TOWARD REAL-TIME, PHYSICALLY-CORRECT SOFT TISSUE BEHAVIOR SIMULATION

Dan Koppel and Yuan-Fang Wang[*]

Department of Computer Science
University of California
Santa Barbara, CA 93106

Shivkumar Chandrasekaran

Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106

ABSTRACT

We present a behavior simulation algorithm that has the potential of enabling physically-correct, photo-realistic, and real-time behavior simulation for soft tissues and organs. Our approach combines a physically-correct formulation based on boundary element methods with an efficient numeric solver. This approach can have a significant impact in off-line surgical training and simulation, and in on-line computer-assisted surgery.

1. INTRODUCTION

In this paper, we describe a technique to accelerate the computations involved in modeling deformations of organs and soft tissues, within a medical context. There are many scenarios, both in off-line training of surgeons and on-line computer-assisted surgery, that the proposed technique can be useful. For example, simulators, used for the purpose of training surgeons in the pre-operative stage, require physically-correct, real-time response of the graphical rendering to inputs from the trainee. However, it has long been recognized that it is difficult to simultaneously satisfy the requirements of physical-correctness and real-time.

In some recent work [1], advantage is taken of the fact that certain approximations can be made when modeling deformable objects. More specifically, assumptions about isotropy and homogeneity of the material can be made with little penalty in accuracy. These special properties allow a volume-based integral equation to be integrated by parts, rendering a surface-based integral equation, and thus substantially fewer nodes after discretization.

We follow a similar mathematical model, but add a novel algorithm developed in [2, 3, 4]. It should be stated that reformulation of the problem into a surface-based integral equation is not sufficient for real-time performance, as the authors of [1] have shown. Their way of creating additional speedup is based on a "low-rank update" method. Although real-time performance is attained, this method is based on the assumption that the deformable object in question is being poked in a "point-like" manner, rather than along a broad surface area. While there may be situations where this holds, it is fairly restrictive, especially in a surgical environment, where organ-organ or organ-wall interactions occur frequently.

The method we have developed does not make the point-like assumption. The elimination of this restriction is useful and can aid in simulating a much wider range of medical procedures, since substantial and widespread time-varying boundary conditions will now be allowed. Simulator systems employing this algorithm will be able to render in real-time scenarios in which organs may come in full contact with other organs or cavity wall.

[*]The research was support in part by Karl-Storz Imaging, Inc. and the State of California Micro Program.

The algorithm developed in [2, 3, 4] is used as a replacement to the "low-rank update" method used in [1]. As stated above, its advantage is that it is able to rapidly process a changing boundary condition which, rather than being point-like, is widespread. The main feature is that it takes into account the fact that the governing matrix derives from a physically-based pairwise interaction, known mathematically as a Green's function. The Green's function, resulting from real-world interactions, has the nice properties of locality and smoothness, which allow the discretized version of it (a matrix) to be expressed in terms of low-rank components. Operations such as matrix inversion can then be accomplished in a much more efficient manner (both in time and space). But for this method, the use of low-rank components does not create the point-like restriction mentioned above.

2. BACKGROUND

A tissue or organ model must capture faithfully the *structure* and *behavior* traits. The structure of a organ can be reconstructed using, say, the Visible Human Dataset. Many algorithms are applicable here, such as the "marching cubes" algorithm and others for constructing surface representation from CT slices or point cloud data of a laser range finder, and the more recent radial basis function-based interpolation and extrapolation techniques. While it might not prove difficult to model the *static structure* of a soft organ correctly, to model the *dynamic behavior* of an organ is a much difficult task. The difficulty is mainly due to two factors: (1) the structure complexity of the model required for physically correct behavior simulation, and (2) the time complexity in solving the resulting governing PDE equations.

It is well known that simulation of deformable behaviors is difficult. So far, approaches to this problem can be roughly classified into two categories: those that aim more at efficiency and those that aim more at accuracy. The former is categorized by many mass-spring, spline, and superquadric models while the latter mainly comprises techniques based on the finite element methods (FEM).

