Dynamic Simulation and Virtual Control of a Deformable Fingertip

Dan Reznik
University of California-Berkeley
dreznik@cs.berkeley.edu

Christian Laugier
LIFIA-INPG, Grenoble, France
laugier@imag.fr

***Abstract*—**
An efficient computational model for the dynamics of a deformable robot fingertip is presented. The dynamic model is based on a discretization of the fingertip's volume into a lattice of masses locally interconnected by damped springs. The lattice's parameters are adjusted in correspondence with bulk properties of the fingertip's deformable material (rubber). In the task studied, the fingertip moves toward a rigid flat surface, contacts it, and presses against it. This motion is commanded by an external feedback controller which communicates with the dynamic model through a *virtual control interface*: The controller applies forces and torques to the dynamic model and the dynamic model responds in real-time with position/velocity/force feedback information. In this fashion, the controller interacts with the fingertip's model in the same way it would interact with the actual physical system. This type of paradigm is envisioned as a prototyping/testing tool in the design of control systems for deformable objects as well as for applications involving the haptic (i.e., sensorially realistic) interaction between a human and a virtual (deformable) object. Graphical snapshots of a real time simulation of the task under study are presented which reveal the physical and computational plausibility of the model.

I. Introduction

In this paper we address the problem of real-time dynamic simulation and control of a "virtual" deformable object. The object is virtual in the sense that it only exists as an internal computational model. The object studied is the fingertip of a robot hand, shown as a 3d model in Figure 1. The fingertip consists of a rigid *rod* partly covered by a soft rubbery *tip*. The rod is attached to the endpoint of a robot, e.g., the last link on a dextrous hand's finger. The following task is studied, Figure 2: First, the finger is placed near a rigid flat surface at a certain pose. It then starts a uniform downward motion toward the surface, and at some point contact is established. The downward motion proceeds up to a certain "depth" beyond contact, causing the tip to deform. The fingertip's motion is commanded by an external PID controller [1], interfaced to the dynamic model via a *virtual control interface*, shown in Figure 3. The controller applies forces and torques to the virtual fingertip so as to maintain its pose and speed constant. The latter responds with position, velocity, and force feedback data. This design is such that the controller interacts with the fingertip's dynamic model in the exact fashion it would interact with the actual physical system – this type of setup is called here *virtual control.*

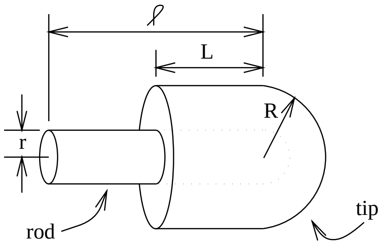

Fig. 1. Geometric model of the deformable finger. The finger is composed of a rigid *rod* and an external deformable *tip*. The rod is the union of a cylinder of radius r and length l and a hemisphere of radius r. The tip is the union of a cylinder of length L and a hemisphere of radius R. Both hemispheres are concentric.

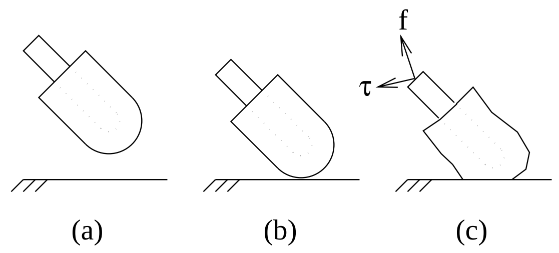

Fig. 2. In the task being studied the fingertip is (a) initially placed near a rigid surface; it is then propelled with constant speed and orientation toward that surface. In (b) the contact is initially established; the motion proceeds causing the tip to deform (c). The deformation generates a force f and torque τ measured at the rod's endpoint, which must be balanced by an external controller.

