Dynamic Free-Form Deformations for Animation Synthesis

Petros Faloutsos, Michiel van de Panne, *Member*, *IEEE*,
and Demetri Terzopoulos, *Member*, *IEEE*

Abstract—Free-form deformations (FFDs) are a popular tool for modeling and keyframe animation. This paper extends the use of FFDs to a dynamic setting. Our goal is to enable normally inanimate graphics objects, such as teapots and tables, to become animated, and learn to move about in a charming, cartoon-like manner. To achieve this goal, we implement a system that can transform a wide class of objects into dynamic characters. Our formulation is based on parameterized hierarchical FFDs augmented with Lagrangian dynamics, and provides an efficient way to animate and control the simulated characters. Objects are assigned mass distributions and elastic deformation properties, which allow them to translate, rotate, and deform according to internal and external forces. In addition, we implement an automated optimization process that searches for suitable control strategies. The primary contributions of the work are threefold. First, we formulate a dynamic generalization of conventional, geometric FFDs. The formulation employs deformation modes which are tailored by the user and are expressed in terms of FFDs. Second, the formulation accommodates a hierarchy of dynamic FFDs that can be used to model local as well as global deformations. Third, the deformation modes can be active, thereby producing locomotion.

Index Terms—Physically based animation, free-form deformations, control synthesis, deformation models, Lagrangian dynamics.

1 Introduction

Deformations play an important role in computer graphics, allowing mundane objects to assume interesting, new shapes. Commercial animated films often depict inanimate objects that come to life and need a way to move as living creatures do. The dinner scene in Disney's animated feature film *The Beauty and the Beast* is a prime example. The cutlery, pots, and other household objects dance, sing, and perform like human actors. Their motions are a result of smooth deformations of their original shapes. Computer graphics can provide helpful tools for modeling and animating such characters [1].

The free-form deformation (FFD) [2], essentially a geometric spline deformation that is manipulable through a lattice of control points, is a popular tool for modeling and animating nonrigid objects [3]. Typically, an object of interest is embedded within the spatial domain of the FFD and the object undergoes deformations as the FFD control points are manipulated. Such animations can be tedious to create, however, because, typically, a large number of control points must be specified manually at keyframes.

This paper generalizes conventional geometric FFDs within a dynamic setting. We introduce the *dynamic FFD* whose control points evolve automatically through time in accordance with mechanical principles. In its basic form, our formulation associates inertial and elastic properties to an embedded solid object to simulate its passive nonrigid motion. Our dynamic FFD representation employs deformation modes which restrict the deformations of an object to those expressible by a set of modal basis functions. The animator can tailor the modes, thereby exercising control over the range of shapes that occur during simulation. Deformation modes yield an efficient dynamics simulation, as each modal amplitude contributes a single degree of freedom. We formulate hierarchical dynamic FFDs, which support properly interacting local and global deformations. Finally, our dynamic FFDs can be active, enabling a normally rigid, static object to locomote under its own motivation.

Dynamic FFDs offer animators the opportunity to apply physical dynamics to the creation of expressive, cartoon-like animations. Fig. 12 shows an example of how to use dynamic FFDs to make static objects become animate. The apple in this figure is embedded in a global FFD lattice and is equipped with a local FFD lattice around the leaf. The teapot in Fig. 12 is also equipped with both global and local lattices. Fig. 15 shows snapshots from a dynamic animation of the teapot, depicting the motion resulting from a lifting force applied to the top of the lid handle.[1] The force vanishes after the 12th frame and the deformable teapot falls to the ground and bounces.

As a second example, Fig. 1 shows a cartoon car driving on a bumpy road. The car has a global deformation mode, as well as a local deformation mode for each wheel. The car rolls on the terrain as a result of a constant, horizontal, driving force.

• *The authors are with the Department of Computer Science, University of Toronto, 10 King's College Rd., Toronto, Ontario, Canada M5S 3G4. E-mail: {pfal, van, dt}@dgp.toronto.edu.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 105392.

1. Note that, in figures illustrating animations, the order of images is indicated by an arrowhead.

Fig. 1. A cartoon car simulation using dynamic FFDs.

A third example illustrates how a cartoon table can animate itself in a spirited fashion using active FFD deformation modes. Fig. 2 shows the table performing a periodic bounding motion which has been synthesized automatically using a set of user-supplied deformation modes. The deformation space was searched for deformations that maximize the distance traveled with each bound.

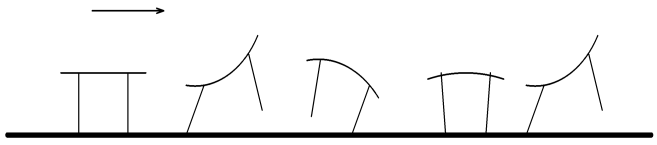


Fig. 2. A cartoon table performing a bounding motion.

The remainder of the paper is organized as follows. Section 2 presents previous work done in the area. Section 3 discusses the relation between our work and prior work. Section 4 describes our dynamics formulation. Section 5 presents the animation system we have developed based on dynamic free-from deformations and explores its capabilities. Section 6 investigates how active deformations can be used to achieve cartoon-like animations, such as walking tables or hopping teapots, through the automated search for optimal locomotion control strategies. Last, Section 7 presents conclusions and discusses future work.

2 Prior Work

The geometry and dynamics of deformable objects have been modeled using a variety of different methods. In general, there are several basic requirements. First, the model must associate a mass density with the geometry. Second, it must have a restricted number of degrees of freedom to permit efficient simulation. Third, the model must incorporate equations of motion that evolve the degrees of freedom in accordance with internal and external forces. In general, we can distinguish between two main approaches toward the modeling of deformable objects, *implicit*[2] and *parametric*. Parametric formulations define deformations explicitly on geometric models. Models are discretized with respect to their material coordinates by using finite differences or finite elements. The use of elastic models in the context of graphics was first presented by Terzopoulos et al. [4]. These physics-based deformable models have been generalized to include visco-elastic properties and inelastic behavior, such as fracture [5]. A deformable model is formulated by Metaxas and Terzopoulos [6], which includes global superquadric deformations and local spline deformations. The most straightforward deformable model consists of a network of springs and point masses. Haumann et al. [7] animate deformable objects using spring-mass models and wind fields.

An advantage of the second major approach, implicit deformations, is that they can deform simple or complex

2. Implicit deformations are often referred to as *global* deformations in the literature. We use the term *implicit* because we need to distinguish between local and global implicit deformations in our formulation.

objects with equal ease. Implicit deformations work by embedding an object in a space. When the space is warped, the embedded object deforms. The deformation is defined by a function that maps a point of the undeformed space onto the deformed space. Free-form deformations (FFDs) [2] are a primary example of implicit geometric deformations in which the space is defined by a multidimensional spline. Kinematically animating the control points which define the spline will animate the deformation imposed on the object. The animator is typically responsible for designing the necessary deformations. The technique described by Coquillart and Jancène [3] provides useful generalizations for creating keyframed FFD animations.

Several techniques using implicit geometric deformations have been proposed for modeling dynamic deformations. Chadwick et al. [8] use implicit deformations in the animation of articulated characters. FFD lattices are attached appropriately on the skeleton, whose dynamic motion deforms the lattices, thereby modeling soft tissue deformations. Pentland and Williams [9] employ the principal deformation modes of an elastic isoparametric hexahedral finite element to animate deforming embedded objects. Witkin and Welch [10] use polynomial global deformations in a physics-based model. Baraff and Witkin [11] use global implicit deformations in conjunction with physics-based simulation to animate polygonal and parametric objects. This formulation is similar to the one presented in [10], which uses deformations that are linear with respect to the state of the object, and is restricted to passive objects. Terzopoulos and Qin [12] model deformable surfaces using NURBS which are given dynamic properties. They also describe a physics-based deformation technique using dynamic NURBS FFDs, although the formulation and application differs from the one we propose.

3 Features of Our Approach

Existing dynamic models are capable of representing many types of animated objects. However, most are designed to model articulated figures [8], [13], [14], realistic creatures [15], or objects not capable of autonomous motion [11], [6], [9], [4], [12], [10]. They are not suitable for cartoon-style character animation, such as a teapot that comes to life. In this paper, we introduce an approach particularly well suited to characters of this type.

Of particular interest to us are techniques that integrate cleanly with standard tools. For this reason, we have sought to develop a dynamic formulation of deformable models based upon standard free-form deformations. Several features of our dynamic FFDs are unique with respect to previous work on implicit deformation methods. Our approach provides the user with an intuitive method to design deformation modes which can be imposed on different objects and used to produce a compact, limited set of allowable deformations. As a point of departure, this strategy is similar to Witkin and Welch's [10], and Baraff and Witkin's [11], who restrict the class of deformations to those expressible as a linear transformation of the state parameters. However, our formulation accommodates a hierarchy of deformations which are applied to objects in a nonlinear fashion with respect to the state parameters. Although some matrices are no longer constant, the different levels of deformations provide the animator with more flexibility and enable the creation of a variety of interesting effects, such as the car with a flexible body and flexible wheels, illustrated in Fig. 1.

With regard to the problem of controlling deformations to produce animation, we make the deformation modes active and implement a complete motion control and motion synthesis system for flexible characters. By contrast, the control method used in [10] is based on following motion paths. Independently of our work, but in the same spirit, Christensen et al. [16] describe a method for automatically synthesizing controllers for simple creatures using global implicit deformations and stochastic search. They model objects using spring-mass lattices that can undergo a set of canonical global implicit deformations; i.e., this work uses implicit deformations to describe the motion of spring-mass systems. Their deformation lattices follow closely the spring-mass approximations of the animated characters, thus deformations cannot be easily reused for different objects. By contrast, our formulation facilitates reusability. We furthermore introduce the use of stochastic methods to automatically synthesize motions for flexible characters based on FFDs.

