Dynamic Animation Synthesis with Free-Form Deformations

Petros Faloutsos
Michiel van de Panne
Demetri Terzopoulos

Abstract

Free-form deformations (FFDs) have long been a popular tool in modeling and key-frame animation. This paper extends the use of FFDs to a dynamic setting. A goal of this work is to enable normally rigid objects, such as teapots and tables, to come alive and learn to move about. Objects are assigned mass distributions and deformation properties, which allow them to translate, rotate, and deform according to internal and external forces. The primary contributions are threefold. First, a dynamic formulation of FFDs is presented. It is based on deformation modes which are tailored by the user and are expressed in terms of FFDs. Second, the formulation allows for a hierarchy of dynamic FFDs which can be used to model local as well as global deformations. Third, the deformation modes can be active and used as a basis for locomotion.

Category: Regular paper

1 Introduction

Deformations play an important role in computer graphics, allowing otherwise uninteresting objects to take useful and interesting shapes. Commercial animated films are full of inanimate characters that come to life and need a way to move as living creatures do. The dinner scene in the film "The Beauty and the Beast" produced by Disney is a prime example. The cutlery, pots and all types of household objects are seen to be dancing, singing, and performing like real actors. Their motions are a result of smooth deformations of their original shape. Computer graphics can provide helpful tools for both modeling and animating such characters. Free-form deformations (FFDs) are a popular tool for modeling and animating deformable objects. Such animations can be tedious to create, however, because of the large number of control points which need to be specified at keyframes.

This paper proposes a means of bringing FFDs into a dynamic environment. In its simplest form, this implies assigning mass and inertial properties to a solid deformed object and being able to simulate its passive motion. The formulation to be presented here makes the deformations themselves dynamic, which we subsequently refer to as *dynamic FFDs*. The deformations can also be active, meaning that a normally rigid, static object can locomote or jump under its own internal forces. This offers an opportunity to help apply physics to generating cartoon-style, expressive animations[1]. Lastly, the dynamic FFDs can be hierarchical, allowing for proper interactions between local and global deformations.

Our representation of dynamic FFDs makes use of deformation modes. Deformation modes restrict the shape of an object to those expressible by a set of modal amplitudes.

There are two fundamental reasons for wanting to use such modes in a dynamic FFD formulation. First, an animator can tailor the deformation modes, thereby having control over the desired shapes which can occur during simulation. Second, using deformation modes allows for an efficient dynamics simulation, as each deformation mode effectively acts as a single degree of freedom.

A first example of how one can use dynamic FFDs to make static objects come alive is shown in Fig. 1. The apple in this figure is equipped with a global lattice as well as a local lattice around the leaf. The teapot in Fig. 1 is also equipped with both global and local deformation modes. Snapshots from a dynamic animation produced for this character are shown in Fig. 15, which shows the motion resulting from a force being applied directly to the center of the lid. The force ceases to exist at the 13th frame.

The second example, seen in Fig. 2, shows a cartoon car driving on a bumpy road. In this case, the car has a global deformation mode assigned to it as well as a local deformation mode for each wheel. The car is rolling on the terrain as a result of a constant, horizontal, driving force.

The last example illustrates how a table can come to life using active FFD deformation modes. Fig. 3 shows a periodic motion which has been automatically synthesized for the table, given a set of user-supplied deformation modes. The deformation space was searched to optimize for the distance traveled.

The remainder of the paper is organized as follows. Section 2 presents the details of the dynamics formulation in addition to previous related work. Section 3 presents an animation system based on the dynamics formulation and explores its capabilities. Section 4 looks at how active deformations can be used to achieve cartoon-style animations, such as walking tables or hopping teapots. An optimization algorithm is used to

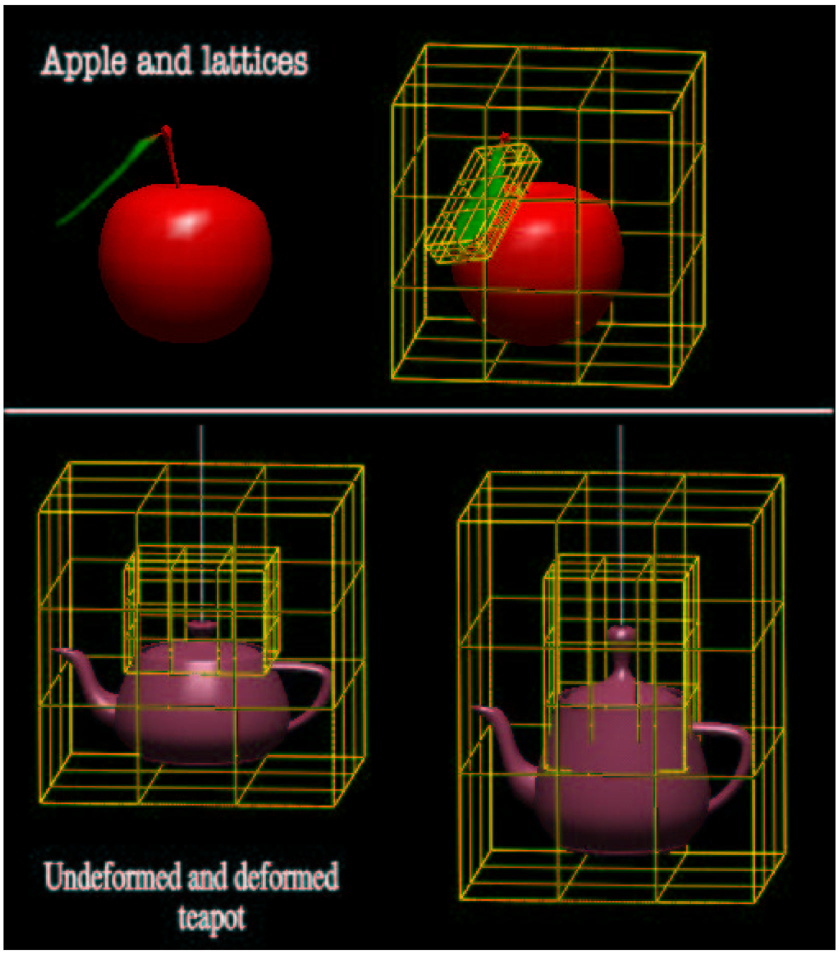

Figure 1: Local and global deformations using dynamic FFDs

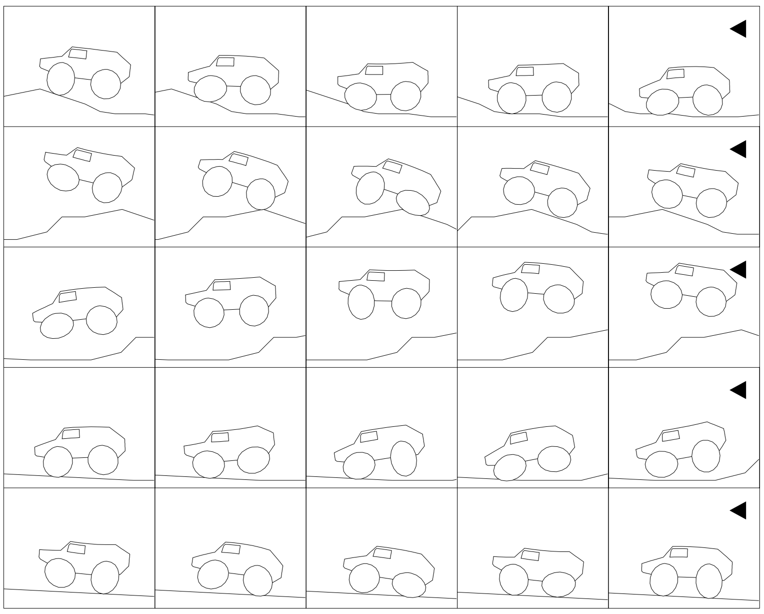

Figure 2: A cartoon car simulation using dynamic FFDs

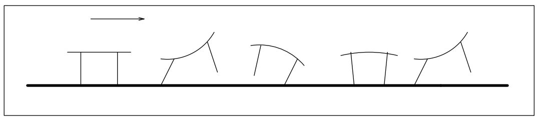

Figure 3: A table performing a bounding motion

automate the search for suitable control strategies. Lastly, section 5 presents conclusions and future work. Note that in figures which illustrate animations,the order in which to read the images is indicated using an arrowhead.