The problem we address is how to correctly and efficiently model tip deformations and how to measure position, velocity, and force torque at the rod's endpoint so this data can be fed oback in real time to the motion controller. We are motivated by the difficulties in building/testing robust motion controllers for deformable objects. Typically, a controller's parameters must be set in close relationship to the (ofter non-linear) dynamic behavior of the system being controlled. For example, certain linearizing control laws implement an "inverse" of the system being controlled [1]. However, the complex geometry changes of deformable objects makes their dynamic behavior hard to model.

Control of deformable objects appears in many fields in robotics such as flexible manipulator design, automated surface finishing, and grasping with soft fingers. In the latter, one often exploits the larger areas of soft contact to add to the robustness of a grasp [2, 3]. In this paper however, we do not address any application-specific issues and focus solely on dynamic modeling of deformation and the control interface. The main contribution is to present an architecture for *virtual control prototyping* whereby a controller can interact with a virtual (i.e., computer simulated) deformable object in the exact same way it would interact with the physical object. We envision this type of architecture as a computer-aided tool for the virtual testing and debugging of motion controllers. This type of design can also be used in virtual reality, haptic interfaces, where the controller is replaced by a human operator interfaced to the dynamic model via a haptic transducer [4, 5]. Applications in this area include force-feedback robotic teleoperation, video games, remote surgery, and interactive solid modeling/sculpting [6, 7, 8].

Dynamic modeling of deformability has been much studied in the fields of computational solid mechanics [9] and computer graphics [10, 11]. In the former, one is typically interested in steady-state stress and strain distributions of a deformable body under external loads, where in the latter, deformation is seen as a tool for producing realistic-looking animations requiring little or no external specification. However, the above computational models suffer from both symbolic and computational complexity. To address that, we propose a physically-based dynamic model targeted at both ease of implementation and computational efficiency. The model is based on a discretization of the tip's volume into a regular lattice of *particles* locally interconnected by damped springs. The model is physically-based since deformations of a material are caused at the microscopic level by visco-elastic interactions between particles (molecules). In particular, for a class of elastic, isotropic materials, molecules are organized as regular lattices where the interaction is essentially local.

In our model, the lattice's internal parameters such as spring/damping constants are adjusted to approximate the bulk material properties of rubber, such as its density, internal energy dissipation, and elastic moduli [12]. The rod is modeled as a rigid body, governed by rigid body dynamics [13]. The rod is "attached" to the tip lattice by a few "ligament" springs which link particles in the tip to points evenly distributed over the rod's surface. A similar approach to combining a rigid model with a deformable discrete system has been used to model the motion of skin and clothes in response to the motions of a rigid skeleton [14].

Contact between the fingertip and the flat surface is modeled as non-slipping with infinite friction [1]. During fingertip motion, particles in the lattice's boundary coming in contact with the surface are "anchored" down at the point of contact – in this fashion, the dynamics of deformation emerges from the accumulation of anchored masses against the surface.

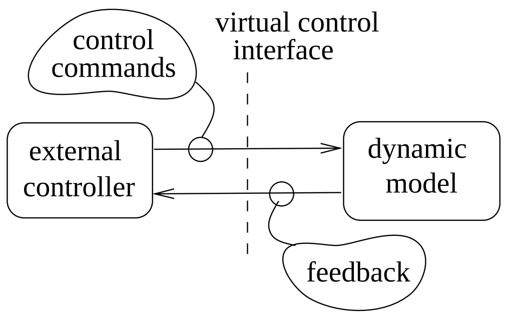

Fig. 3. The virtual control interface. The external controller sends commands (e.g., forces, torques) to the dynamic model on the right, which in turn feeds simulation-specific information (e.g., measured positions & forces) back to the controller. This design is such that the dynamic model can be transparently replaced by its real-world counterpart.

Related work: Discrete particle-based paradigms as models for deformable continua were originally proposed in 1940s. They have recently regained popularity amongst robotics and dynamic simulation investigators [15, 16, 17, 14]. The main advantages of this paradigm are (1) its physically-based appeal, (2) the simplicity of implementation and generality, and (3) computational efficiency: dynamic updates and collision detection, regarded as the major bottleneck in dynamic simulations [18], can be computed in linear time in the number of particles in the lattice.