4 Formulation of Dynamic FFDs

In the first step of our approach, the user designs the deformation modes to apply to an object. For example, Fig. 3 shows two deformation modes created for a 2D table and how they interact. When the dynamic motion of this table is simulated, the shape of the table is restricted to shapes expressible as a linear combination of the two modes, as is shown in the figure. Deformation modes provide a way of actuating characters which normally have no moving parts. Deformation modes

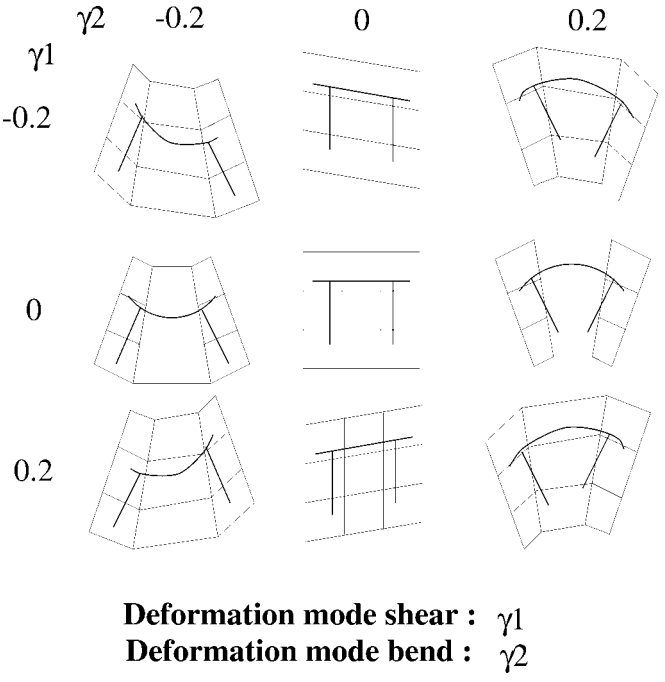


Fig. 3. FFD deformation modes designed for a table.

also provide a way for an animator to directly specify the range of allowable shapes of a character, which simplifies both the simulation and control of the character.

4.1 Deformation Modes

A deformation mode is defined using a single instance of an FFD deformation, specified by a user-supplied matrix $\mathbf{d} = \left[\mathbf{d}_x\ \mathbf{d}_y\ \mathbf{d}_z\right]$ which determines the maximum displacement of the lattice points along each dimension. As shown in Fig. 3, multiple deformations are combined in a linear fashion by adding the relative displacements of the control points. The amount of deformation for each mode is controlled by an amplitude parameter $\gamma_i \in [-1, 1]$ that operates as a scalar multiplier or weight for the deformation matrix **d**. Our implementation uses a $4 \times 4 \times 4$ control point lattice for the FFD and Bernstein polynomials as the basis functions. The 2D example in Fig. 3 shows two deformation modes, bend and shear, and their linear combinations.

4.2 Hierarchical Deformations

To implement hierarchical deformations, we also allow local FFD lattices to be imposed on parts of an object. The points affected by a local lattice are also affected by the global lattice, as shown in Fig. 12.

Currently, it is the user's responsibility to ensure that local deformations do not destroy the continuity of the object's surface. This can be done by using local deformations which do not affect specific points on the boundary of the local lattices. For example, in Fig. 12, this is done by ensuring that the deformation modes designed for the local lattice do not affect the bottom two rows of control points, which are affected only by the global deformation.

4.3 Geometric Representation

Our objects are able to deform and also undergo rigid body motion. Deformations are considered with respect to a noninertial coordinate system attached to the object as shown in Fig. 4. This coordinate system follows the rigid body motion of the object, translating and rotating with respect to the inertial, world coordinate system. Transformations are applied to the object in the following order:

1) local deformations,
2) global deformations,
3) rotation,
4) translation.

Assuming D_G global deformation modes, D_l deformation modes associated with the lth local lattice, and assuming L local lattices, the degrees of freedom for the object are given by the vector

$$\mathbf{q} = \left[\overbrace{\overbrace{\underbrace{t_x}_{\text{translation}}\underbrace{t_y t_z}_{\text{translation}}}^{\text{global motion}}\ \overbrace{\underbrace{\theta_x}_{\text{rotation}}\underbrace{\theta_y\theta_z}_{\text{rotation}}}^{\text{global motion}}}^{\text{global motion}}\ \overbrace{\underbrace{\gamma_{G1}\cdots\gamma_{GD_G}}_{\text{global lattice}}\ \underbrace{\gamma_{11}\cdots\gamma_{1D_1}}_{\text{local lattice 1}}\cdots\underbrace{\gamma_{L1}\cdots\gamma_{LD_L}}_{\text{local lattice } L}}^{\text{deformation amplitudes}}\right], \quad (1)$$

where the t_i are translation parameters, the θ_i are Euler angles,[3] and the γ_{ij} are deformation amplitudes.

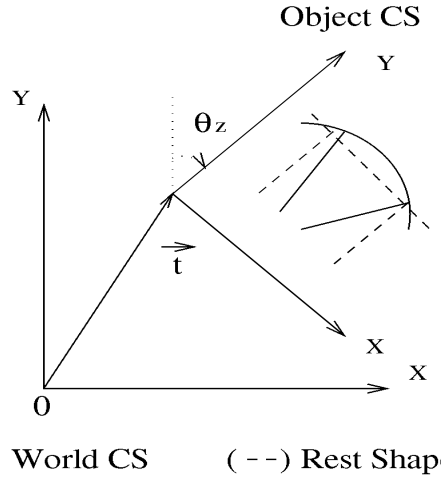


Fig. 4. The table moving in space using a bend deformation mode.

The components of **q** are a set of generalized coordinates (degrees of freedom) which determine the position of all points on a deformed object. The world coordinates of an object point **P**, in terms of its local lattice coordinates (s_l, t_l, u_l) and the generalized coordinates in **q** are given by

$$\mathbf{P} = \mathbf{t} + \mathbf{R}\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\left(\sum_{n=1}^{D_G}\gamma_n^G\begin{bmatrix} d^n_{Gijk,s_G} \\ d^n_{Gijk,t_G} \\ d^n_{Gijk,u_G}\end{bmatrix} + \begin{bmatrix} L^{s_G}_{Gijk} \\ L^{t_G}_{Gijk} \\ L^{u_G}_{Gijk}\end{bmatrix}\right) B(s_G, i)B(t_G, j)B(u_G, k), \quad (2)$$

where **t** is the vector of translation parameters, **R** is the compound rotation matrix, and, with index G indicating quantities associated with the global lattice, D is the number of animator-specified deformation modes for the lattice, γ_n is the amplitude of the nth deformation mode, d^n_{ijk} are constants which define the nth deformation mode of the i, j, k-control point of the lattice, L_{ijk} are undeformed lattice coordinates, and $B(s, i) = \binom{3}{i}(1-s)^{3-i}s^i$ is the Bernstein polynomial. For those points that are affected by local deformations, their parameter space coordinates (s_G, t_G, u_G) with respect to the global lattice are in turn given by

$$\begin{bmatrix} s_G \\ t_G \\ u_G\end{bmatrix} = a\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\left(\sum_{n=1}^{D_l}\gamma_n^l\begin{bmatrix} d^n_{lijk,s_l} \\ d^n_{lijk,t_l} \\ d^n_{lijk,u_l}\end{bmatrix} + \begin{bmatrix} L^{s_l}_{lijk} \\ L^{t_l}_{lijk} \\ L^{u_l}_{lijk}\end{bmatrix}\right) B(s_l, i)B(t_l, j)B(u_l, k), \quad (3)$$

where index l indicates quantities with respect to the lth local lattice, **a** is a constant matrix that transforms the local

3. Euler angles as a means to parameterize rotation have two well-known disadvantages: The order in which they are applied can change the resulting rotation and certain rotation sequences may lead to singularities such as the Gimbal lock [17]. For this reason interpolation between Euler angles is generally avoided in favor of interpolation of quaternions [18]. In our formulation, the order in which Euler angles are applied is dictated unambiguously by the equations of motion. During simulation, the Euler angles are updated as part of the numerical integration of the equations of motion, without interpolation, and the updates are much less than 90 degrees, thus precluding the possibility of Gimbal lock. Consequently, we may employ Euler angles without danger, and have chosen to use them for their simplicity.

lattice coordinates (s_l, t_l, u_l) to global lattice coordinates (s_G, t_G, u_G), and the remaining quantities have similar interpretations to their counterparts in (2).

Equations (2) and (3) are the starting point for the simulation of the dynamics of a deformable object. They prescribe that we apply local deformations using (3) before global deformations using (2).[4] The dynamics formulation presented next will ensure that a force which produces a local deformation appropriately affects the global deformations.

4.4 Equations of Motion

We use a Lagrangian formulation [19] to make our flexible models dynamic. A mass distribution is associated with the object by discretizing it in material coordinates using mass points. For example, a 2D-table could be assigned a discrete mass distribution, as shown in Fig. 5. In practice, approximating an object's distribution using 4 – 10 mass points is sufficient to yield convincing motions. The mass points are embedded in the same space as the object geometry and are thus also affected by local and global deformations. The kinetic energy of the M point masses is

$$E = \sum_{i=1}^{M} \frac{m_i}{2} \dot{\mathbf{x}}_i^T \dot{\mathbf{x}}_i, \tag{4}$$

where m_k is the kth mass and $\mathbf{x}_k$ its position with respect to the world coordinate system.