2 Dynamic Simulation of Deformable Models

The physical simulation of rigid and flexible objects involves four major problems: modeling the object, formulating the equations of motion, detecting collisions, and resolving collisions. In our work we are most concerned with the first two of these problems. Of particular interest to us are approaches that integrate cleanly with well-known existing tools. For this reason, we seek a dynamics formulation based upon free-form deformations.

2.1 Prior Work

The geometry and dynamics of deformable objects can be modeled using one of several different methods. In general, all methods are numerical approximations of some sort, although some are better than others. Several basic requirements are called for, however. First, the model must have a means of defining a mass density associated with the geometry. Second, it must have a restricted number of degrees of freedom to allow for efficient simulation. Third, the models must have a mechanism to take into account both internal and external forces. In general, we can distinguish between two main approaches towards the modeling of deformable objects, *implicit*[1] and *parametric*.

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

Implicit deformations work by embedding an object in a lattice which is subsequently warped, thus deforming the embedded object with it. The deformation is defined by a function that maps a point of the undeformed space onto the deformed one. Free-form deformations (FFDs) [2] are a primary example of implicit deformations. Animating the control points which define an FFD lattice will animate the deformation imposed on the object. The animator is typically responsible for designing the necessary deformations. The technique described in [3] provides useful generalizations on creating keyframed FFD animations.

Parametric or *explicit* formulations define deformations directly on flexible models. Models are discretized with respect to material coordinates by using finite differences or finite elements. The most straightforward model consists of a network of springs and point masses. The use of elastic models is first presented in the context of graphics in [4]. The models have been generalized to include visco-elastic properties and inelastic behavior such as fracture [5]. In [6], a model based on superquadrics is used that incorporates the global shape parameters of a conventional superquadric with the local degrees of freedom of a spline. In [7] deformable objects are animated using spring-mass models and wind fields.

Several techniques using implicit models have been proposed as a means to model dynamic deformations. In [8], polynomial global deformations are used in a physics-based model. The set of allowed deformations are all those expressible as a linear transformation of the state parameters. Our work makes use of a somewhat similar idea as a point of departure. The control method used in [8] is based on following motion paths. In [9] global implicit deformations are used in conjunction with physics based-simulation for the animation of polygonal and parametric objects. This formulation is similar to the one presented in [8], uses deformations that are linear with respect to the state of the object, and is restricted to passive objects.

In [10], deformable surfaces are modeled using NURBS which are given dynamic properties. The same work also implements a dynamic deformation technique using FFDs, although the formulation and application is different from the one we propose. In [11], implicit deformations are used to model the muscle deformation of articulated characters. The main goal of the deformation aspect of this work is to model the deformation of the muscle as the relative position of the associated bones change. FFD lattices are attached appropriately on the skeleton whose dynamic motion deforms the lattices. The dynamics are applied directly to the lattice and not to the muscle model itself.

Simultaneously with our work, [12] shows a method of automatically synthesizing controllers for simple creatures using global implicit deformations and stochastic search. Objects are modeled using spring-mass lattices that can undergo a set of canonical global implicit deformations. Thus, different objects may use different lattices whose shape is restricted by the shape of the object they model. Thus, deformations should be defined separately for each lattice. Lastly, the work in [13] is an innovative application of principal deformation modes to deformable animated objects. An advantage of implicit models is that they can deform a simple object and a complex object with equal ease.

Our work involves only implicit deformations and, in particular, dynamic FFDs. Several features of the proposed dynamic FFDs are unique with respect to previous work on implicit methods. Our approach provides the user with an intuitive method to design deformation modes which can then be imposed on different objects and used to produce a compact, limited set of allowable deformations. Our formulation allows for a hierarchy of deformations which are applied on objects in a non-linear fashion with respect to the state parameters. As a result, the system matrix is not constant, in contrast with [8, 9]. In addition, we make the deformation modes active and implement a complete motion control and motion synthesis system for flexible characters.

2.2 Our Dynamic Model

Existing dynamic models are capable of representing many types of animated objects. However, they are mostly oriented towards articulated figures [11, 14, 15], realistic creatures [16] or objects not capable of autonomous motion [9, 6, 13, 4, 10, 8]. Most 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 for characters of this type.

The approach begins with the user designing the allowable deformation modes. Fig. 4 shows an example of two deformation modes created for a table and how they interact. When the dynamic motion of this table is simulated, the shape of the table is restricted to those expressible with the two modal amplitudes. Deformation modes provide a way of adding actuation to characters which otherwise have no moving parts. Deformation modes 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.

2.3 Deformation Modes

A deformation mode is defined using a single instance of an FFD deformation, specified using a user-supplied matrix $\mathbf{d} = [\mathbf{d}_x \;\; \mathbf{d}_y \;\; \mathbf{d}_z]$ which defines the maximum displacement of the lattice points along each dimension. As shown in Fig. 4, multiple deformations are

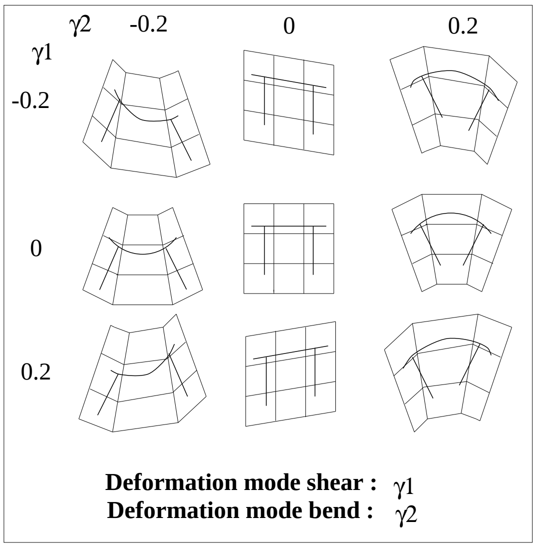

Figure 4: FFD deformation modes designed for a table.

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 $\mathbf{d}$. Our implementation uses a $4 \times 4 \times 4$ control point lattice for the FFD and uses Bernstein polynomials as the basis functions. The 2D examples in Fig. 4 show deformation modes, bend and shear, and their linear combinations.