This paper is organized as follows: in Section II the dynamic model of the deformable fingertip is explained. Section III describes how the fingertip is controlled to execute its task. In Section IV, a simulation of the experiment is presented. Section V presents conclusions and directions for future work.

II. Modeling a deformable fingertip

A. Particle systems

A *particle system* , is a tuple $(\mathcal{P}, \mathcal{S})$ made up of a set $\mathcal{P}$ of *particles* and a set $\mathcal{S}$ of *springs*. Each particle $p \in \mathcal{P}$ is a tuple $(m, \mathbf{x}, \mathbf{v}, \mathbf{f})$ made up of a scalar m encoding the particle's *mass*, and three 3d-vectors $\mathbf{x}$, $\mathbf{v}$, and $\mathbf{f}$ encoding respectively the particle's *position*, *velocity*, and *external forces*. Each spring $s \in \mathcal{S}$ is a tuple (i, j, k, μ, l_0) of two integers a and b indicating that particles $p_i, p_j \in \mathcal{P}$ are connected by the spring, and three scalars k, μ, and l_0 encoding respectively the spring's *elastic constant*, *damping factor*, and *rest length*.

A spring s applies forces to its endpoint particles p_i, p_j whenever the distance between these deviates from the spring's rest length l_0. In the adopted model, the applied force is proportional to the length deviation (linear elasticity), discounted by a damping term proportional to the rate of length change (viscous damping).

Let $\mathbf{d} = \mathbf{x}_j - \mathbf{x}_i$ be the vector between spring s's endpoints, and $\hat{\mathbf{d}} = \mathbf{d}/||\mathbf{d}||$ be the unit vector in that direction. If p_i and p_j are translating with velocities $\mathbf{v}_i$ and $\mathbf{v}_j$, the instantaneous change in spring length $||\dot{\mathbf{d}}||$ will be given by $(\mathbf{v}_j - \mathbf{v}_i) \cdot \hat{\mathbf{d}}$ [15]. Let $\mathbf{u}_i$ and $\mathbf{u}_j$ be the forces spring s applies to its endpoint particles p_i and p_j; Then, under the accepted force model:

$$\begin{aligned} \mathbf{u}_i &= [k(||\mathbf{d}|| - l_0) + \mu||\dot{\mathbf{d}}||]\hat{\mathbf{d}} \qquad (1) \\ \mathbf{u}_j &= -\mathbf{u}_i \end{aligned}$$

Particle motion is ruled by the following differential equations, given by Newton's Law [13]:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v} \qquad (2) \\ \dot{\mathbf{v}} &= \mathbf{f}/m \qquad (3) \end{aligned}$$

Where vector $\mathbf{f}$ represents the sum of spring forces acting on particle p at a given moment. This quantity is "accumulated" for each particle according to procedure *AccumulateForces*, shown in Figure 4.

B. Numeric integration

The *dynamic state* of a particle system is a snapshot of particles' positions and velocities at a given time t. Given a small interval δt, Eqns. (2) and (3) can be numerically integrated so as to compute a new dynamic state at time $t + \delta t$. For simplicity's sake, we use Euler's method of numeric integration which amounts to the following first-order approximation of the integral of a function $\dot{g}(t)$ [19]:

$$g(t + \delta t) = g(t) + \dot{g}(t)\delta t \qquad (4)$$

The error incurred by this integration method is of order $O(\delta t^2)$ [19]; this fact is used to select a particular δt which will keep integration errors under a certain tolerance. A better alternative would be to use a numeric integrator with an automatic step-size control, such as a variable step Runge-Kutta [20], or the method proposed by Joukhadar [15] whereby the integration step is dynamically modified based on the amount of spurious energy added or removed from the particle system by the integration error.

```
Proced.:  AccumulateForces( , )
Input:    Particle system , = (P, S)
Output:   Computes forces f, acting on each
          particle p ∈ P.
Begin
    For each $p = (m, \mathbf{x}, \mathbf{v}, \mathbf{f}) \in \mathcal{P}$ do:
        Let $\mathbf{f} = 0$;
    EndFor;
    For each $s = (a, b, k, \mu, l_0) \in \mathcal{S}$ do:
        Compute $\mathbf{u}_i$ and $\mathbf{u}_j$ according to (1);
        Let $\mathbf{f}_i = \mathbf{f}_i + \mathbf{u}_i$;
        Let $\mathbf{f}_j = \mathbf{f}_j - \mathbf{u}_j$;
    EndFor;
End.

Proced.:  EulerStep( , $\delta t$)
Inputs:   Particle system , = (P, S);
          Integration step $\delta t$
Output:   Updates x and v of each
          particle p ∈ P.
Begin
    For each $p \in \mathcal{P}$ do:
        Let $\mathbf{x} = \mathbf{x} + \mathbf{v}\delta t$;
        Let $\mathbf{v} = \mathbf{v} + \frac{\mathbf{f}}{m}\delta t$;
    EndFor;
End.

Proced.:  AnimationLoop( , $\delta t$, steps)
Inputs:   Particle system , = (P, S);
          Integration step $\delta t$
          Number of desired updates steps
Output:   Updates and displays particle system.
Begin
    Loop steps times:
        AccumulateForces( , );
        EulerStep( , $\delta t$);
        DisplayParticleSystem( , );
    EndLoop;
End.
```

Fig. 4. The procedures used for particle system dynamics.

The *EulerStep* procedure, shown in Figure 4, makes use of Eqns. (2,3,4) to update the position and velocity of all the particles.

A quasi-continuous evolution of the dynamic state can be visualized if the *AccumulateForces* and *EulerStep* procedures are called repeatedly, within a loop, as done by procedure *AnimationLoop*, shown in Figure 4.

The routine *DisplayParticleSystem*, used above, is responsible for the rendering of geometric primitives one wishes to visualize during the simulation, e.g., the boundary of the deformable tip and the rigid rod.

C. Volumetric sampling

The rubbery tip is modeled as a particle system , $= (\mathcal{P}, \mathcal{S})$. Particles are placed at regular 3d grid points within the tip's volume. Assume the center of the tip's hemisphere coincides with the origin and that the axis of its cylindrical part coincides with the z axis. Given three integers i, j, and k, define a 3d vector $\mathbf{w}_{ijk} = (w_i, w_j, w_k)^T = (Ai - R/2, Aj - R/2, Ak - L)^T$, where A is a discretization constant. Define set $\mathcal{P}$ as the union of all $\mathbf{w}_{ijk}$ falling within either the tip's hemisphere or its cylindrical portion, and outside the rod:

$$\begin{aligned}\mathcal{P} = \; & \{\mathbf{w}_{ijk} | r < ||\mathbf{w}_{ijk}|| \leq R, w_k \geq 0\} \cup \\ & \{\mathbf{w}_{ijk} | r < \sqrt{w_i^2 + w_j^2} \leq R, -L \leq w_k < 0\}\end{aligned}$$

Referring to Figure 1, the volume V of the tip equals $(R^3 - r^3)2\pi/3 + (R^2 - r^2)\pi L$. So given a choice of A the number of particles $|\mathcal{P}|$ in the discretization will be roughly V/A^3, since each particle can be associated with an A-sided cubicle centered at the particle (the union of all such cubicles tesselate the tip's volume).

D. Spring interconnection

Each mass is connected to its 26 closest neighbors by springs. This arrangement generates a cubical lattice, characteristic of certain solid materials [21]. Macroscopically, this type of lattice exhibits elastic isotropic behavior, i.e., its deformation properties are independent of the orientation and placement of applied external forces [12]. Under the above interconnection scheme, for a given number of particles $|\mathcal{P}|$, the total number of springs $|\mathcal{S}| \simeq 13|\mathcal{P}|$, since there are 26-connections/mass and each spring is counted twice. The initial distance between particles a spring interconnects is used to define the spring's rest length l_0.