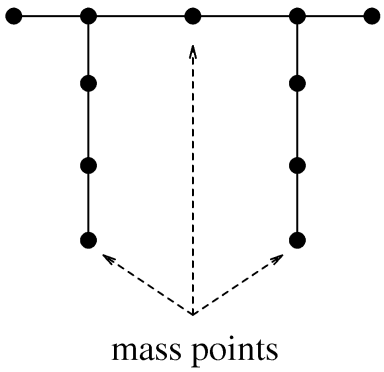


Fig. 5. Mass distribution for a 2D-table.

As an object deforms, it builds up internal strain energy that resists further deformation. These energies are defined by the user and are a function of the deformation amplitudes γ_i. Typically, we associate with deformations a potential energy of the form

$$V = \sum_{i=1}^{D} \frac{k_i}{2} \left(\gamma_i - \gamma_{i0}\right)^2, \tag{5}$$

where k_i is the elasticity constant associated with deformation mode i and γ_{i0} is the rest state of the mode. The rest states may be functions of time $\gamma_{i0}(t)$ and they may be varied actively to control the modes, in effect forming an object's "muscle" actuators.

The equations of motion can be derived by applying the Lagrangian formulation of the equations of motion with respect to the generalized coordinates q. The Lagrangian formulation is based on the following equation:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} - Q_k = 0, \tag{6}$$

where $\mathcal{L}$ is the Lagrangian, q_k is the kth generalized coordinate ($k = 1, \ldots, D$), Q_k is the total generalized force acting on the object along the q_k generalized coordinate, and the dot indicates a time derivative. We define the Lagrangian $\mathcal{L}$ to be the kinetic energy (4) minus the potential energy (5): $\mathcal{L} = E - V$. According to the derivation in Appendix A, the equations of motion assume the form

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{Q} + \frac{1}{2}\dot{\mathbf{q}}^T \frac{\partial \mathbf{M}}{\partial \mathbf{q}} \dot{\mathbf{q}} - \left(\frac{\partial \mathbf{M}}{\partial t}\right)\dot{\mathbf{q}}, \tag{7}$$

where $\mathbf{M}$ is the mass matrix, K is a diagonal stiffness matrix, and $\mathbf{Q}$ is the vector of generalized external forces. Equation (7) is a system of dim($\mathbf{q}$) coupled second-order ordinary differential equations in dim($\mathbf{q}$) unknowns.

The system is numerically integrated forward through time, as follows: At each time step of the simulation, the linear system (7), whose form is $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{b}$, is solved for $\ddot{\mathbf{q}}$ using the Choleski factorization of $\mathbf{M}$. Finally, we obtain the new values for the generalized velocities $\dot{\mathbf{q}}$ and the generalized positions $\mathbf{q}$ by integrating twice as follows:

$$\dot{\mathbf{q}}(t + \Delta t) = \dot{\mathbf{q}}(t) + \Delta t \ddot{\mathbf{q}}(t),$$

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) + \Delta t\, \dot{\mathbf{q}}(t) + \frac{\Delta t^2}{2} \ddot{\mathbf{q}}(t).$$

4.5 External Forces

Collision detection and resolution are not a research issue in our work. We have implemented simple existing methods for demonstration purposes. In particular, the system automatically samples the object at a set of collision points on the surface such that the relative distances between collision points are sufficiently small.[5] For example, we used 300 collision points for the teapot shown in Fig. 13. At each timestep, all the collision points are checked against the ground. For each collision point penetrating the ground, a collision force is activated as follows.

Ground contact is modeled using a penalty method which simulates the ground collision force using a spring and damper unit, as shown in Fig. 6a. When a point on the object penetrates the ground, a spring and damper unit is attached between the point of entry and the object point. The absolute value of the ratio between the friction force and the vertical force is not allowed to exceed $\tan(\theta)$, thus implementing a Coulomb friction model. The constants defining the stiffness of the unit are chosen such that the interpenetrations caused by collisions are small.

4. We have implemented only two levels of deformations for demonstration purposes. However, the formulation presented above can easily be generalized to an arbitrarily deeply nested hierarchy of local lattices by expressing (s_l, t_l, u_l) as a function of another level of lattices in a fashion similar to (3).

5. There is no need to associate mass with the collision points by virtue of the implicit deformations and the Lagrangian dynamics formulation.

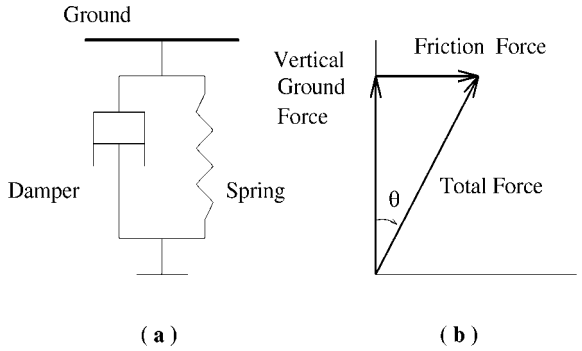


Fig. 6. Ground force model.

Ground contact and gravity are treated as independent external forces in the dynamics formulation. All external forces must be transformed from their Cartesian representation to the generalized one. After transformation, the external forces appear in the equations of motion as a set of generalized forces $\mathbf{Q}$. This is done by computing $\mathbf{Q}_F = \mathbf{J}^T\mathbf{F}$, where $\mathbf{J}$ is the Jacobian matrix derived in Appendix A.

5 A Dynamic FFD Animation System

We have implemented an interactive animation system using the dynamic FFD formulation. Fig. 7 illustrates the conceptual components of our system. In general, our system can automatically turn any geometric object into a dynamic deformable model by automatically computing mass distributions, attaching FFD-lattices, and assigning default values to all the dynamic parameters such that the object can support its own weight. Currently, only a global lattice is attached automatically, since the system cannot identify suitable parts on the object (local lattices are attached by the user). The system automatically computes a discrete representation of the object's mass distribution using a set of point masses, unless the user prefers to supply his or her own. Our algorithm ensures that all the FFD lattices contain a suitable number of mass points and that the actual geometric center of the object is near the center of mass of the computed mass distribution. The dynamic behavior of the object is determined by the deformation modes, each of which has its own stiffness and damping properties. The parameters defining the deformation modes, namely, the lattice displacements matrices, the stiffness parameters, and the damping parameters, can be loaded and set interactively. With the deformation modes loaded, the animator can produce either key-framed or physics-based motion, or alternate between them according to the characteristics desired of the motion. Fig. 8 illustrates an animation produced using four deformation modes and a mix of keyframing and dynamic simulation. The system allows the user to change the deformation modes at any point during a key-framed or physics-based animation in order to change the functionality of the object.

5.1 Key-Framing

Once an object and a set of deformation modes is loaded, the user can specify a deformation mode and the desired

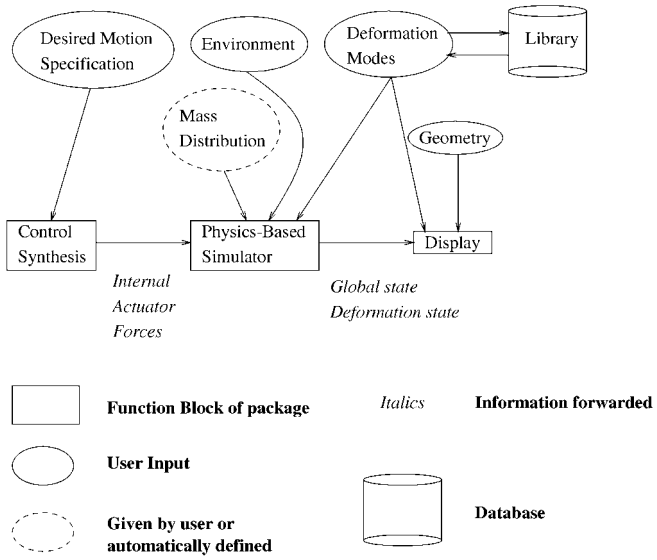


Fig. 7. Overview of the system.

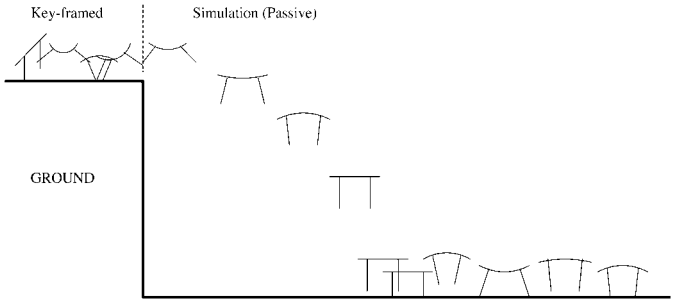


Fig. 8. Cartoon table jumping off a cliff.

amplitude to interactively deform the object. To prevent the object from losing contact with the ground when deforming during key-framing, the system allows a point of the object to be constrained to a fixed position. As in a typical key-framing system, the user can set key-frames and the system automatically interpolates between them to produce a complete sequence.[6] In Fig. 8, the first part of the motion is key-framed.