In addition to being able to deform, the object is free to translate and rotate in 3D-space. Translation and rotation are effected on a global coordinate system with respect to which global deformations are considered, as shown in Fig. 5. We assume the following order in the application of the motions of the object: (1) local deformations, (2) global deformations, (3) rotation, (4) translation. Assuming D_G global deformation modes, D_l deformation modes associated with the l-th local lattice and assuming L local lattices, the degrees of freedom for the object are given by the vector:

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

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

Euler angles as a means to parameterize rotation have two disadvantages. First, the order in which they are applied can change the resulting rotation and second, certain series of rotations may lead to singularities such as the Gimbal lock [17]. For this reason interpolation between Euler angles is generally avoided and, instead, interpolation of quaternions [18] is used. In our formulation, the order in which Euler angles are applied is well-defined and dictated by the equations of motion. During simulation, the Euler angles are only updated as part of the numerical integration of the equations of motion. The simulation and the numerical integration steps do not involve interpolation. In addition, the change in value for each of the Euler angles is much less than 90 degrees between subsequent timesteps. This eliminates singularities like the Gimbal lock, although we could have used equally well quaternions in our formulation. We have chosen to use the Euler angles for their simplicity. However, in the keyframing component of our system rotations are interpolated using quaternions. The parameters specified by $\mathbf{q}$ are a set of generalized degrees of freedom which serve to fully describe the position of all points on a deformed object. The following expression describes the world coordinates of a point, given its local lattice coordinates (s_l, t_l, u_l) and the generalized coordinates $\mathbf{q}$:

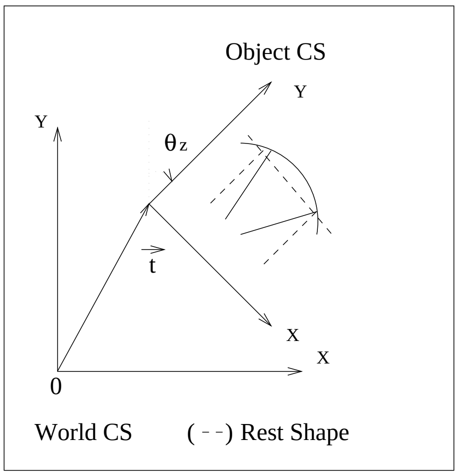

Figure 5: The table moving in space using a bend deformation mode.

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

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

where $\mathbf{P}$ is an object point in world coordinates, index G indicates quantities associated with the global lattice, similarly index l with respect to the l-th local lattice, $\mathbf{t}$ is the vector of translation parameters, $\mathbf{R}$ is the compound rotation matrix, γ_n is the amplitude of the n-th deformation mode, d^n_{ijk} are constants which define the n-th deformation mode of the i, j, k-control point of a lattice, $B(s,i)$ is the Bernstein polynomial $\binom{3}{i}(1-s)^{3-i}s^i$, $\mathbf{L}_{ijk}$ are the undeformed lattice coordinates, D is the number of deformation modes defined for a lattice by the animator, and $\mathbf{a}$ is a constant matrix that transforms the local lattice coordinates (s_l, t_l, u_l) to global lattice coordinates (s_G, t_G, u_G). Equation (3) calculates the new parameter space coordinates (s_G, t_G, u_G) with respect to the global lattice for those points that are affected by local deformations. This is explained furhter in the following section. Equations (2) and (3) are the starting point for the simulation of the dynamics of an object.

2.4 Hierarchical Deformations

To implement hierarchical, deformations we 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. 1. We apply local deformations before the global deformations using Equation (3). The dynamics formulation itself will ensure that a force which produces a local deformation can also affect the global deformations.

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 appropriate points on the boundary of the local lattices. In the example shown in Fig. 1 this is done by ensuring that the deformation modes designed for the local lattice do not affect the bottom two rows of control points.

We have implemented only two levels of deformations for demonstration purposes. However, the mechanics of the formulation presented above can be easily generalized to an arbitrarily 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 Equation (3).

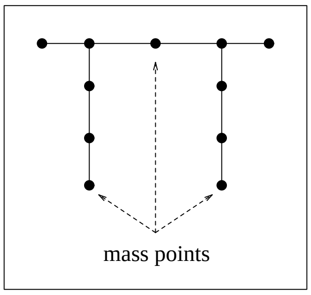

Figure 6: Mass distribution for a 2D-table

2.5 Equations of Motion

To incorporate dynamics into our flexible models we use a Lagrangian formulation [19]. A mass distribution is associated with the object by discretizing the object in material coordinates using mass points. For example, a 2D-table could be assigned a discrete mass distribution as shown in Fig. 6. In practice, approximating an object's distribution using $4 - 10$ mass-points is sufficient to yield convincing motions. These mass-points are embedded in the same space as the object geometry and are thus also affected by local and global deformations. 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:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q_k}} - \frac{\partial \mathcal{L}}{\partial q_k} - Q_k = 0, \tag{4}$$

where $\mathcal{L}$ is the Lagrangian, q_k is the k-th generalized coordinate ($k = 1, ..., D$), 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 and include potential energies as generalized forces, $\mathbf{Q}$. Appendix A contains the algebra that leads from Equation (4) to the equations of motion.

Assuming that **Q** includes internal and external forces, the equations of motion have the following general form, as shown in Appendix A:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q} + \frac{1}{2}\dot{\mathbf{q}}^T \frac{\partial \mathbf{M}}{\partial \mathbf{q}}\dot{\mathbf{q}} - \left(\frac{\partial \mathbf{M}}{\partial t}\right)\dot{\mathbf{q}}. \tag{5}$$

The above equations form a system of the form $\mathbf{Mx} = \mathbf{b}$, of $dim(\mathbf{q})$ coupled equations with $dim(\mathbf{q})$ unknowns. The system is of second order with respect to $\mathbf{q}$ and needs to be numerically integrated forward through time, which is done as follows. At each time step of the simulation, the linear system (5) is solved for $\ddot{\mathbf{q}}$ using Choleski factorization. Once solved, we obtain the new values for $\mathbf{q}$, $\dot{\mathbf{q}}$ by integrating twice. There is a number of methods we can use to do the integration. For demonstration purposes we have chosen to use using Euler integration and finite differences as follows:

$$\begin{aligned} \dot{\mathbf{q}}_n &= \frac{\mathbf{q}_n - \mathbf{q}_{n-1}}{\Delta t}, \\ \ddot{\mathbf{q}}_n &= \frac{\mathbf{q}_n - 2\mathbf{q}_{n-1} + \mathbf{q}_{n-2}}{\Delta t^2}. \end{aligned}$$

2.6 External Forces

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, Q. This is done as follows: $\mathbf{Q}_F = \mathbf{J}^T\mathbf{F}$ where $\mathbf{J}$ is the Jacobian matrix presented in Appendix A .

Ground contact is modeled using a penalty method which simulates the ground force

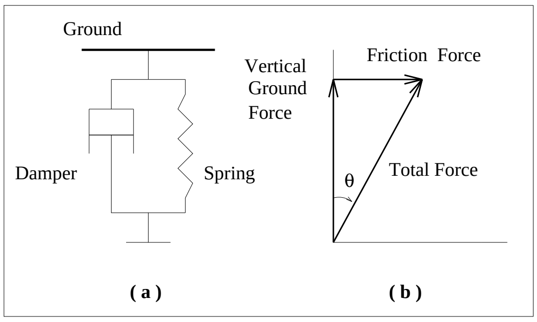

Figure 7: Ground force model

using a spring and damper unit as shown in Fig. 7 (a). When a point on the object passes below 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 have been chosen such that the interpenetrations caused by collisions are sufficiently small.

Collision detection and resolution are not a research issue in this work. We have implemented existing simple methods just for demonstration purposes. In particular, the system automatically samples the object at a set of sample points on the surface such that the relative distances between these collision points are sufficiently small. At each timestep all the collision points are checked against the ground. For each collision point penetrating the ground, a penalty force is activated as described above. There is no need to associate mass with the collision points because of the implicit deformations we use and the Lagrangian dynamics. Each ground force is transformed to a generalized force using the Jacobian as described above. For the teapot that appears in Fig. 13 we used have 300 collision points.