Efficient models usually employ a small number of model control parameters (e.g., spring constants in the mass-spring models and control vertex positions in the spline models). Fewer parameters allow for faster simulation. However, it can often be difficult to control fine-level details with few tunable parameters. Furthermore, there is no guarantee that the simulation will be truthful if the model parameters do not encode the material property in a physically meaningful way, and this, is often very difficult or impossible to achieve. For example, if the deformable object is made of incompressible material, the volume should be preserved at all times. Pressure on one side of the object should induce swelling on some other side. This type of "action at a distance" phenomenon is often hard to simulate correctly using the efficient models.

On the other hand, accurate models based on FEM allow for realistic simulation of material properties. Constitutive equations that capture the relation between stress and strain enable faithful simulation. However, numeric solution of the resulting PDE often requires fine discretization of the domain, which gives rise to large and time consuming matrix inversion problem and limits the real-time performance.

Our modeling scheme aims to achieve the best of both worlds by providing necessary accuracy at a speed comparable to that of the efficient models. We use a general class of numeric simulation methods that are called "boundary element methods" or BEM [5]. Naively speaking, BEM for structure simulation concentrates the analysis power on the boundary of the object (or the surface of a 3D organ). Intuitively, for the same level of discrete resolution, the BEM-based methods have the potential of being significantly less expensive than their FEM-based counterparts. This is because that the FEM methods employ $O(n^3)$ variables (representing the displacement of the body at a particular point under externally applied forces and torques), scattered both in the interior and on the boundary of the object. The BEM methods employ $O(n^2)$ variables (representing surface displacement and traction) only on the boundary of the object, with n representing the resolution along a particular dimension. Hence, BEM achieves one order of magnitude in saving in terms of problem size.[1]

While this simple analysis might look promising, the reality is never this straightforward. No matter it is an FEM- or a BEM-based simulation, the gist of the simulation all comes down to solving a system of linear equations of the form $\mathbf{AX} = \mathbf{B}$ over time (or a time marching problem), where $\mathbf{A}$ involves the material properties such as the Lamé constants λ and μ (and other quantities that have to do with the discretization and interpolation functions in the elements), $\mathbf{B}$ involves known boundary conditions (e.g., known displacement and traction at certain surface and interior points), and $\mathbf{X}$ are the unknown displacement and traction inside and on the surface of the object.

Numeric analysts will quickly point out that while BEM requires less number of variables — which results in a much smaller system of equations ($O(n^2)$ for BEM vs. $O(n^3)$ for FEM) — the real complexity of the BEM solution can be higher. This is because that while FEM results in a bigger matrix $\mathbf{A}$, $\mathbf{A}$ is often well conditioned and sparse. Efficient solutions exist for many classes of well conditioned, sparse matrices. which bring the complexity of solution to $O(n^3)$. BEM, on the other hand, always results in a dense matrix $\mathbf{A}$ [5], and the complexity of the solution can be proportional to the cube of the matrix size. This results in a total complexity of $O(n^6)$ which is even more expensive than FEM. Hence, without an efficient numeric solution method BEM is just a "castle-in-the-sky" or "door-painted-on-the-wall" solution.

An efficient solution was recently proposed by James and Pai in [1]. They have adopted a method that allows the inverse of $\mathbf{A}$ to be computed once and off-line. When the simulation is running, they employ a low-rank update algorithm (due to Sherman, Morrison, and Woodbury [6]) which efficiently updates the inverse of the $\mathbf{A}$ matrix. The low-rank update assumption is valid if only a small number of boundary conditions change over time. E.g., the surgeon slides a sharp instrument across the surface of an organ, but the organ is otherwise undisturbed. In this case, only a few contact points (or boundary conditions) change. They were able to demonstrate real-time performance for fairly complicated deformable simulation.

However, While this approach partially solves the efficiency problem, it does restrict the type of collision an organ can experience. This is because in cramped body cavities, low-rank update is generally not a valid assumption. Organ-organ collision and organ-body wall collision can drastically change the boundary conditions and violate the "low-rank" assumption. When the low-rank assumption no longer holds, the matrix inversion becomes the dominant computation and is as expensive as $O(m^3)$, where m is the number of nodes being perturbed (e.g., those in contact with other organs or the body wall).