5.2 Physics-Based Simulation

The physics-based simulator of our system implements the numerical integrator for the Lagrangian equations of motion described in Section 4.4. The user can interactively modify a number of parameters, such as the ground stiffness, flags that enable or disable effects such as gravity, and parameters that control the stiffness and the damping of the deformation modes, in order to provide the object with the desired elasticity and the appropriate environment.

An example of the kind of animations that can be produced with our physics-based simulator is presented in Fig. 15. In this case, the teapot is equipped with both local and global deformations. The lattices are attached to the teapot as shown in Fig. 12. A spring is attached to the lid and it lifts

6. We interpolate rotations using quaternions, since the difference in rotations between keyframes can be large.

the teapot. The spring vanishes after the 12th frame and the teapot rebounds elastically, vibrating as it drops to the ground, and bouncing several times before coming to rest. Both the global and the local lattice are associated with the same stretch deformation. However, the global deformation is stiffer than the local one.

The simulation runs at interactive speeds for a variety of 2D and 3D objects. Table 1 shows the CPU time that each simulation iteration takes for different numbers of deformation modes. All experiments were performed on a Silicon Graphics R4000 Indigo^2 running at 100 MHz and involved graphics objects such as the teapot of Fig. 9, apples, and wine glasses, each discretized with six mass points and equipped with 300–500 collision points. The motions produced were pleasing and convincing. The complexity of the simulation is linear with respect to the number of point masses and the number of lattice points. The total number of deformation modes D determines the dimension of the linear system that is solved in each simulation iteration, thus the complexity of the simulation process is $O(D^3)$. To further improve the efficiency of the simulation, we include a mechanism that deactivates (eliminates) deformation degrees of freedom if the magnitudes of the associated amplitude and velocity are below a threshold. This makes practical the use of many local lattices, which are only incorporated into the equations of motion in an on-demand basis. The deactivated degrees of freedom are reactivated as soon as external forces appear. We have performed two experiments to test the effectiveness of this feature. In the first experiment, the cartoon car shown in Fig. 1 rolls down a bumpy steep hill. It was equipped with one global deformation and one local deformation on each of the wheels. The simulation consumed 133 seconds of CPU time with the aforementioned efficiency feature disabled, while it consumed 121 seconds with the feature enabled. In the second experiment, the teapot shown in Fig. 12, which was equipped with one global deformation and six local deformations (three on the handle and three on the spout), performed a free-fall. The simulation time without the efficiency mechanism was 12 seconds; it was seven seconds with the mechanism.

Stability is an important issue for numerical simulation. The main factors affecting the stability of our system are the time step, the stiffness introduced by the collision penalty method, the mass distribution, the chosen set of deformations, and elasticities k_i. The mass distribution of an object must have a natural connection with the associated deformation modes. For example, a squash deformation defined along the x-axis for a one-dimensional rod aligned with the y-axis has no meaning, and it yields an unstable system. The generalized coordinates of an object should be independent from each other as required by the Lagrangian dynamics formulation. For instance, the system will be ill-conditioned if the same deformation mode is defined twice. The more similar two deformation modes are, the less numerically stable the system will be.

TABLE 1
RUN TIMES

# GLOBAL DEF. MODES	# LOCAL DEF. MODES	TIMES (SEC)
1	0	0.154324192
2	0	0.157693698
4	0	0.167255900
8	0	0.199321312
10	0	0.228473064
1	1	0.160967480
1	2	0.166869388
1	4	0.179693202
1	8	0.209430214
8	8	0.400348746

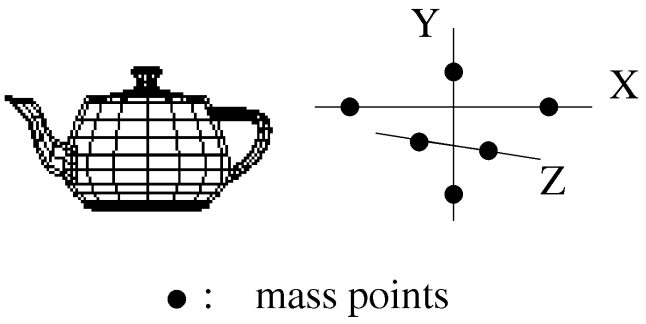


Fig. 9. A 3D teapot and an associated mass distribution.

6 MOTION SYNTHESIS FOR ACTIVE DEFORMATIONS

The dynamic models described in a previous section incorporate physics into a geometric model, and are useful for both passive and active characters. It is desirable to simulate active characters realistically and incorporate into the model the ability to produce autonomous motion. For realistic locomotion, the movement should arise from controlled actuation. Many different methods have been developed to deal with the control of dynamic graphics objects [13], [15], [20], [21], [22], [23]. A simple yet effective technique based on *cyclic pose control graphs*, which is suitable for articulated figures, is presented by van de Panne et al. [24]. The technique has been extended to acyclic graphs in order to achieve nonperiodic motions [14].

The control problem is often formulated as an *optimization problem*. The object is required to perform a task while minimizing a cost or maximizing an objective function. It has been shown that optimized control can be used to automatically synthesize controllers capable of making active simulated creatures locomote [13], [16], [23], [25], [26]. Motion and controller synthesis are often addressed using probabilistic optimization methods because they are easy to implement, are suitable for searching large spaces, and can avoid local suboptima. Controllers are repeatedly generated and subsequently evaluated using forward simulation. The motion synthesis problem is thus tackled by searching the space of possible controllers for those that produce suitable motions. Common search methods for the stochastic motion synthesis problem are *genetic algorithms* [25], [26] and *simulated annealing* [13], [23]. To our knowledge, previous work on controlling active deformable models has always used spring-mass models [22], [15], [23], whereas we use dynamic FFDs.

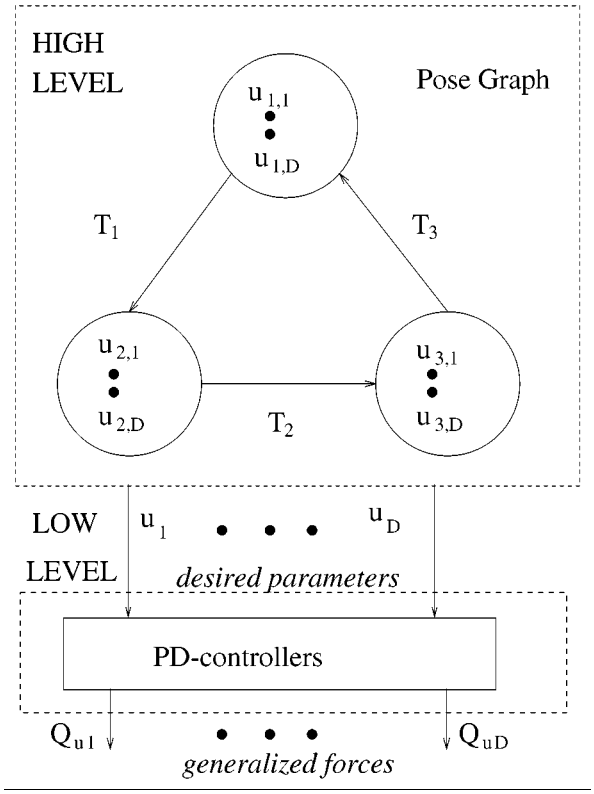


Fig. 10. The control structure.

6.1 Dynamic Control Model

The control of our dynamically deformable characters is based on open-loop[7] pose controllers which determine the high level control. The pose controller structure operates as a finite state machine. Each state defines a pose and specifies the desired shape of a character. "Shape" here refers to a character's internal degrees of freedom. Thus, a pose $\hat{\mathbf{q}}$ is defined in terms of the amplitudes of the deformation modes, $\hat{\mathbf{q}} = (\gamma_1, \ldots, \gamma_D)$, where D is the total number of global and local deformation modes. In the following, $\mathbf{u}$ indicates the desired poses. The pose controller structure is shown in Fig. 10. Each state (pose) has an associated *transition* time that specifies the time period for which the associated pose remains active. In the current version of our system, we have implemented only *cyclic* pose graphs, which result in periodic motions. The controller cycles continuously through the pose graph. The active pose drives low-level PD-controllers which make the object deform into the desired shape. Note that the desired shape is not necessarily achieved because of the presence of external forces. The deformation of the object into the desired pose can be done either by using potential energy terms in the Lagrangian, or by adding deformation forces to the vector of generalized forces. We have chosen to use the latter method. Hence, we associate PD-controller with each deformation mode and it produces a generalized control force along the associated generalized coordinate. The generalized control forces are defined as

$$\mathbf{Q}_u = -\left(\mathbf{C}(\hat{\mathbf{q}} - \mathbf{u}) + \mathbf{D}\dot{\hat{\mathbf{q}}}\right),$$

where $\mathbf{C}$ is a diagonal stiffness matrix, $\mathbf{D}$ is a diagonal damping matrix, and $\mathbf{u}$ are the desired pose parameters.

An animator can interactively specify a deformation mode to be active or passive. Active deformation modes are subject to control forces and they contribute to actuated motions, while passive modes appear only as a result of external forces. In general, the active deformation modes should be stiffer and more damped than the passive ones. Active deformations should not produce oscillations because most familiar muscle-based motions are nonoscillatory. Passive deformations should typically produce oscillations because they represent the effect of elastic strain energy being dissipated over time. However, the animator has complete control over the deformation parameters and can change them interactively at any point according to the desired result.