2.7 Internal Deformations

As an object deforms, it builds up internal strain energy that resists deformation. These energies are defined by the user and are a function of the deformation amplitudes γ_i. Typically, we associate deformations with a potential energy of the form $V_i = K(\gamma_i - \gamma_{i0})^2$, where K is a constant and the resting state γ_{i0} can be varied over time to allow for active control over a deformation, effectively forming an object's "muscles". Assuming that the potential energy V corresponds to a conservative field, the associated force is $\mathbf{F} = -\nabla V$.

3 A Dynamic Animation System

We have implemented an interactive animation system using the dynamic FFD formulation. Fig. 8 illustrates the conceptual components of our system. Generally, our system can automatically turn any geometric model into a dynamic one by automatically computing mass distributions, attaching FFD-lattices, and giving meaningful 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. 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. The algorithm we used ensures that all the FFD lattices contain a number of mass points and that the actual geometric center of the object is near to the center of mass of the computed mass distribution. Note that in this context an object is not associated with a stiffness matrix. Instead the dynamic behaviour of the object is determined by the deformation modes, each of which has its own stiffness

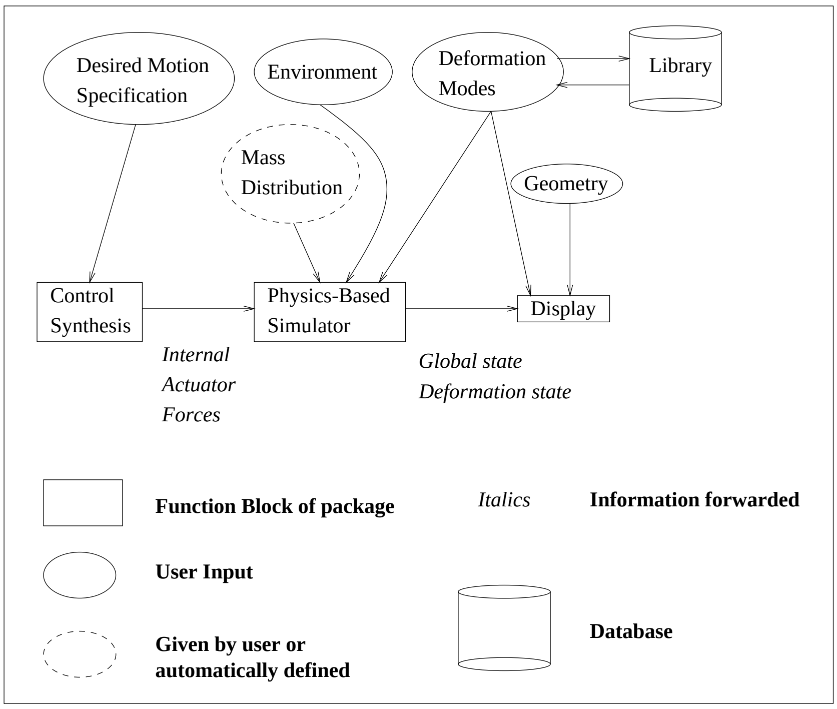

Figure 8: Overview of the system

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 properties of the desired motion. Fig. 9 illustrates an animation 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.

3.1 Key-Framing

Once an object and a set of deformation modes is loaded, the user can specify a deformation mode and the desired amplitude to interactively deform the object. To prevent

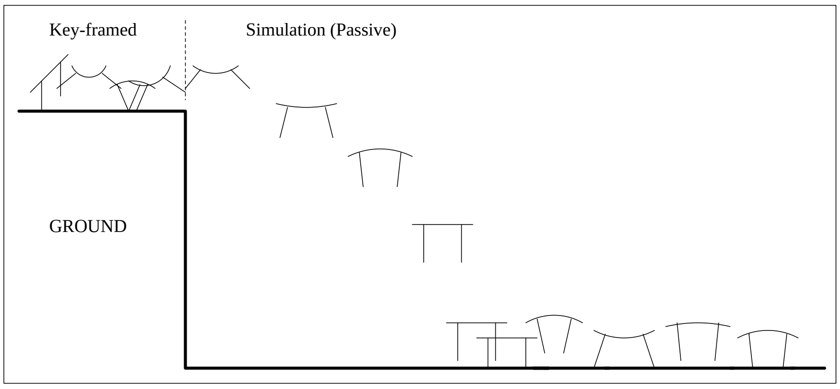

Figure 9: Table jumping off a cliff

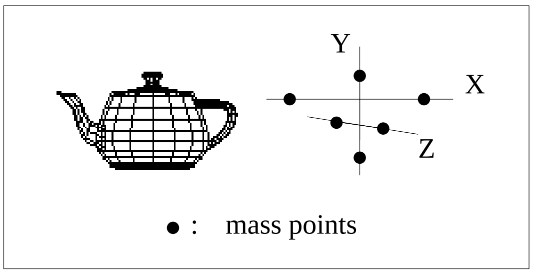

Figure 10: A 3D teapot and an associated mass distribution

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. In Fig. 9 the first part of the motion is key-framed.

3.2 Physics-Based Simulation

The physics-based simulator of our system implements a numerical integrator for the Lagrangian equations of motion as given by Equation (5). The user can interactively modify a number of parameters, such as the ground stiffness, flags that turn on/off 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

Table 1: Running 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

appropriate environment.

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 objects such as the teapot of Fig. 10, apples, and glasses, each discretized with 6 mass points and equipped with between $300-500$ collision points. The motions produced were appealing and sufficiently convincing. The complexity of the simulation is linear with respect to the number of point masses and the number of lattice points. The number of total deformation modes D_t determines the dimension of the linear system that is solved in each simulation iteration, thus the complexity of the simulation process is $O(D_t^3)$. To further improve the efficiency of the simulation, we allow the user to turn on a mechanism which deactivates (eliminates) deformation degrees of freedom if the magnitudes of the associated amplitude and velocity are below a threshold. This allows for 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. To test the effectiveness of this feature we have performed two experiments. For the first one, the cartoon car shown in Fig. 2 rolls down a bumpy steep hill. It was equipped with one global deformation and one local deformation on each of the wheels. With the above efficiency feature deactivated the simulation consumed 133 seconds of CPU time, while with the feature activated it consumed 121 seconds. For the second experiment, the teapot shown in Fig. 1 performed a free-fall equipped with one global deformation and six local deformations: three on the handle and three on the spout. The simulation time without the efficiency mechanism was 12 seconds and with the mechanism was 7 seconds.

Stability is an important issue for a simulation based on numerical methods. The main factors affecting the stability of our system are the time step, the stiffness introduced by the collision penalty method, the mass distribution, and the chosen set of deformations. 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 for this object. In such a case, the system is unstable. The generalized coordinates of an object should be independent from each other as required by the Lagrangian dynamics formulation. Thus, if the same deformation mode is defined twice the system will be ill-conditioned. The more similar two deformation modes are, the less numerically stable the system will be.

4 Motion Synthesis for Active Deformations

The dynamic models described in a previous section incorporate physics into a geometric model, and encompass both passive and active characters. For active characters, it is desirable to simulate them realistically, incorporating in the model the ability to use and control actuators in order to produce autonomous motion. For realistic movements, the motion should arise from control actions. There are many different methods that deal with the control of simulated objects [20, 21, 22, 14, 23, 16]. A simple-but-effective technique based on *cyclic pose control graphs* and suited for articulated figures is presented in [24]. The work in [15] extends this technique to acyclic graphs in order to achieve non-periodic motions.