On the other hand, our proposed numeric algorithm (to be discussed in more detailed in Sec. 4) exploits a different mechanism to reduce update complexity. Our observation is that the kernel function in the fundamental solution of the BEM is usually smooth, which results in the coefficient matrix having a block-wise low-rank structure, which we call sequentially semi-separable (SSS) for the one-dimensional case, and hierarchically semi-separable (HSS) for higher dimensions. Exploiting this particular matrix structure in our simulation, we are able to achieve real-time behavior simulation on ordinary PC of fairly complex organs (Sec. 5 provides timing and simulation results). Our simulation allows large changes in boundary conditions, such as those resulted from organ-organ and organ-body wall collision, which is not possible in [1]. This is a significant improvement that increases the applicability of the BEM methods.

3. BEM-BASED FORMULATION

Our organ deformation modeling is based on several physical concepts: First is the very definition of how an object is deformed. By defining a vector $\mathbf{u}$ at every location which denotes the displacement of a body point from its unperturbed, resting position, we can specify how much the object has moved or deformed.

Next, the idea of strain is defined: $\epsilon_{ij} = \frac{1}{2}(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j} + \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}_i}) \quad 1 \leq i, j \leq 3$. This expression (which is a tensor expression representing nine quantities) captures how much directional compression occurs at each location along the three axial direction $x_i, 1 \leq i \leq 3$. The term "directional" is used to indicate that an object may be stretched in one direction, while squeezed in some other direction. The idea of stress (σ_{ij}) is closely related and similarly is described by a tensor expression. While strain measures movement, stress measures force.

For the types of materials that we are modeling the two quantities are related through a linear relationship called the "constitutive equation": $\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + 2\mu\epsilon_{ij}$, where δ_{ij} is the Kronecker delta. The Lamé constants λ and μ describe the material properties and can model either rigid or flexible objects. In addition, they can capture the "averse-ness" of the material to changes in volume or to changes in shape. The next equation forms the foundation of much of continuum mechanics. It describes the balancing of internal and external forces: $\frac{\partial \sigma_{ij}}{\partial \mathbf{x}_j} + \mathbf{b}_i = 0$, where $\mathbf{b}$ denotes the force per unit volume exerted on the body. Based on it, an impor-

[1] One might argue that there are ways to reduce the problem size for FEM, e.g., multi-resolution and adaptive grid. We are aware of the possibilities. The above analysis serves only as an illustration and does not mean to ignore these possibilities. Furthermore, similar effi cient numeric techniques are often applicable to BEM to achieve a corresponding reduction in problem size as well.

tant identity, known as Somigliana's identity, can be derived. Lack of space prohibits a derivation of the identity here, but it can be established in a series of straightforward steps [5].

$$\begin{aligned} & \mathbf{c}(\mathbf{x})\mathbf{u}(\mathbf{x}) + \int_\Gamma \mathbf{p}^*(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y})d\Gamma(\mathbf{y}) \\ = \ & \int_\Gamma \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{p}(\mathbf{y})d\Gamma(\mathbf{y}) + \int_\Omega \mathbf{u}^*(\mathbf{x},\mathbf{y})\mathbf{b}(\mathbf{y})d\Omega(\mathbf{y}) \end{aligned} \tag{1}$$

and Γ and Ω denote the boundary and interior of the object, respectively. The discrete version of the identity is the "workhorse" behind the boundary element methods.

$$\begin{aligned} & \tfrac{1}{2}\mathbf{u}_i + \textstyle\sum_{j=1}^N \left(\int_{\Gamma_j} \mathbf{p}^*(\mathbf{x}_i,\mathbf{y})\mathbf{u}(\mathbf{y})d\Gamma(\mathbf{y}) \right) \mathbf{u} \\ & = \textstyle\sum_{j=1}^N \left(\int_{\Gamma_j} \mathbf{u}^*(\mathbf{x}_i,\mathbf{y})\mathbf{u}(\mathbf{y})d\Gamma(\mathbf{y}) \right) \mathbf{p} \end{aligned} \tag{2}$$