E. Selecting dynamic parameters

The deformable tip is made of soft rubber; this material's typical density ρ is 0.5 g/cm^3 [22]. To maintain homogeneity, all particles in the discretization are assigned equal mass $m = \rho V/|\mathcal{P}|$, such that their sum equals the fingertip's total mass ρV. Under small deformations, rubber can be treated as an isotropic elastic material in that the tensile and shear *strains*, written as ϵ and γ, are proportional to the applied tensile and shear *stresses*, written as σ and τ [12]:

$$\begin{aligned}\sigma &= E\epsilon \\ \tau &= G\gamma\end{aligned}$$

where E and G are the material's *Young's* and *shear moduli*, respectively. For soft rubber, these parameters take the typical values of $E = 40$ gf/cm^2, and $G = 14$ gf/cm^2 [22].

The following rationale is used in choosing the spring constants k for each spring in the model: consider several particles of mass m connected in series by one-dimensional springs along the x-axis, Figure 5. Let all such springs have the same length l_0 and the same spring constant k. In the limit, as $l_0 \to 0$, the discrete springs coalesce into a single elastic body, so that ml_0 becomes the material's mass density ρ, and the product kl_0 becomes its Young's modulus, E.

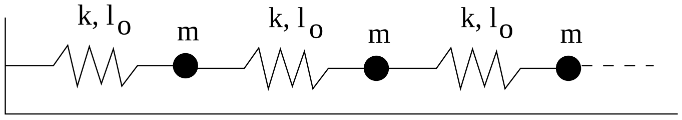

Fig. 5. Several particles of equal mass m are interconnected in series by one-dimensional springs of equal spring constant k and rest length l_0. In the limit, as $l_0 \to 0$, $ml_0 \to \rho$ and $kl_0 \to E$.

If tensile stress is applied normal to a finely sampled cubical lattice along the x direction, horizontal springs configured in a manner similar to Figure 5 will resist the deformation; here we ignore the stress resistance offered by non-horizontal springs, and simply choose $k = E/l_0$, where l_0 is the spring's rest length. To maintain equal elastic response in all directions (isotropy), all spring constants in the model are set to E/l_0. Though the G parameter was not used in calculating k, the system will naturally resist shear stresses due to the structural robustness of the lattice – we are currently investigating how to precisely emulate G and E simultaneously with a proper choice of spring constants.

The choice of μ, the damping constant, should be such that the system exhibits the shock absorption properties of rubber. We choose μ such that the system nearly *critically damped*, i.e., we select the lowest value of μ which prevents oscillatory motions of particles in response to external transient forces. Consider two particles of mass m connected by a spring of elastic constant k. Such a system is equivalent to a system composed of a spring of constant k with one endpoint attached to a wall and the other attached to a single particle of mass $m' = m/2$ [13]. In turn, a 1-mass oscillator is said to be *critically damped* when $\mu = 2\sqrt{m'k}$ (m' is known as the *equivalent mass*). This formula is used to critically damp the model's springs by assuming each spring only "sees" the masses it connects (in reality the mass equivalent will be higher, since other springs emanate from a given spring's endpoints; in turn, under the proposed method for choosing μ, the system to be slightly overdamped).

A last question relates to the choice of the integration step δt. Besides the integration error concern, the integration step must be such that it is much shorter than the inverse of the largest oscillatory frequency we wish to capture in the system. The oscillatory behavior of a particle system can be expressed by a weighted sum of fundamental modes; in each such mode all particles vibrate with the same frequency [23]. Given the critical damping imposed at each spring, higher fundamental frequencies will be sharply attenuated. A conservative estimate of the highest oscillation frequency is made as follows: let each spring and the masses it connects be treated as an isolated system. If the spring has an elastic constant $k = E/l_0$, the equivalent mass it "sees", $m' = m/2$, gives rise to a *natural oscillation frequency* $\omega_0 = \sqrt{k/m'}$ [13]. We simply take this value as the largest frequency component to be sampled and numerically integrated. In particular, we choose $\delta t << 1/\omega_0$.