6.2 Motion Synthesis

Given an object and a set of deformation modes, it is convenient, and, for complex characters, necessary, to have a procedure that automatically produces controllers capable of making the object perform interesting modes of locomotion. Previous work in this area addresses the case of articulated figures with some notable exceptions [16], [23]. In our system, we use *simulated annealing* [27] because it can avoid local optima and it has proven to be suitable for our purposes. The process begins with an initial pose controller supplied by the user. Using forward simulation, the controller is evaluated according to a cost function. A new controller is then produced by stochastically modifying the initial one. The new controller is evaluated in the same way and the two controllers are compared. The better controller is selected and the procedure is repeated until a stopping criterion is met. Sometimes the process selects a new controller which performs worse than the previous one as a means to avoid local optima. The simulated annealing algorithm that we use is summarized in Fig. 11. The value of r controls the number of iterations that the algorithm performs. A fixed number of iterations can also be used. All our experiments converged to a solution in less that 100 annealing iterations.

1. Specify an initial configuration P
2. Specify an initial temperature $T > 0$
3. Pick a cooling rate $0 < r < 1$
4. Do forward simulation for time t_f and calculate the cost Cost (P)
5. While $T > T_{frozen}$
   (a) Pick a random neighbor P' of P
   (b) Do forward simulation for time t_f and calculate the cost $Cost(P')$
   (c) Let $\Delta = Cost(P') - Cost(P)$
   (d) If $\Delta \leq 0$ set $P = P'$
   (e) If $\Delta > 0$ set with $P = P'$ probability $e^{-\Delta/T}$
   (f) Set $T = r \cdot T$
6. Return P

Fig. 11. Using simulated annealing to find a suitable active controller.

7. The controller has no external feedback, i.e., it does not have an indication of how well it operates.

In the controller synthesis procedure, the poses are snapshots of the character's deformations during a motion that the animator would like the character to perform. The choice of an initial pose controller is important because it can significantly affect the final motion. The most important choice in defining an optimization process is the choice of the cost function, which is minimized by the simulated annealing process. The function quantifies the desirability of motions. There are many parameters which serve to define such a function, such as the speed, acceleration, work, and orientation of the object during the motion, and the desired trajectory that the object must follow. Realistic motions should also consume a reasonable amount of control energy. The difficulty of designing the cost function depends on the complexity of the character and the desired motion. Typical cost functions do exist for simple motions such as walking, hopping, and shuffling. Our system provides the user with a number of typical functions, such as those given in Table 2. For more complex motions, if none of the functions provided by the system yield good results, the user may need to resort to his or her own intuition.

We formulate our experiments as minimization problems. In order to prevent the synthesis of unrealistic high-energy motions, the cost function that we used in all cases grows linearly with the control energy consumed. Specific functions are presented in the next section as we discuss the results of the motion synthesis experiments.

6.3 Results

One of our experiments involved the synthesis of a controller that would make a 2D table perform a bounding motion. The table is equipped with a set of four deformations modes: vertical shear, horizontal bend, vertical squash, and horizontal squash. The first two are active deformation modes, and the remaining two allow for additional passive motion. Our most successful experiment performed optimization on all the parameters defining the pose graph. After 60 simulation trials, the table was able to perform a stable periodic mode of locomotion, shown in Fig. 2. The cost function that we used rewards the distance traveled and penalizes the energy expended. Table 2 presents the cost functions and the associated result; $E_c = \sum_{i=1}^{D} \left| \hat{q}[i] * Q_u[i] \right|$, where Q_u are the control forces, is a simplified expression of the energy consumed, t_x is the distance traveled along the x-direction, and $\dot{t}_x$ is the velocity along the x-direction. If $t_x \dot{t}_x < 0$, the cost is set to a high value.

The following experiment shows how different optimization functions can produce different motions. We construct a cost function consisting of two terms

$$f_{\cos t} = K_1 E_c - K_2 * \left| t_x \right|, \tag{8}$$

where E_c, t_x are as defined above and K_1, K_2 are constants. The choice of these constants guides the optimization towards different goals. We use this optimization function to produce different hopping motions for the teapot, as shown in Fig. 16. The motion produced by the initial, handcrafted controller is shown in the left panel of Fig. 16. For $K_1 = 0$ and $K_2 = 1$, the optimization process produces a controller resulting in larger hops and therefore consuming more control energy. The motion is shown in the right panel of Fig. 16. An intermediate case attempts to strike a compromise between the control energy and the distance traveled using $K_1 = 1$ and $K_2 = 1$. The resulting motion is shown in the center panel of Fig. 16. The case where $K_1 = 1$ and $K_2 = 0$ produces a trivial solution, namely a controller that keeps the teapot in place.

TABLE 2
COST FUNCTIONS

COST FUNCTION	COMMENTS
$1/t_x$	Hopping motion with unrealistic jumps
E_c/t_x	Hopping motion, normal jumps
$E_c/\left(1.0 \times 10^7 \sqrt{t_x \dot{t}_x}\right)$	Bounding motion
$E_c/\left(5.0 \times 10^7 \sqrt{t_x \dot{t}_x}\right)$	Faster bounding motion (Fig. 2)
$E_c/\left(5.0 \times 10^7 \sqrt{t_x}\right)$	Shuffling-like motion (Fig. 13)

The design of an appropriate optimization function to produce a desired motion is potentially a difficult task, and is especially difficult for unstable motions. We had to perform several trial-and-error experiments in order to choose the appropriate constants in the optimization function to obtain variations of an initial hopping motion. The shuffling motion is more stable and was therefore much easier to produce.

The optimization can be performed with respect to different subsets of the control parameters. Some experiments with the cartoon table optimize the transition times between different poses, some optimize the poses, and some optimize both poses and transition times. The more parameters that are used in the optimization process, the larger the space of admissible control functions, and, consequently, the more likely it is to arrive at an efficient solution. Naturally, the larger the control space, the slower the optimization process is at finding an optimal solution.

A second example is a 3D teapot which learns to locomote using two active deformation modes. Fig. 13 shows the teapot shuffling. The same motion has been reused and applied to the wine glass that appears in Fig. 14. In this example, the teapot is performing an automatically synthesized hopping motion using two deformation modes, the glass is making use of the shuffling motion produced for the teapot, and the apple is key-framed to fly above them using a local shear deformation on the leaf.

7 CONCLUSIONS

We have proposed and implemented a framework for the animation of deformable characters. Our approach generalizes standard free-form deformations to include physical dynamics. The formulation accommodates both local and global dynamic FFDs and ensures their proper

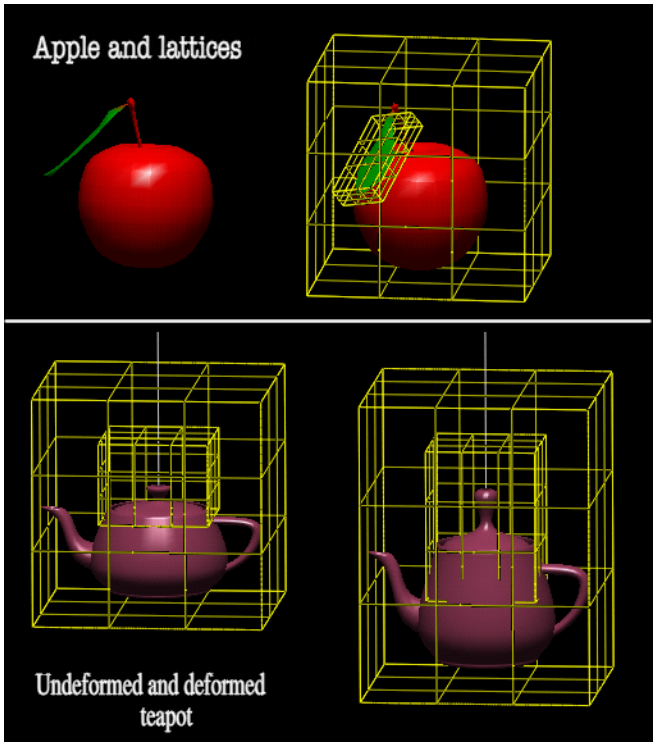


Fig. 12. Local and global deformations using dynamic FFDs.

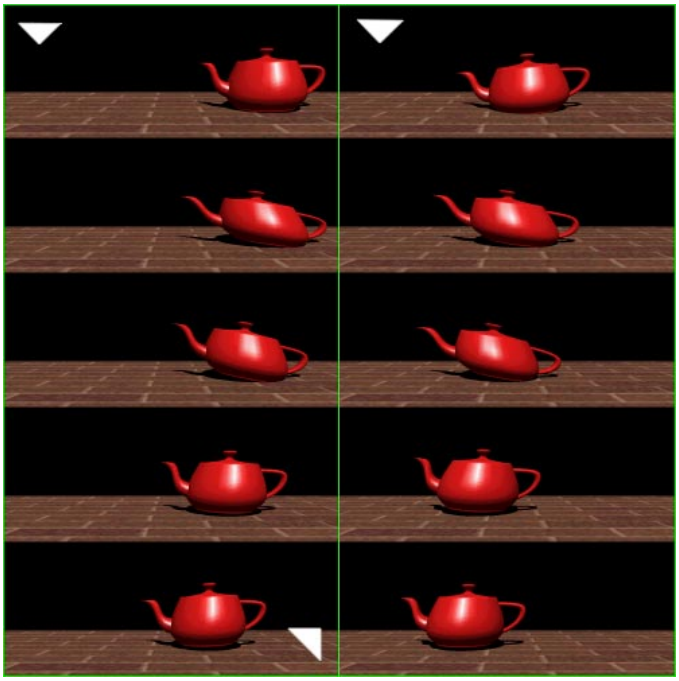


Fig. 13. 3D teapot performing a bounding motion.

