A. E. H. Love. *A Treatise on the Mathematical Theory of Elasticity*. Dover, New York, 1944.

[22] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 309–312, July 1992.

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

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

[25] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In John Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 269–278, August 1988.

[26] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 205–214, July 1987.

[27] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (from goop to glop). In *Proceedings of Graphics Interface '89*, pages 219–226, June 1989.

[28] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In Andrew Glassner, editor, *Computer Graphics (SIGGRAPH 94 Conference Proceedings)*, pages 43–50. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[29] Keith Waters. A muscle model for animating three-dimensional facial expression. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 17–24, July 1987.