F. *Collision modeling*

A rigid surface is modeled as an infinite plane Π perpendicular to the z axis. At every simulation step, particles lying on the finger's boundary which are found to have crossed Π are retracted and anchored (till the end of the simulation) to the nearest point on the plane. This is a simplified model of infinite-friction contact (no slipping). As the finger descends, more and more particles become permanently anchored to the surface, resulting in visible deformation of the tip's structure. Under this simplified collision/contact model, interactions between the finger and the surface can be resolved in time linear in the number of boundary particles in the tip discretization.

III. Virtual control

The motion of the finger toward the rigid surface is controlled by an adaptive PID controller [1], interfaced to the dynamic model as illustrated in Figure 3. Though the nonlinear deformation characteristics of the fingertip are not known, the controller's gains are adjusted experimentally so as to critically-damp the system (control prototyping).

The controller attempts to maintain a constant approach speed and rod orientation before, during, and into the contact/deformation phase. The controller can apply a force and a torque to the rigid rod's endpoint. The rod's motion is computed using rigid body dynamics [13]. A few internal particles in the tip are connected by springs to points on the rod's surface. The rod reacts to the resultant of forces applied by both the controller and the springs which connect it to the particle system.

Force and torque are calculated at the rod's endpoint by summing the spring forces acting on the particles lying on the rod's surface. This information is relayed back to the control system which in turn adjusts the force and torque it applies to the rod so a constant approach speed and orientation can be maintained.

IV. Simulation results

A simulation program was written which takes as input a high-level description of the geometric and dynamic parameters of the system, and generates a 3d graphical animation of the experiment. Four snapshots of a run of the simulator are shown in Figure 6 – in the sequence, only boundary masses/springs and the rod are portrayed. All processing takes place in real time, including dynamic updates, and collision detection/contact processing. The program also displays, at the rod's endpoint the force and torque which are calculated during run-time and fed back to the PID controller.

V. Conclusion

In this paper we presented a computational model for the dynamics of a deformable robot fingertip as it interacts with a rigid surface. The model's motion is commanded by an external controller over a virtual control interface; through this interface, the controller interacts with the simulated deformable model in the exact same way it would interact with the actual physical system. The proposed dynamic model is computationally efficient and results in physically reasonable motion and deformation. The model's discrete masses greatly simplify collision detection and force/torque computation, permitting real-time visualization and task control.

Future work will focus on simulation of soft contact grasps. The contact model will be enhanced to allow for slipping – this can be done by keeping track of the tangential force acting on each "anchored" particle – if this force exceeds a threshold, the particle can be essentially released from the surface. We also intend to develop a formal correspondence between discrete mass-spring-damper lattices and continuum models of elastic media, starting by using G, the shear modulus in the selection of spring constants within the lattice. Finally, we intend to compare simulation results with force and deformation data obtained experimentally.