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 [23, 25, 26, 14]. Motion and controller synthesis are often addressed using probabilistic methods because they are easy to implement, are suitable for searching large spaces, and can avoid local optima. 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 ones which produce suitable motions. Common search methods for the stochastic motion synthesis problem are *genetic algorithms* [25, 26] and *simulated annealing* [14, 23].

To our knowledge, previous work on controlling active deformable models have always used spring-mass models [22, 16, 23].

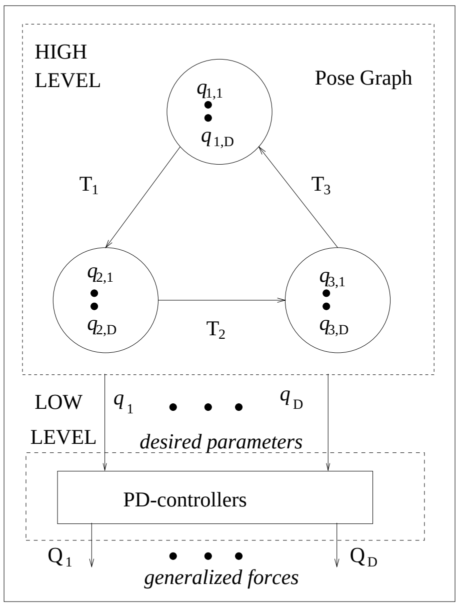

Figure 11: The control structure

4.1 Our Dynamic Control Model

The control of our characters is based on pose controllers which determine the high level control, as shown in Fig. 11. The pose control 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 $\mathbf{q}$ is defined in terms of the amplitudes of the deformation modes, $\mathbf{q} = (\gamma_1, \ldots, \gamma_D)$, where D is the total number of global and local deformation modes. Because we use open-loop control[2], the rigid body parameters are not used in the control process. Each state (pose) has an associated *transition* time. The latter specifies the time period for which the associated pose remains active. 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

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

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 directly adding deformation forces to the vector of generalized forces. We have chosen to use the latter method. A PD-controller is associated with each deformation mode and produces a generalized control force along the associated generalized coordinate. Control forces are defined as

$$\mathbf{Q}_r = -(\mathbf{C}(\mathbf{q} - \mathbf{q}_0) + \mathbf{D}\dot{\mathbf{q}}),$$

where $\mathbf{Q}_r$ are the generalized control forces, $\mathbf{C}$ is a diagonal stiffness matrix, $\mathbf{D}$ is a diagonal damping matrix and $\mathbf{q}_0$ are the desired pose parameters.

An animator can interactively specify a deformation mode to be active or passive. Active deformations modes are subject to control forces and they contribute to active 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 typically non-oscillatory. 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. In the current version of our system, we have implemented only *cyclic* pose graphs which result in periodic motions.

4.2 Motion Synthesis

Given an object and a set of deformation modes, it is convenient (and for complex

```
1. Specify an initial configuration $P$
2. Specify an initial temperature $T > 0$
3. Pick a freezing 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 calcu-
       late the cost $Cost(P')$
   (c) Let $\Delta = Cost(P') - Cost(P)$
   (d) If $\Delta \leq 0$ set $P = P'$
   (e) If $\Delta > 0$ set $P = P'$ with probability $e^{-\Delta/T}$
   (f) Set $T = r \times T$
6. Return P
```

Figure 12: Using simulated annealing to find suitable active control

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 only the case of articulated figures with the exception of [12, 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. All our experiments converged to a solution in less that 100 annealing iterations. The process begins with am 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 changing the initial one. The new controller is evaluated in the same way and the two controllers are compared. The best controller is kept and the procedure is repeated until a stopping criterion is met. Sometimes the process keeps a new controller which performs worse than the previous one as a means to avoid local optima. The algorithm used is summarized in Fig. 12. The value of r controls the number of iterations that the algorithm performs. A fixed number of iterations can also be used.

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 produced motion. The most important choice in defining an optimization process is the choice of the *cost* or *optimization function*. This function is minimized or maximized by the annealing process. The optimization function characterizes 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 optimization function depends on the complexity of the character and the desired motion. For simple motions such as walking, hopping, and shuffling, typical optimization functions do exist. 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. The optimization function that we used in all cases depended linearly on the control energy consumed, in order to prevent the synthesis of unrealistic high-energy motions. Specific functions are presented in the next section, where we discuss the results of the motion synthesis experiments.

4.3 Results

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

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. 3. The cost function used rewards the distance traveled and penalizes the energy used. Table 2 presents the cost functions and the associated result; $E_c = \sum_{i=1}^{D} abs(q[i] * Qk[i])$ where $\mathbf{Qk}$ 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.

A second example is that of a 3D teapot which learns to locomote using two active deformation modes. Fig. 13 shows the teapot shuffling. The same motion has been applied to the glass that appears in Fig. 14. In this example, the teapot is performing

Table 2: Cost functions

Cost function	*Comments*
$1/t_x$	Hopping motion with unrealistic jumps
E_c/t_x	Hopping motion, normal jumps
$E_c/(1.0 \times 10^7 \sqrt{t_x \dot{t}_x})$	Bounding motion
$E_c/(5.0 \times 10^7 \sqrt{t_x \dot{t}_x})$	Faster bounding motion (Fig. 3)
$E_c/(5.0 \times 10^7 \sqrt{t_x})$	Shuffling-like motion (Fig. 13)