where $\mathbf{c}$ is a (known) function that depends only on the geometry of the surface, and again $1 \le i \le 3$. The main feature of this identity is that it involves unknowns ($\mathbf{u}$ denotes unknown displacement and $\mathbf{p}$ denotes the unknown traction—surface force per unit area) only on the surface of the deformed object. It also makes use of some very useful and established functions, known as "Kelvin's fundamental solutions" ($\mathbf{u}^*$ and $\mathbf{p}^*$) [5]. These functions play the mathematic role of the kernel functions and are essential in converting the volume-based formulation into a surface-based one. These kernel functions describe the simplest possible deformation in a material and are useful to describe any general, complex deformation. They are expressed, in our problem, as

$$u^*_{ij}(\mathbf{x},\mathbf{y}) = \frac{1}{16\pi(1-\nu)G}\left\{\frac{(3-4\nu)\delta_{ij}}{r} + \frac{r_i r_j}{r^3}\right\} \tag{3}$$

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

where $G(=\mu)$ is the shear modulus and $\nu(=\frac{\lambda}{2(\lambda+\mu)})$ is the Poisson ratio. $\mathbf{r} = \mathbf{x} - \mathbf{y}$, $r = |\mathbf{r}|$, $\frac{\partial r}{\partial \mathbf{n}(\mathbf{y})} = \frac{\mathbf{r}\cdot\mathbf{n}(\mathbf{y})}{r}$, $\mathbf{n}$ is the normal vector at a surface point, and again $1 \le i, j \le 3$. The above equations give the perturbation effect of a spatially concentrated force applied at a location $\mathbf{x}$ in the direction i would have at a location $\mathbf{y}$ in the direction j. While the expression might look complicated, the important thing is that there is an analytical solution to Kelvin's problem and the solution decays away from the perturbation point and is smooth except when $\mathbf{x} = \mathbf{y}$. This is an important property that we will use in our fast solver, which is described below.

4. FAST DIRECT SOLVER

A key to our approach is the exploitation of a new class of direct solvers that are capable of drastically reducing the run-times of current deformable body simulation techniques. This is accomplished by significantly speeding up the solution of the resulting system of linear equations **AX=B** ***without*** the restrictive low-rank update assumption used by James and Pai [1].

The key idea behind the new fast solvers is that the coefficient matrix $\mathbf{A}$ has a certain "low-rank structure." In Figure 1 we give a pictorial depiction of this low-rank structure for an elongated deformable body. In this figure every off-diagonal sub-matrix (without a dot in it) has small numerical rank independent of the size of the original matrix. For more complicated bodies, the picture looks more complex, but the essential idea continues to hold true: large portions of the off-diagonal blocks have a low numerical rank, independent of the size of the matrix.

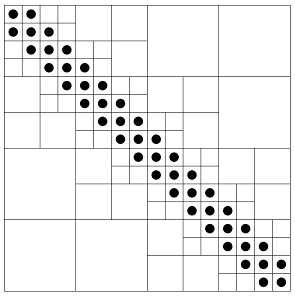

Fig. 1. General structure of the coefficient matrix.

This structure was first exploited in the classical Fast Multipole Method of Greengard and Rokhlin. [7]. Recently Chandrasekaran and Gu have made the observation that we can essentially solve these same equations in nearly linear time. Their observation is based on the fact that the solution for the elasticity equations can be described in terms of a Green's function, and that this Green's function is smooth away from the diagonal. Hence we can expect $\mathbf{A}^{-1}$ to exhibit the same low-rank off-diagonal structure. The mechanics of this algorithm are far too complicated to be presented here. Therefore we will try to motivate it by linking it to some work on sparse matrices.