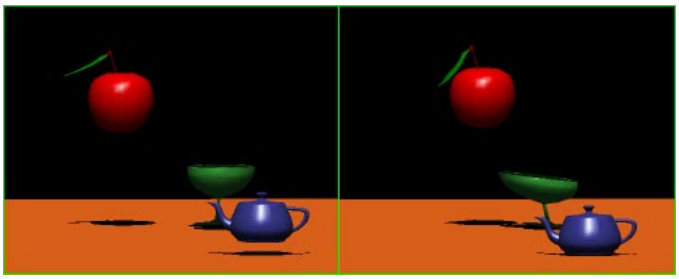


Fig. 14. A cartoon race.

interaction. User-tailored deformations provide predictability of results and yield efficient dynamic simulations. The dynamic deformations can be made active; thus, the technique is particularly amenable to transforming inanimate objects, such as teapots, into animate characters. Existing motion-synthesis techniques can be applied to the active deformations to automate the process of answering questions such as: How should a squash-and-stretch teapot move?

Free-form deformations are a standard tool that animators use to make inanimate objects come to life in a key-framed animation. The approach proposed here can take the FFD lattices defined for keyframing and make them active and dynamic. Because squash-and-stretch deformations are typical of cartoon animation, libraries of deformation modes and standardized motions could be provided to animators. Different characters can reuse stored libraries of deformation modes, deformations can be loaded or changed, and key-framed and physics-based motion can be exchanged as needed.

Our work can be extended in various ways. With respect to modeling, we could combine deformation modes nonlinearly. Dynamic constraints could be implemented in order to allow the construction of compound objects, such as articulated figures with deformable parts. We would also like to experiment with different FFD lattices. It would be interesting to use a formulation based on higher order Bernstein polynomials that will provide more flexibility with respect to the design of deformation modes. Similarly, we would like to try FFD formulations with B-spline basis functions. B-spline functions will allow the user to define lattices with different numbers of control points, and will provide local control. We would also like to experiment with hierarchies of lattices that share control points. Finally, to improve the quality of the dynamics, a more refined and efficient method for detecting and resolving collisions should be implemented.

Appendix A
Derivation of the Equations of Motion

This appendix derives the terms of (6) and shows how to calculate the equations of motion (7). As discussed in Section 4.4, the Lagrangian is $\mathcal{L} = E - V$ where E is the kinetic energy of the object defined by (4), and V is its deformation potential energy defined by (5). The object is discretized in material coordinates using point masses. The kinetic energy of the k-th point mass is $E_k = \frac{m_k}{2}\dot{\mathbf{x}}_k^T\dot{\mathbf{x}}_k$, where m_k is the mass and $\mathbf{x}_k$ is the position of the point in world coordinates. Using (2) and (3), we write E_k with respect to the generalized coordinates. To that end, we first note that $\dot{x}_i = \sum_j \frac{\partial x_i}{\partial q_j}\dot{q}_j$. Defining the Jacobian matrix $\mathbf{J}$ as $J_{ij} = \partial x_i / \partial q_j$, we can write $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$. Assume an object with D_G global deformation modes and L local lattices,

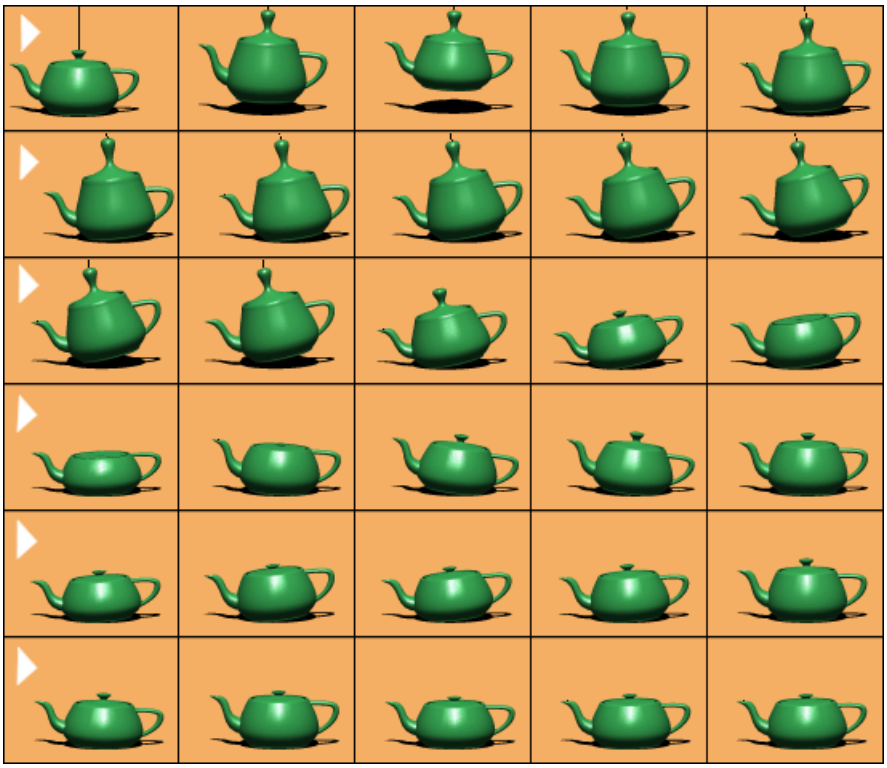


Fig. 15. A deformable teapot is lifted from the lid.

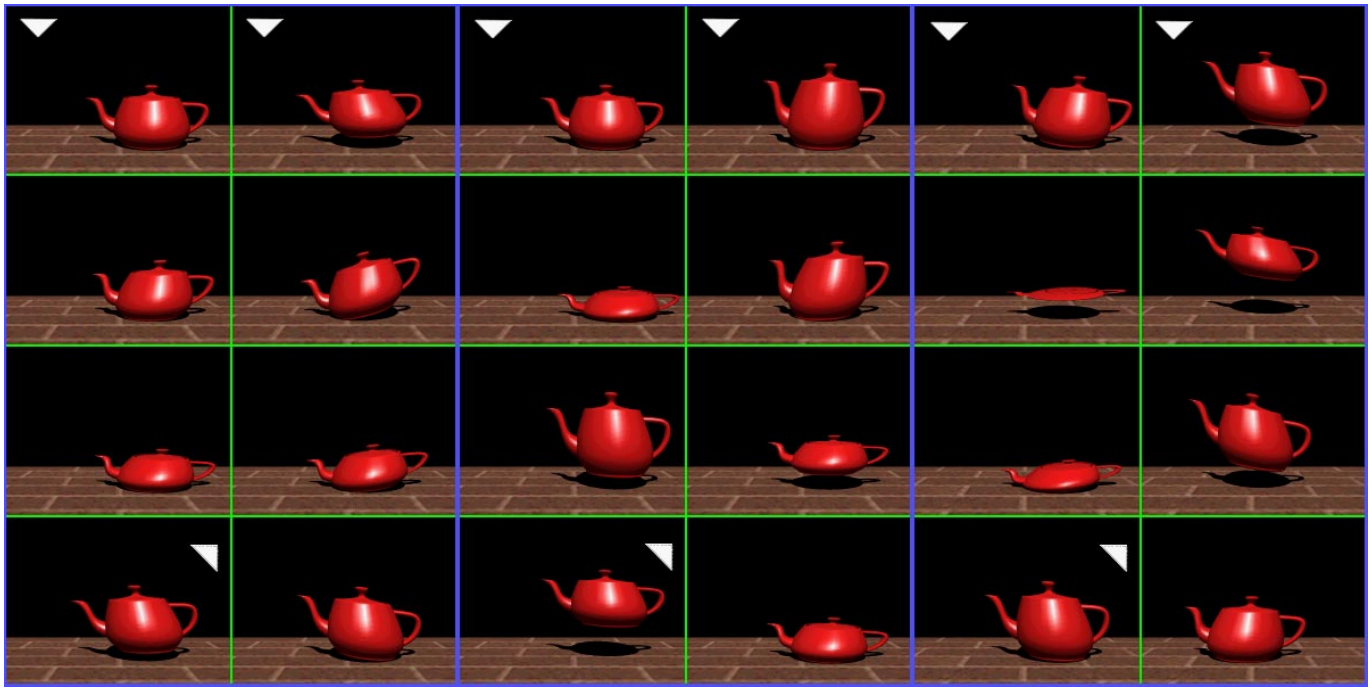


Fig. 16. Teapot performing various hopping motions.

each equipped with D_l deformation modes. The vector of generalized coordinates as defined in (1) is

$$\mathbf{q} = \left[\mathbf{t}^T \ \theta_x \ \theta_y \ \theta_z \ \gamma_{G1} \cdots \gamma_{GD} \ \gamma_{11} \cdots \gamma_{1D_1} \cdots \gamma_{L1} \cdots \gamma_{LD_L}\right]. \quad (9)$$

Denoting global quantities with index G and local quantities with index l, the expression for the Jacobian matrix evaluated at point $\mathbf{P}$ is

$$\mathbf{J} = \left[\mathbf{I}_3 \ \mathbf{A} \ \mathbf{B} \ \mathbf{C}_1 \cdots \mathbf{C}_L\right], \quad (10)$$

where

$$\mathbf{A} = \left[\frac{\partial \mathbf{R}}{\partial \theta_x}\mathbf{P} \ \frac{\partial \mathbf{R}}{\partial \theta_y}\mathbf{P} \ \frac{\partial \mathbf{R}}{\partial \theta_z}\mathbf{P}\right],$$

$$\mathbf{B} = \left[b_n\right] = \left[\mathbf{R}\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\mathbf{d}_{G,ij}^{n}B(s_G,i)B(t_G,j)B(u_G,k)\right], n = 1\ldots D_G,$$

$$\mathbf{C}_l = \left[c_{lm}\right] = \left[\mathbf{R}\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\left(\sum_{n=1}^{D_G}\mathbf{d}_{G,ij}^{n} + \mathbf{L}_{G,ijk}\right)\frac{\partial}{\partial\gamma_{lm}}\left(B(s_G,i)B(t_G,j)B(u_G,k)\right)\right],$$

$m = 1\ldots D_l$.