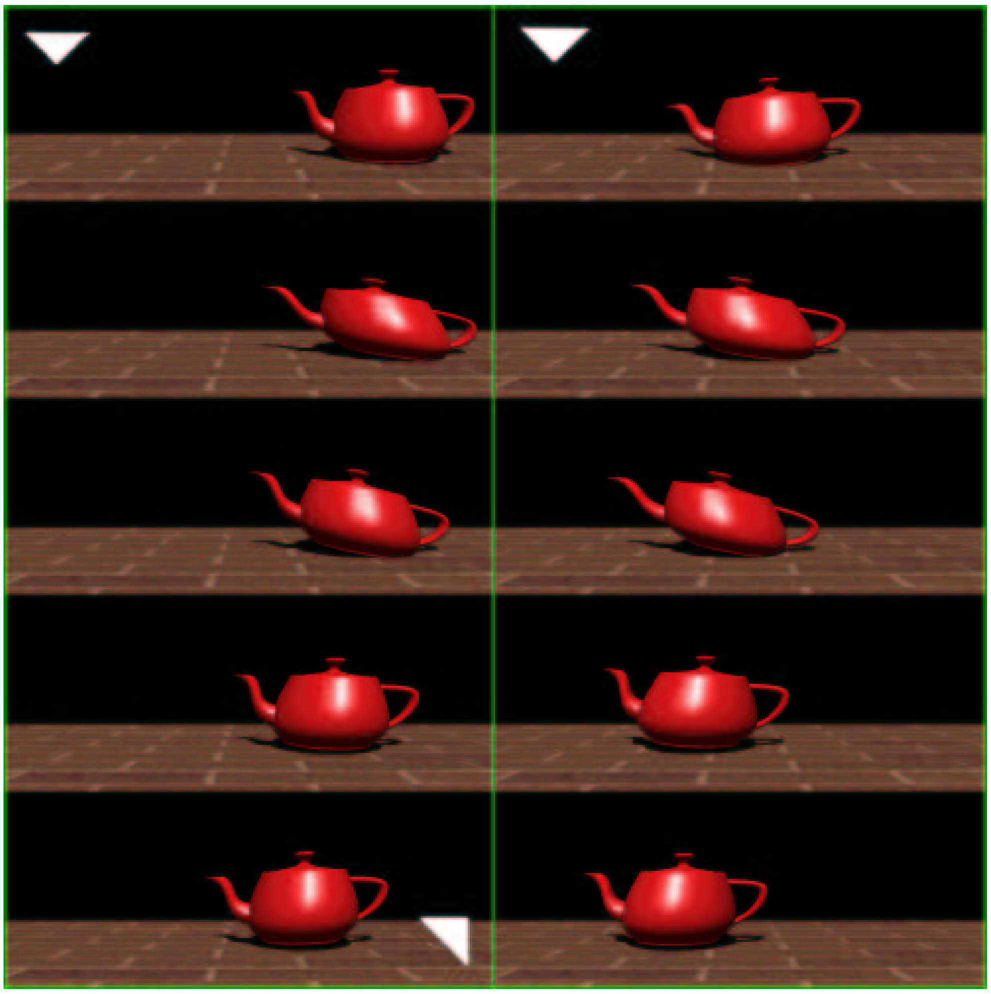

Figure 13: 3D Teapot performing a bounding motion

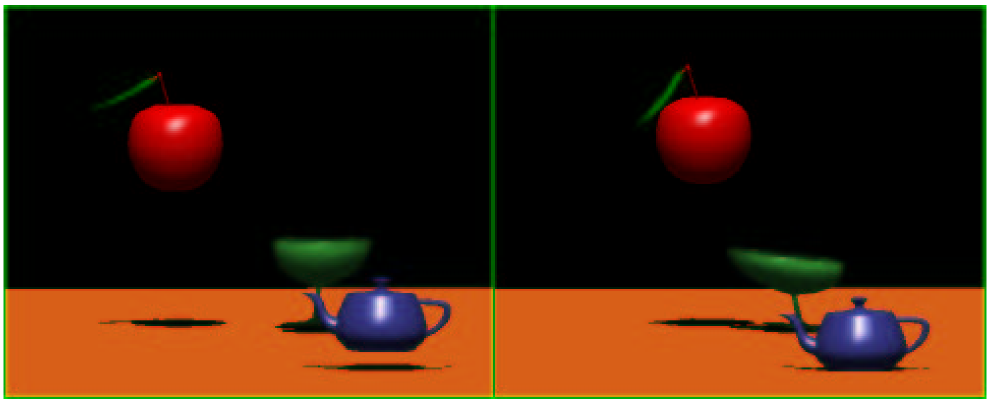

Figure 14: A cartoon race

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.

A more evident example of global and local deformation is presented in Fig. 15. The lattices are attached to the teapot as shown in Fig. 1. A spring is attached to the lid and lifts the teapot. The spring is removed at the 13th frame. Both the global and the local lattice are associated with the same stretch deformation. However, the global deformation is stiffer than the local one.

5 Conclusions

We have proposed and implemented a framework for the animation of deformable characters. Our approach takes traditional free-form deformations and extends them to a dynamic setting. The formulation provides for both local and global dynamic FFDs and ensures for their proper interaction. User-tailored deformations provide predictability of results and yield efficient dynamic simulations. The dynamic deformations can be made active, making the technique particularly suitable to transforming inanimate objects,

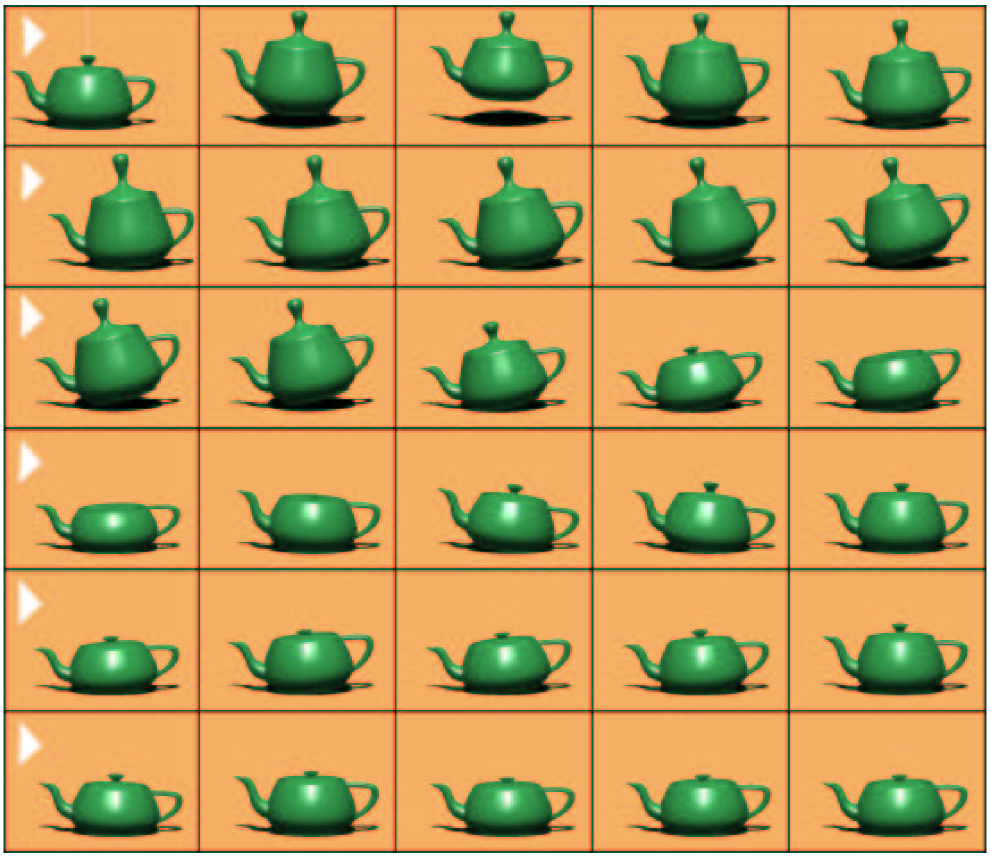

Figure 15: A teapot is lifted from the lid

such as teapots, into animate characters. And how should a squash-and-stretch teapot move? Existing motion-synthesis techniques can be applied to the active deformations to automate the process of answering such questions.

Free-form deformations are a tool one would likely use to make solid objects come alive 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 so typical to cartoon-style animation, libraries of deformation modes and standardized motions could be provided to animators. Different characters can make use of existing sets 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 deformations in non-linear ways. Dynamic constraints could be implemented in order to allow the construction of compound objects. This could be applied to articulated figures having deformable parts. We would also like to experiment with FFD blocks based B-spline basis function and hierarchical structures with shared control points. Lastly, to improve upon the dynamics, a complete and efficient method for detecting and resolving collisions can be implemented.