If we look at the kernel of the integral operator (Eqs. 3 and 4) for the BEM method, it is readily apparent that the interaction between points $\mathbf{x}$ and $\mathbf{y}$ increases as the points get closer. Now, suppose we artificially set to zero all those entries in $\mathbf{A}$ other than those that correspond to nearest neighbor interactions. Call this new matrix $\mathbf{B}$. Then, it is intuitively clear that $\mathbf{B}$ will look (structurally) very much like a sparse matrix obtained by discretization of a partial differential operator on the surface of the body [5]. The reason is that, we usually discretize partial differential operators by using a stencil that covers the nearest neighbors on the grid (e.g., $\frac{\partial^2 f_{i,j}}{\partial x^2} \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta x^2}$). Also, it is clear that the entries we are setting to zero in $\mathbf{A}$ will not be the largest entries in the matrix. Hence this resulting sparse matrix $\mathbf{B}$ can serve as a reasonable approximation of $\mathbf{A}$. Using standard sparse matrix theory, one can then claim that this sparse approximation to $\mathbf{A}$ can be solved (or factored if you will) efficiently.

Let us go back to $\mathbf{B}$, the sparse approximation to $\mathbf{A}$. Chandrasekaran and Gu have developed a new class of fast direct solvers that solve linear systems involving sparse matrices similar to $\mathbf{B}$ in linear time. In Table 1 we show some representative timings obtained for their algorithm on a 1.6GHz, 2GB Pentium 4 machine for similar types of sparse matrices.[2]

Observe that the algorithm is performing at real-time speeds even for 3969 unknowns, which translate into roughly 1300 surface nodes or triangles (because each node is represented by a three-wide vector—3D displacement or traction). Modern desktops are capable of running at much higher-speeds and we expect to be able to solve even larger systems in real-time.

[2]These sparse matrices were for flat grids, whereas $\mathbf{B}$ corresponds to a grid on a curved surface. But this is not critical to the algorithm.

Table 1. Preliminary timing results

Number of unknowns	CPU run-time in seconds
961	0.01
3,969	0.03
16,129	0.15
65,025	0.81
261,121	3.80
1,046,529	17.00

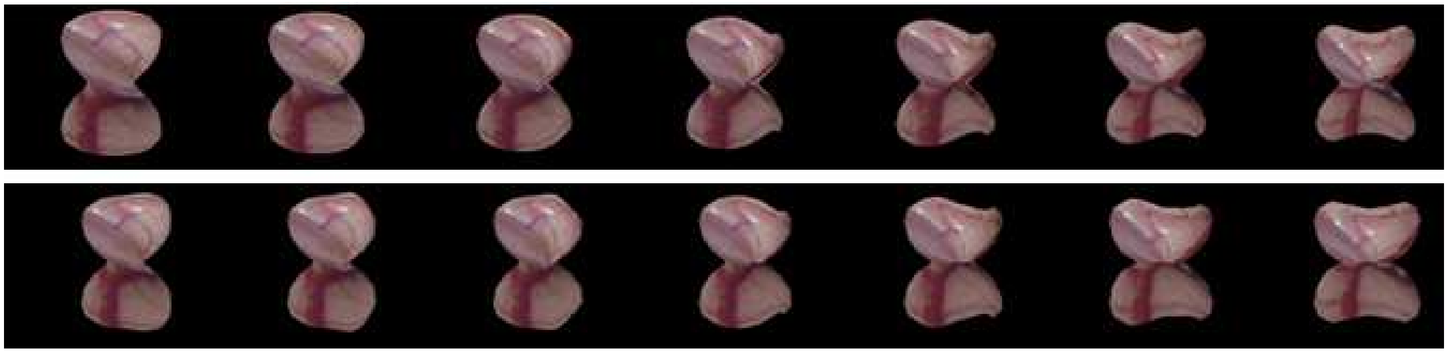

Fig. 2. Deformable animation using LAPACK/Blas solver (upper) and our fast solver (lower) on the same cardioid model with 832 triangles.