Recall that the global lattice coordinates (s_g, t_G, u_G) are a function of the local ones as shown in (3). The kinetic energy of mass point k with respect to the generalized coordinates is

$$E_k = \frac{m_k}{2}\dot{\mathbf{x}}_k^T\dot{\mathbf{x}}_k = \frac{m_k}{2}\left(\mathbf{J}_k\dot{\mathbf{q}}\right)^T\mathbf{J}_k\dot{\mathbf{q}} = \frac{m_k}{2}\dot{\mathbf{q}}^T\mathbf{J}_k^T\mathbf{J}_k\dot{\mathbf{q}}.$$

The total kinetic energy of the object is

$$E = \sum_k E_k = \sum_k \frac{m_k}{2}\dot{\mathbf{q}}^T\mathbf{J}_k^T\mathbf{J}_k\dot{\mathbf{q}} = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}},$$

where $\mathbf{M} = \sum_k m_k\mathbf{J}_k^T\mathbf{J}_k$ is a symmetric generalized mass matrix. We can calculate the required derivatives of the Lagrangian in (6) as follows

$$\frac{\partial\mathcal{L}}{\partial\dot{q}_k} = \frac{1}{2}\left(\sum_j\delta_{ki}M_{ij}\dot{q}_j + \sum_i\dot{q}_iM_{ij}\delta_{kj}\right) = \frac{1}{2}\left(\sum_j M_{kj}\dot{q}_j + \sum_i\dot{q}_iM_{ik}\right)$$
$$= \frac{1}{2}\left(\sum_j M_{kj}\dot{q}_j + \sum_j M_{kj}\dot{q}_jM_{jk}\right),$$

where δ_{ki} is the Kronecker δ-function. Since $\mathbf{M}$ is symmetric, $M_{kj} = M_{jk}$, thus $M_{kj}\dot{q}_j = \dot{q}_jM_{jk}$, and

$$\frac{\partial\mathcal{L}}{\partial\dot{q}_k} = \sum_j M_{kj}\dot{q}_j \Rightarrow \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{q}_k} = \sum_j M_{kj}\ddot{q}_j + \sum_j\frac{dM_{kj}}{dt}\dot{q}_j$$
$$= \sum_j M_{kj}\ddot{q}_j + \sum_j\left(\sum_l\frac{\partial M_{kj}}{\partial q_l}\dot{q}_l\right)\dot{q}_j. \quad (11)$$

The second term that involves the Lagrangian in (6) is calculated as follows:

$$\frac{\partial\mathcal{L}}{\partial q_k} = \frac{1}{2}\dot{q}_k\frac{\partial\mathbf{M}}{\partial q_k}\dot{q}_k - \mathbf{K}\mathbf{q}, \quad (12)$$

where $K_{ij} = k_i$ for $i = j$ and $K_{ij} = 0$ for $i \neq j$.

The derivatives of $\mathbf{M}$ required in (11), (12) can be calculated as follows:

$$\frac{\partial\mathbf{M}}{\partial q_j} = \frac{\partial}{\partial q_j}\sum_k m_k\mathbf{J}_k^T\mathbf{J}_k = \sum_k m_k\left[\left(\mathbf{J}_k^T\frac{\partial\mathbf{J}_k}{\partial q_j}\right)^T + \mathbf{J}_k^T\frac{\partial\mathbf{J}_k}{\partial q_j}\right]. \quad (13)$$

Determining the partial derivatives of $\mathbf{J}$ with respect to the generalized coordinates $\mathbf{q}$, as required by (13), is a rather lengthy, but mechanical process. We present them below for completeness. To follow the notation, the reader is referred to the definition of the vector of generalized coordinates given by (1). In addition, we denote $\mathbf{O}_n$ an $n \times n$ zero matrix, $\mathbf{0}$ is a zero column vector, $\mathbf{J}_I$ the Jacobian matrix defined in (10) evaluated with $\mathbf{R}$ being the identity matrix, $\mathbf{J}_I[q_k]$ the column of $\mathbf{J}_I$ that corresponds to the qth component of vector $\mathbf{q}$, and we define $BBB \equiv B(s_G,i')B(t_G,j')B(u_G,k')$. The primes are used to distinguish between the i, j's that appear in (14)-(17). Overbraces are used to compress a number of similar columns into one. The partial derivatives of the Jacobian are

$$\frac{\partial\mathbf{J}}{\partial t_i} = \mathbf{O}_{\dim(\mathbf{J})}, i = x, y, z \quad (14)$$

$$\frac{\partial\mathbf{J}}{\partial\theta_j} = \left[\mathbf{O}_3\ \overbrace{\frac{\partial^2\mathbf{R}}{\partial\theta_i\partial\theta_j}\mathbf{P}}^{i=x,y,z}\ \overbrace{\frac{\partial\mathbf{R}}{\partial\theta_j}\mathbf{J}_I\left[\gamma_{Gi}\right]}^{i=1\ldots D_G}\ \overbrace{\frac{\partial\mathbf{R}}{\partial\theta_j}\mathbf{J}_I\left[\gamma_{1i}\right]}^{i=1\ldots D_1}\cdots\ \overbrace{\frac{\partial\mathbf{R}}{\partial\theta_j}\mathbf{J}_I\left[\gamma_{Li}\right]}^{i=1\ldots D_L}\right], \quad (15)$$

$$\frac{\partial\mathbf{J}}{\partial\gamma_{Gj}} = \left[\mathbf{O}_3\ \overbrace{\frac{\partial\mathbf{R}}{\partial\theta_i}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3}d_{\gamma_{Gj}}BBB}^{i=x,y,z}\ \overbrace{\mathbf{0}}^{i=1\ldots D_G}\ \overbrace{\mathbf{R}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3}d_{\gamma_{Gj}}\frac{\partial BBB}{\partial\gamma_{1i}}}^{i=1\ldots D_1}\right.$$
$$\left.\cdots\ \overbrace{\mathbf{R}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3}d_{\gamma_{Gj}}\frac{\partial BBB}{\partial\gamma_{Li}}}^{i=1\ldots D_L}\right], \quad (16)$$

$$\frac{\partial\mathbf{J}}{\partial\gamma_{lj}} = \left[\mathbf{O}_3\ \overbrace{\frac{\partial\mathbf{R}}{\partial\theta_i}\mathbf{J}\left[\gamma_{lj}\right]}^{i=x,y,z}\ \overbrace{\mathbf{R}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3}d_{\gamma_{Gi}}\frac{\partial BBB}{\partial\gamma_{lj}}}^{i=1\ldots D_G}\ \overbrace{\mathbf{0}}^{i=1\ldots D_1}\ \cdots\ \overbrace{\mathbf{0}}^{i=1\ldots D_{l-1}}\right.$$
$$\left.\overbrace{\mathbf{R}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3}\left(\sum_{n=1}^{D_G}d^n + L_{ijk}^G\right)\frac{\partial^2 BBB}{\partial\gamma_{lj}\partial\gamma_{li}}}^{i=1\ldots D_l}\ \overbrace{\mathbf{0}}^{i=1\ldots D_{l+1}}\ \cdots\ \overbrace{\mathbf{0}}^{i=1\ldots D_L}\right]. \quad (17)$$

Finally, we calculate the derivatives of BBB that appear in the above equations. To distinguish between the three polynomials, we denote $B_{i'}B_{j'}B_{k'} \equiv BBB$. Applying the chain rule,

$$\frac{\partial BBB}{\partial\gamma_{li}} = \frac{\partial B_{i'}}{\partial\gamma_{li}}B_{j'}B_{k'} + B_{i'}\frac{\partial B_{j'}}{\partial\gamma_{li}}B_{k'} + B_{i'}B_{j'}\frac{\partial B_{k'}}{\partial\gamma_{li}}, \quad (18)$$

$$\frac{\partial^2 BBB}{\partial\gamma_{li}\partial\gamma_{lj}} = \frac{\partial^2 B_{i'}}{\partial\gamma_{li}\partial\gamma_{lj}}B_{j'}B_{k'} + \frac{\partial B_{i'}}{\partial\gamma_{li}}\frac{\partial B_{j'}}{\partial\gamma_{lj}}B_{k'} + \frac{\partial B_{i'}}{\partial\gamma_{li}}B_{j'}\frac{\partial B_{k'}}{\partial\gamma_{lj}} +$$
$$\frac{\partial B_{i'}}{\partial\gamma_{lj}}\frac{\partial B_{j'}}{\partial\gamma_{li}}B_{k'} + \quad (19)$$
$$B_{i'}\frac{\partial^2 B_{j'}}{\partial\gamma_{li}\partial\gamma_{lj}}B_{k'} + B_{i'}\frac{\partial B_{j'}}{\partial\gamma_{li}}\frac{\partial B_{k'}}{\partial\gamma_{lj}} + \frac{\partial B_{i'}}{\partial\gamma_{lj}}B_{j'}\frac{\partial B_{k'}}{\partial\gamma_{li}} +$$
$$B_{i'}\frac{\partial B_{j'}}{\partial\gamma_{lj}}\frac{\partial B_{k'}}{\partial\gamma_{li}} + \quad (20)$$

$$B_{i'}B_{j'}\frac{\partial^2 B_{k'}}{\partial\gamma_{li}\partial\gamma_{lj}} \tag{21}$$