A Derivation of the Equations of Motion

This appendix derives the terms of Equation (4) and shows how to calculate the equations of motion (5). Starting from the Lagrangian and considering only the kinetic energy E we obtain $\mathcal{L} = E$. The object is discretized in material coordinates using point masses. The kinetic energy of the k-th point mass is $E_k = \frac{1}{2} m_k \dot{\mathbf{x}}_k^T \dot{\mathbf{x}}_k$, where m_k is the point mass and $\mathbf{x}_k$ is the position of the point in world coordinates. Using Equations (2) and (3) we can write E_k with respect to the generalized coordinates. To do that 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 L local lattices each one equipped with D_l deformation modes. The vector of generalized coordinates as defined in Equation (1) is:

$$\mathbf{q} = \begin{bmatrix} \mathbf{t}^T & \theta_x & \theta_y & \theta_z & \gamma_{G1} & \dots & \gamma_{GD} & \gamma_{11} & \dots & \gamma_{1D_1} & \dots & \gamma_{L1} & \dots & \gamma_{LD_L} \end{bmatrix}. \quad (6)$$

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

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{A} & \mathbf{B} & \mathbf{C}_1 & \dots & \mathbf{C}_L \end{bmatrix}, \quad (7)$$

where

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \theta_x}\mathbf{P} & \frac{\partial \mathbf{R}}{\partial \theta_y}\mathbf{P} & \frac{\partial \mathbf{R}}{\partial \theta_z}\mathbf{P} \end{bmatrix}, \\
\mathbf{B} &= [b_n] = \left[\mathbf{R} \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} \mathbf{d}^n_{G,ij} B(s_G, i) B(t_G, j) B(u_G, k) \right], \quad n = 1 \dots D_G, \\
\mathbf{C}_l &= [c_{lm}] = \left[\mathbf{R} \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} \left(\sum_{n=1}^{D_G} \mathbf{d}^n_{G,ij} + \mathbf{L}_{G,ijk}\right) \frac{\partial}{\partial \gamma_{lm}} (B(s_G, i) B(t_G, j) B(u_G, k)) \right], \\
m &= 1 \dots D_l.
\end{aligned}$$

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

$$E_k = \frac{1}{2} m_k \dot{\mathbf{x}}_k^T \dot{\mathbf{x}}_k = \frac{1}{2} m_k (\mathbf{J}_k \dot{\mathbf{q}})^T \mathbf{J}_k \dot{\mathbf{q}} = \frac{1}{2} m_k \dot{\mathbf{q}}^T \mathbf{J}_k^T \mathbf{J}_k \dot{\mathbf{q}}.$$

The total kinetic energy of the object is $E = \sum_k E_k$ and thus the Lagrangian is $\mathcal{L} = \sum_k E_k = \sum_k \frac{1}{2} m_k \dot{\mathbf{q}}^T \mathbf{J}_k^T \mathbf{J}_k \dot{\mathbf{q}} = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}}$ where $\mathbf{M} = \sum_k m_k \mathbf{J}_k^T \mathbf{J}_k$ is a symmetric generalized mass matrix. Having an expression for the Lagrangian, we can calculate the required derivatives for Equation (4) as follows

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

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

$$\frac{\partial \mathcal{L}}{\partial \dot{q}_k} = \sum_j M_{kj} \dot{q}_j \Rightarrow \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} = \sum_j M_{kj} \ddot{q}_j + \sum_j \frac{dM_{kj}}{dt} \dot{q}_j = \sum_j M_{kj} \ddot{q}_j + \sum_j \left(\sum_l \frac{\partial M_{kj}}{\partial q_l} \dot{q}_l\right) \dot{q}_j. \tag{8}$$

The second term that involves the Lagrangian in Equation (4), is calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial q_k} = \frac{1}{2} \dot{q}_k \frac{\partial \mathbf{M}}{\partial q_k} \dot{q}_k. \tag{9}$$

The derivatives of $\mathbf{M}$ required in Equations (8), (9) can be calculated as follows:

$$\frac{\partial \mathbf{M}}{\partial q_j} = \frac{\partial}{\partial q_j} \sum_k m_k \mathbf{J}_k^T \mathbf{J}_k = \sum_k m_k \left[\left(\mathbf{J}_k^T \frac{\partial \mathbf{J}_k}{\partial q_j}\right)^T + \mathbf{J}_k^T \frac{\partial \mathbf{J}_k}{\partial q_j}\right]. \tag{10}$$

Determining the partial derivatives of $\mathbf{J}$ with respect to the generalized coordinates $\mathbf{q}$, as required by Equation (10), is a somewhat lengthy, but mechanical process. We present them below for purposes of completeness. To follow the notation the reader is referred to the definition of the vector of generalized coordinates shown in Equation (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 Equation (7) evaluated with $\mathbf{R}$ being the identity matrix, $\mathbf{J}_I[q_k]$ the column of $\mathbf{J}_I$ that corresponds to the q -th 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 Equations (11) - (14). Overbraces are used to compress a number of similar columns in one. The partial derivatives of the Jacobian are

$$\frac{\partial \mathbf{J}}{\partial t_i} = \mathbf{O}_{dim(\mathbf{J})},\ i = x, y, z \tag{11}$$

$$\frac{\partial \mathbf{J}}{\partial \theta_j} = \left[\begin{array}{cccccc} \mathbf{O}_3 & \overbrace{\frac{\partial^2 \mathbf{R}}{\partial \theta_i \partial \theta_j}\mathbf{P}}^{i=x,y,z} & \overbrace{\frac{\partial \mathbf{R}}{\partial \theta_j}\mathbf{J}_I[\gamma_{Gi}]}^{i=1\ldots D_G} & \overbrace{\frac{\partial \mathbf{R}}{\partial \theta_j}\mathbf{J}_I[\gamma_{1i}]}^{i=1\ldots D_1} & \cdots & \overbrace{\frac{\partial \mathbf{R}}{\partial \theta_j}\mathbf{J}_I[\gamma_{Li}]}^{i=1\ldots D_L} \end{array} \right], \tag{12}$$

$$\begin{aligned} \frac{\partial \mathbf{J}}{\partial \gamma_{Gj}} = \Bigg[& \mathbf{O}_3 \quad \overbrace{\frac{\partial \mathbf{R}}{\partial \theta_i} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} d_{\gamma_{Gj}} BBB}^{i=x,y,z} \quad \overbrace{\mathbf{0}}^{i=1\ldots D_G} \quad \overbrace{\mathbf{R} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} d_{\gamma_{Gj}} \frac{\partial BBB}{\partial \gamma_{1i}}}^{i=1\ldots D_1} \\ & \ldots \quad \overbrace{\mathbf{R} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} d_{\gamma_{Gj}} \frac{\partial BBB}{\partial \gamma_{Li}}}^{i=1\ldots D_L} \Bigg], \end{aligned} \tag{13}$$

$$\begin{aligned} \frac{\partial \mathbf{J}}{\partial \gamma_{lj}} = \Bigg[& \mathbf{O}_3 \quad \overbrace{\frac{\partial \mathbf{R}}{\partial \theta_i} \mathbf{J}[\gamma_{lj}]}^{i=x,y,z} \quad \overbrace{\mathbf{R} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} d_{\gamma_{Gi}} \frac{\partial BBB}{\partial \gamma_{lj}}}^{i=1\ldots D_G} \quad \overbrace{\mathbf{0}}^{i=1\ldots D_1} \quad \ldots \quad \overbrace{\mathbf{0}}^{i=1\ldots D_{l-1}} \\ & \overbrace{\mathbf{R} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} (\sum_{n=1}^{D_G} \mathbf{d}^n + L^G_{ijk}) \frac{\partial^2 BBB}{\partial \gamma_{lj} \partial \gamma_{li}}}^{i=1\ldots D_l} \quad \overbrace{\mathbf{0}}^{i=1\ldots D_{l+1}} \quad \ldots \quad \overbrace{\mathbf{0}}^{i=1\ldots D_L} \Bigg]. \end{aligned} \tag{14}$$