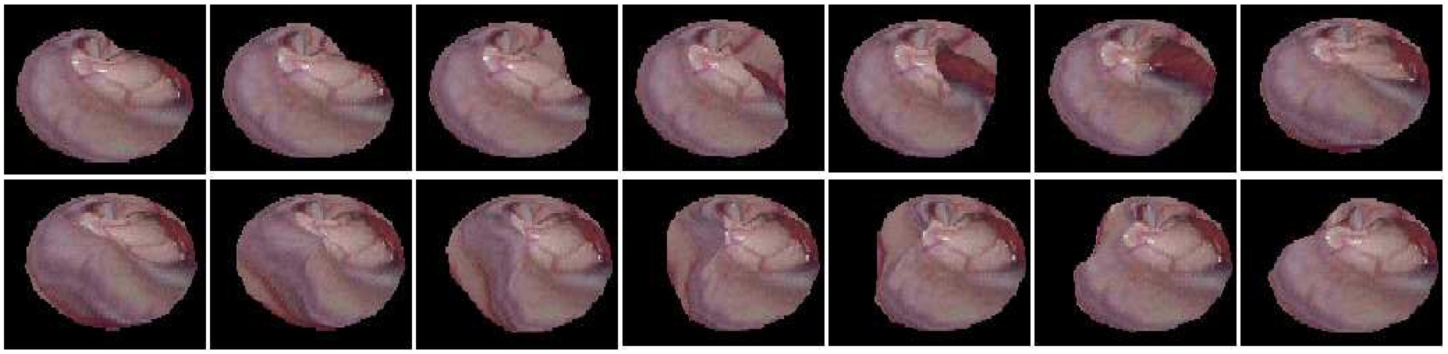

Fig. 3. Sample graphic simulation results using our BEM formulation and fast solver. The sequence is shown left to right and top to bottom.

Anyway, it is clear from this table that the new solvers by Chandrasekaran and Gu can solve sparse matrices of the structure possessed by $\mathbf{B}$ in nearly linear time. Hence, there is some reason to believe that it should be possible to solve (or factor if you will) $\mathbf{A}$ itself in near linear time. However, the linear-time algorithm is too complicated to explain over here, and we refer interested readers to the papers instead [2, 3, 4].

5. EXPERIMENTAL RESULTS

Fig. 2 shows the animation results of poking a cardioid with a large, extended surface contact moving westward along the middle of the shape. The same BEM formulation was used for the upper and lower rows, and the difference was in the numeric solvers employed for matrix inversion. The upper row depicts the results using the LAPACK/Blas solver, while the lower row the results of our fast solver. Blas solver uses standard Gaussian elimination for matrix inversion, which has a complexity of $O(n^3)$. For this particular object with 832 triangles, our solver was running at real time while Blas was about 300 times slower (9.1 seconds/frame). As can be seen from Fig. 2 the results look qualitatively similar.

Fig. 3 shows another animation sequence using our BEM formulation and (yet unoptimized) fast solver on a disk-shaped organ. The simulation was done in real time with a large perturbation went along the equator from right to left. We were able to simulate this effect realistically with the volume preserved.

6. CONCLUSION

This paper presents our formulation of a real-time, physically-correct behavior simulation algorithm for soft tissues and organs. Our future work is to construct a surgical simulator to validate such an algorithm and to apply it for on-line computer-assisted surgery.

7. REFERENCES

[1] D. L. James and D. K. Pai, “Artdefo - accurate real time deformable objects,” in *Siggraph 1999, Computer Graphics Proceedings*, Alyn Rockwood, Ed., Los Angeles, 1999, pp. 65–72, Addison Wesley Longman.

[2] S. Chandrasekaran and M. Gu, “Fast and stable algorithms for banded plus semi-separable matrices,” *SIAM J. Matrix Anal. Appl.*, vol. 25, no. 2, pp. 373–384, 2003.

[3] S. Chandrasekaran and M. Gu, “A fast and stable solver for recursively semi-separable systems of equations,” in *Structured matrices in mathematics, computer science and engineering, II*. 2001, AMS Publications.

[4] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A. van der Veen, “Fast stable solvers for sequentially semi-separable linear systems of equations,” Tech. Rep., Department of mathematics, UC Berkeley, 2003.

[5] James H. Kane, *Boundary Element Analysis in Engineering Continuum Mechanics*, Prentice Hall, Englewood Cliffs, NJ, 1994.

[6] G. H. Golub and C. F. van Loan, *Matrix Compujtations, 2nd Ed.*, John Hopkins University Press, Baltimore, MD, 1996.

[7] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *J. Comp. Phys.*, vol. 73, pp. 325–348, 1987.