Showing the partial derivatives of one of the polynomials $B_{i'}, B_{j'}, B_{k'}$ is sufficient to complete the derivation of the Jacobian matrix (note that in relation with (10), $B_{i'} \equiv B(s_G, i)$):

$$\frac{\partial B_{i'}}{\partial\gamma_{li}} = \frac{\partial B_{i'}}{\partial s_G}\frac{\partial s_G}{\partial\gamma_{li}}, \frac{\partial^2 B_{i'}}{\partial\gamma_{li}\gamma_{lj}} = \frac{\partial^2 B_{i'}}{\partial s_G^2}\frac{\partial s_G}{\partial\gamma_{lj}}\frac{\partial s_G}{\partial\gamma_{li}}, \tag{22}$$

with

$$\frac{\partial s_G}{\partial\gamma_{li}} = \mathbf{a}\sum_{i'=0}^{3}\sum_{j'=0}^{3}\sum_{k'=0}^{3} d^n_{li'j'k',s_l} B(s_l, i')B(t_l, j')B(u_l, k') \tag{23}$$

$$\frac{\partial^2 s_G}{\partial\gamma_{lj}\partial\gamma_{li}} = 0. \tag{24}$$

ACKNOWLEDGMENTS

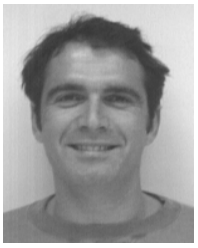
The authors would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Information Technology Research Center (ITRC) of Ontario, and the University of Toronto.

Authors' Note: *Animations demonstrating the principles of our techniques can be found at the following web site:* `http://www.dgp.toronto.edu/people/pfal/animations.html`.

While these animations are not essential to understanding the techniques, they do convey qualities of the motions which cannot easily be described. In order to limit file size, the image quality of the above animations is modest.

REFERENCES

[1] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 21, no. 4, pp. 35–44, 1987.
[2] T.W. Sederberg and S.R. Parry, "Free-Form Deformations of Solid Geometric Models," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 20, no. 4, pp. 151–160, Aug. 1986.
[3] S. Coquillart and P. Jancène, "Animated Free-form Deformation: An Interactive Animation Technique," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 25, no. 4, pp. 23–26, July 1991.
[4] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically Deformable Models," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 21, no. 4, pp. 205–214, July 1987.
[5] D. Terzopoulos and K. Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 22, no. 4, pp. 269–278, Aug. 1988.
[6] D. Metaxas and D. Terzopoulos, "Dynamic Deformation of Solid Primitives with Constraints," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 26, pp. 309–312, July 1992.
[7] D.R. Haumann, J. Wejchert, K. Arya, and B. Bacon, "An Application of Motion Design and Control in Physically-Based Animation," *Proc. Graphics Interface '91*, pp. 279–286, 1991.
[8] J.E. Chadwick, D.R. Haumann, and R.E. Parent, "Layered Construction of Deformable Animated Characters," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 23, no. 3, pp. 243–252, July 1989.
[9] A. Pentland and J. Williams, "Good Vibrations: Modal Dynamics for Graphics and Animation," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 23, no. 3, pp. 215–222, July 1989.
[10] A. Witkin and W. Welch, "Fast Animation and Control of Nonrigid Structures," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 24, no. 4, pp. 243–252, Aug. 1990.
[11] D. Baraff and A. Witkin, "Dynamic Simulation of Nonpenetrating Flexible Bodies," *Proc. ACM SIGGRAPH: Computer Graphics*, vol. 26, no. 2, pp. 303–308, July 1992.
[12] D. Terzopoulos and H. Qin, "Dynamic NURBS with Geometric Constraints for Interactive Sculpting," *ACM Trans. Graphics*, vol. 13, no. 2, pp. 103–136, Apr. 1994.
[13] M. van de Panne and E. Fiume, "Sensor-Actuator Networks," *Proc. ACM SIGGRAPH: Computer Graphics*, pp. 335–342, Aug. 1993.
[14] M. van de Panne, R. Kim, and E. Fiume, "Synthesizing Parameterized Motions," *Proc. Fifth Eurographics Workshop Animation and Simulation*, Oslo, Sept. 1994.
[15] X. Tu and D. Terzopoulos, "Artificial Fishes: Physics, Locomotion, Perception and Behavior," *Proc. ACM SIGGRAPH: Computer Graphics*, pp. 43–50, July 1994.
[16] J. Christensen, J. Marks, and J.T. Ngo, "Automatic Motion Synthesis for 3D Mass-spring Models," Technical Report MERL TR95-01, 1995, to appear in *Visual Computer*.
[17] A. Watt and M. Watt, *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.
[18] P.E. Nikravesh, "Spatial Kinematic and Dynamic Analysis with Euler Parameters," *Computer Aided Analysis and Optimization Mechanical System Dynamics*, E.J. Haug, ed., vol. 9 *NATO ASI, F*, pp. 261–281. Springer-Verlang, 1984.
[19] J.B. Marion and S.T. Thornton, *Classical Dynamics of Particles and Systems*, third edition. Harcourt Brace Jovanovich, 1988.
[20] D. Zeltzer, "Motor Control Techniques for Figure Animation," *IEEE Computer Graphics and Applications*, pp. 53–59, Nov. 1992.
[21] J.K. Hodgins and M.H. Raibert, "Biped Gymnastics," *Int'l J. Robotics Research*, vol. 9, no. 2, pp. 115–132, Apr. 1990.
[22] G.S.P. Miller, "The Motion Dynamics of Snakes and Worms," *Proc. SIGGRAPH '88*, vol. 22, no. 4, pp. 169–178, Aug. 1988.
[23] R. Grzeszczuk and D. Terzopoulos, "Automated Learning of Muscle-based Locomotion Through Control Abstraction," *Proc. ACM SIGGRAPH: Computer Graphics*, pp. 63–70, Aug. 1995.
[24] M. van de Panne, R. Kim, and E. Fiume, "Virtual Wind-up Toys for Animation," *Graphics Interface*, pp. 208–315, 1994.
[25] J.T. Ngo and J. Marks, "Spacetime Constraints Revisited," *Proc. ACM SIGGRAPH: Computer Graphics*, pp. 343–350, Aug. 1993.
[26] K. Sims, "Evolving Virtual Creatures," *Proc. Siggraph '94, ACM Computer Graphics*, pp. 15–22, 1994.
[27] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 671–680, May 1983.



Petros Faloutsos is a PhD candidate in the Department of Computer Science at the University of Toronto, where he is currently working on computer animation, control, and dynamic modeling. He received his BSc in electrical engineering from the National Technical University of Athens, Greece, in 1993, and his MSc in computer science from the University of Toronto in 1995.

Michiel van de Panne received his BSc in electrical engineering from the University of Calgary in 1987, and his MASc and PhD in electrical and computer engineering from the University of Toronto in 1989 and 1994, respectively. He is an assistant professor in the Department of Computer Science at the University of Toronto, where he is an active member of the Dynamic Graphics Project. His interests include computer animation, control and simulation techniques, robotics, and selected topics in modeling and rendering.

Demetri Terzopoulos (S'78, M'85) received the BEng. degree with distinction in honors electrical engineering and the MEng. degree in electrical engineering from McGill University, Montreal, Canada, in 1978 and 1980, respectively, and the PhD degree in artificial intelligence from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1984.

He is a professor of computer science and electrical and computer engineering at the University of Toronto, where he leads the Visual Modeling Group, and is a fellow of the Canadian Institute for Advanced Research. From 1985-92, he was affiliated with Schlumberger, Inc., serving as program leader at research labs in Palo Alto, California, and Austin, Texas. During 1984-85, he was a research scientist at the MIT Artificial Intelligence Lab, Cambridge, Massachusetts. He has been a consultant to Intel, Digital, Hughes, NEC, Ontario Hydro, and Schlumberger.

His published works include more than 180 scientific articles, primarily in computer vision and graphics, and also in computer-aided design, medical imaging, artificial intelligence, and artificial life, including the recent edited volumes *Real-Time Computer Vision* (Cambridge University Press, 1994), and *Animation and Simulation* (Springer-Verlag, 1995). His contributions have been recognized with several awards. In 1996, the Natural Sciences and Engineering Research Council of Canada awarded him the E.W.R. Steacie Memorial Fellowship. His other prizes include three University of Toronto Excellence Awards, an award from the American Association for Artificial Intelligence in 1987 for his work on deformable models in vision, an award from the IEEE in 1987 for introducing active contours ("snakes"), an award from NICOGRAPH in 1996 for his work on human facial modeling and animation, and awards from the International Digital Media Foundation in 1994 and from Ars Electronica in 1995 citing his work on artificial animals. He currently serves on the editorial boards of the journals *Videre*, *Medical Image Analysis*, *Graphical Models and Image Processing*, and the *Journal of Visualization and Computer Animation*. He has served on ARPA, NIH, and NSF advisory committees and is a member of the New York Academy of Sciences and Sigma Xi.