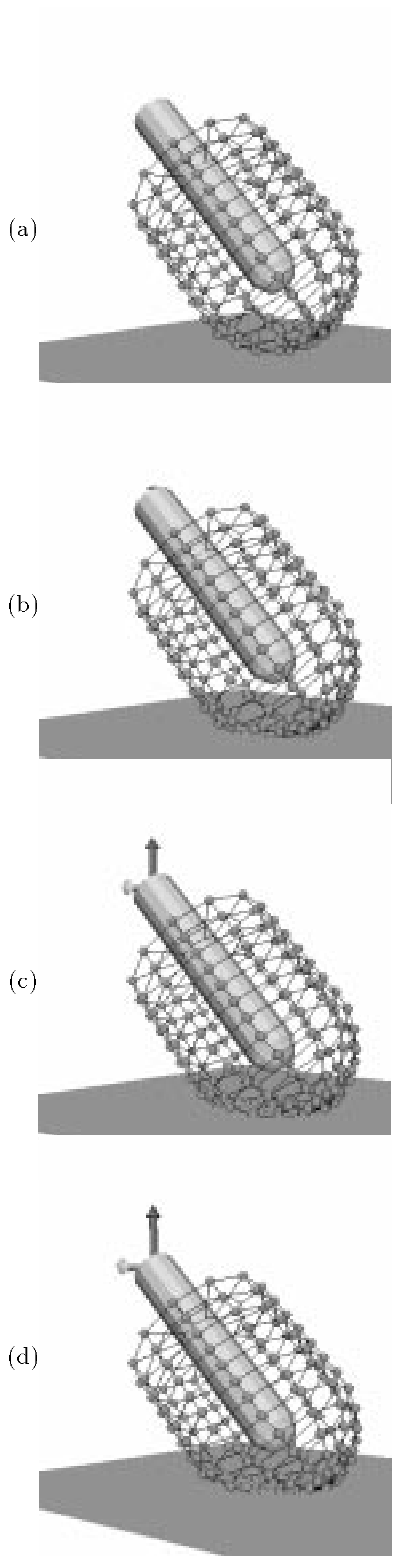

Fig. 6. 3d graphical output of the simulation program.

References

[1] J. Craig. *Introduction to robotics, mechanics and control.* Addison-Wesley, Reading, MA, 2nd edition, 1989.

[2] V.-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3), June 1988.

[3] C. Bard, J. Troccaz, and G. Vercelli. Shape analysis and hand preshaping for grasping. In *International Workshop On Intelligent Robots and Systems*, Osaka, Japan, November 1991.

[4] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. Zilles. Haptic rendering: programming touch interaction with virtual objects. In *Symposium on interactive 3D graphics*, Monterey, CA, 1995.

[5] J. Hollerbach and S. Jacobsen. Haptic interfaces for teleoperation and virtual environments. In *Workshop on Simulation and Interaction in Virtual Environments*, Iowa City, IA, 1995.

[6] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, August 1988.

[7] C. Chen, M. Trivedi, and C. Bidlack. Simulation and animation of sensor-driven robots. *IEEE Transactions on Robotics and Automation*, 10(5), October 1994.

[8] P. Lee, D. Ruspini, and O. Khatib. Dynamic simulation of interactive robotic environment. In *IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994.

[9] O. Zienkiewicz and R. Taylor. *The finite element method.* McGraw-Hill, London, UK, 4th edition, 1989.

[10] D. Terzopoulos, J. Patt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987.

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

[12] S. Timoshenko and J. Goodier. *Theory of elasticity.* McGraw-Hill, Tokyo, Japan, 1970.

[13] H. Goldstein. *Classical mechanics.* Addison-Wesley, Reading, MA, 2nd edition, 1980.

[14] J. Hodgins, L. Wooten, D. Brogan, and J. O'Brien. Animating human athletics. In *SIGGRAPH*, 1995.

[15] A. Joukhadar and C. Laugier. Fast dynamic simulation of rigid and deformable objects. In *International Workshop On Intelligent Robots and Systems*, Pittsburgh, PA, August 1995.

[16] M. Desbrun and M.-P. Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH*, Los Angeles, CA, 1995.

[17] J.-P. Gourret, N. Thalmann, and D. Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–30, July 1989.

[18] B. Mirtich. Impulse-based dynamic simulation. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, chapter 13. A. K. Peters, 1995.

[19] M. Green. Using dynamics in computer animation: control and solution issues. In N. Badler, B. Barsky, and D. Zeltzer, editors, *Making them move*, chapter 14, pages 281–314. Morgan Kaufmann Publishers, Inc., 1991.

[20] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical recipes in C.* Cambridge University Press, New York, NY, 2nd edition, 1992.

[21] T. S. Hutchinson and D. Baird. *The physics of engineering solids.* John Wiley & Sons, Inc., New York, NY, 1963.

[22] B. Irons and N. Shrive. *The finite element primer.* John Wiley & Sons, Inc., New York, NY, 1983.

[23] R. Steidel. *An introduction to mechanical vibrations.* John Wiley & Sons, Inc., New York, NY, 3rd edition, 1989.