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

Since the chain rule applies

$$\frac{\partial BBB}{\partial \gamma_{li}} = \frac{\partial B_{i'}}{\partial \gamma_{li}} B_{j'} B_{k'} + B_{i'} \frac{\partial B_{j'}}{\partial \gamma_{li}} B_{k'} + B_{i'} B_{j'} \frac{\partial B_{k'}}{\partial \gamma_{li}}, \tag{15}$$

$$\frac{\partial^2 BBB}{\partial \gamma_{li} \partial \gamma_{lj}} = \frac{\partial^2 B_{i'}}{\partial \gamma_{li} \partial \gamma_{lj}} B_{j'} B_{k'} + \frac{\partial B_{i'}}{\partial \gamma_{li}} \frac{\partial B_{j'}}{\partial \gamma_{lj}} B_{k'} + \frac{\partial B_{i'}}{\partial \gamma_{li}} B_{j'} \frac{\partial B_{k'}}{\partial \gamma_{lj}} + \frac{\partial B_{i'}}{\partial \gamma_{lj}} \frac{\partial B_{j'}}{\partial \gamma_{li}} B_{k'} + \tag{16}$$

$$B_{i'} \frac{\partial^2 B_{j'}}{\partial \gamma_{li} \partial \gamma_{lj}} B_{k'} + B_{i'} \frac{\partial B_{j'}}{\partial \gamma_{li}} \frac{\partial B_{k'}}{\partial \gamma_{lj}} + \frac{\partial B_{i'}}{\partial \gamma_{lj}} B_{j'} \frac{\partial B_{k'}}{\partial \gamma_{li}} + B_{i'} \frac{\partial B_{j'}}{\partial \gamma_{lj}} \frac{\partial B_{k'}}{\partial \gamma_{li}} + \tag{17}$$

$$B_{i'} B_{j'} \frac{\partial^2 B_{k'}}{\partial \gamma_{li} \partial \gamma_{lj}}. \tag{18}$$

Showing the partial derivatives of one of the polynomials $B_{i'}, B_{j'}, B_{k'}$ is sufficient to complete the presentation of the mathematics involving the Jacobian matrix. Note that in relation with Equation (7) $B_{i'} \equiv B(s_G, i)$.

$$\frac{\partial B_{i'}}{\partial \gamma_{li}} = \frac{\partial B_{i'}}{\partial s_G} \frac{\partial s_G}{\partial \gamma_{li}}, \quad \frac{\partial^2 B_{i'}}{\partial \gamma_{li} \gamma_{lj}} = \frac{\partial^2 B_{i'}}{\partial s_G^2} \frac{\partial s_G}{\partial \gamma_{lj}} \frac{\partial s_G}{\partial \gamma_{li}}, \tag{19}$$

with

$$\frac{\partial s_G}{\partial \gamma_{li}} = \mathbf{a} \sum_{i'=0}^{3} \sum_{j'=0}^{3} \sum_{k'=0}^{3} d^{i}_{li'j'k',s_l} B(s_l, i') B(t_l, j') B(u_l, k') \tag{20}$$

$$\frac{\partial^2 s_G}{\partial \gamma_{lj} \partial \gamma_{li}} = 0. \tag{21}$$

References

[1] J. Lasseter, “Principles of traditional animation applied to 3-d computer animation,” *Proceedings of 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," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 20, no. 4, pp. 151–160, August 1986.

[3] S. Coquillart and P. Jancène, "Animated free-form deformation: An interactive animation technique," *Proceedings of 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," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 21, no. 4, pp. 205–214, July 1987.

[5] K. Fleischer D. Terzopoulos, "Modeling inelastic deformation: Viscoelasticity, plasticity, fracture," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 22, no. 4, pp. 269–278, August 1988.

[6] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints," *Proceedings of 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," *Proceedings of Graphics Interface '91*, pp. 279–286, 1991.

[8] A. Witkin and W. Welch, "Fast animation and control of nonrigid structures," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 24, no. 4, pp. 243–252, August 1990.

[9] D. Baraff and A. Witkin, "Dynamic simulation of non-penetrating flexible bodies," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 26, no. 2, pp. 303–308, July 1992.

[10] D. Terzopoulos and H. Qin, "Dynamic NURBS with geometric constraints for interactive sculpting," *ACM Transactions on Graphics*, vol. 13, no. 2, pp. 103–136, April 1994.

[11] J. E. Chadwick, D. R. Haumann, and R. E. Parent, "Layered construction of deformable animated characters," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 23, no. 3, pp. 243–252, July 1989.

[12] J. Christensen, J. Marks, and J. T. Ngo, "Automatic motion synthesis for 3D mass-spring models," Tech. Rep., MERL TR95-01, 1995, To appear in the Visual Computer.

[13] A. Pentland and J. Williams, "Good vibrations: Modal dynamics for graphics and animation," *Proceedings of ACM SIGGRAPH: Computer Graphics*, vol. 23, no. 3, pp. 215–222, July 1989.

[14] M. van de Panne and E. Fiume, "Sensor-actuator networks," *Proceedings of ACM SIGGRAPH: Computer Graphics*, pp. 335–342, August 1993.

[15] M. van de Panne, R. Kim, and E. Fiume, "Synthesizing parameterized motions," *Fifth Eurographics Workshop on Animation and Simulation, at Oslo*, September 1994.

[16] X. Tu and D. Terzopoulos, "Artificial fishes: Physics, locomotion, perception and behavior," *Proceedings of ACM SIGGRAPH: Computer Graphics*, pp. 43–50, July 1994.

[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,” in *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, Edward J. Haug, Ed., vol. 9 of *NATO ASI, F*, pp. 261–281. Springer-Verlang, 1984.

[19] J. B. Marion and S. T. Thornton, *Classical Dynamics of Particles and Systems*, Harcourt Brace Jovanovich, Publishers, third edition, 1988.

[20] D. Zeltzer, “Motor control techniques for figure animation,” *IEEE Computer Graphics and Applications*, pp. 53–59, November 1992.

[21] J. K. Hodgins and M. H. Raibert, “Biped gymnastics,” *International Journal of Robotics Research*, vol. 9, no. 2, pp. 115–132, April 1990.

[22] G. S. P. Miller, “The motion dynamics of snakes and worms,” *Proceedings of SIGGRAPH ’88*, vol. 22, no. 4, pp. 169–178, August 1988.

[23] R. Grzeszczuk, “Automated learning of muscle-based locomotion through control abstraction,” *Proceedings of ACM SIGGRAPH: Computer Graphics*, pp. 63–70, August 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,” *Proceedings of ACM SIGGRAPH: Computer Graphics*, pp. 343–350, August 1993.

[26] K. Sims, “Evolving virtual creatures,” *Proceedings of 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 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. He 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.

Demetri Terzopoulos (S'78, M'85) was born in Greece. He received the B.Eng. degree with distinction in Honours Electrical Engineering and the M.Eng. degree in Electrical Engineering from McGill University, Montreal, Canada, in 1978, and 1980, respectively, and the Ph.D. degree in Artificial Intelligence from the Massachusetts Institute of Technology, Cambridge, MA, in 1984.

He is 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, CA, and Austin, TX. During 1984-85 he was a research scientist at the MIT Artificial Intelligence Lab, Cambridge, MA. He has been a consultant to Digital, Hughes, NEC, Ontario Hydro, and Schlumberger.

His published works include more than 150 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 Univ. Press ’94) and “Animation and Simulation” (Springer-Verlag ’95). 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 awards include three university 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 pioneering active contours or “snakes”, and awards from the Canadian Academy of Multimedia Arts and Sciences in 1994 and from Ars Electronica in 1995 citing his work on artificial animals. He currently serves on the editorial boards of the journals *Medical Image Analysis*, *Graphical Models and Image Processing*, and the *Journal of Visualization and Computer Animation*. He has served on ARPA and NIH advisory committees and is a member of the New York Academy of Sciences and Sigma Xi.