Fast Animation and Control of Nonrigid Structures

Andrew Witkin
William Welch
School Of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We describe a fast method for creating physically based animation of non-rigid objects. Rapid simulation of non-rigid behavior is based on global deformations. Constraints are used to connect non-rigid pieces to each other, forming complex models. Constraints also provide motion control, allowing model points to be moved accurately along specified trajectories. The use of deformations that are linear in the state of the system causes the constraint matrices to be constant. Pre-inverting these matrices therefore yields an enormous benefit in performance, allowing reasonably complex models to be manipulated at interactive speed.

Keywords — Constraints, Simulation, Animation

1 Introduction

A good deal of work has been done toward the use of physical simulation as a means of producing animation. Despite the wealth of impressive results, physically based modeling is not in wide use, because of the unfamiliarity and complexity of the methods, because of the computational expense of simulation, and because of the difficulty of controlling simulated objects. In this paper we present a fast and simple formulation for building and controlling models composed of non-rigid pieces. Our method has three major parts:

Non-rigid objects. Our model for non-rigid dynamics is based on global deformations with relatively few degrees of freedom. This formulation provides objects that deform in a geometrically simple but physically correct way. A further simplification is achieved by restricting attention to deformations that are linear in the system's state variables. This class includes both free-form deformations of the sort described in [14], and simpler affine deformations. One beneficial effect of this simplification is that the body's mass matrix is constant, allowing it to be pre-inverted.

Attachment constraints. Point-to-point attachment constraints are used to build complex models from the simple non-rigid pieces. Attachment constraints are implemented using a method fully described in [19], and related to those in [4] and [11]. The idea is to calculate a constraint force that accurately counters applied forces that would otherwise pull the pieces apart. Ordinarily, obtaining the constraint force requires the solution of a linear system at least once per time step. The linearity restriction, however, causes the constraint matrix to be constant, except when constraints are added or deleted. By pre-inverting this matrix, we need only perform a matrix multiplcation instead of an inversion at each step.

Motion control. The same machinery that supports attachments allows specified object points to be moved accurately on arbitrary trajectories. Controlling the motion of a point is not fundamentally different from nailing it in place, except that the "nail" is moving as a known function of time, giving rise to simple additive terms that take account of its motion. As with attachment constraints, the restriction to linear deformations ensures a constant constraint matrix.

Together, these elements give us the capability to create non-rigid pieces, wire them together in arbitrary ways, then control the motion of arbitrary points on the objects. A reasonably accurate view of the resulting models is to regard them as articulated puppets, whose parts are made of rubber, jello, or other non-rigid materials, with specified points under full control of the "puppeteer." Our

The work reported in this paper was support in part by Apple Computer.

method is fast enough to be practical: using widely available hardware, we can achieve interactive, even real-time performance with models that are complicated enough to be interesting subjects for animation.

This basic capability is only a starting point. Given the ability to control the motion of arbitrary points, how shall we specify where they should go? Interesting approaches include direct manual control and interactive keyframing. Our recent emphasis, however, has been on the development of a vocabulary of goal-directed behaviors that are chained to create complex actions. An example of a simple atomic behavior is one that smoothly moves a body point to a specified position and velocity over a specified time interval. A slightly more elaborate one makes a body point chase and capture a moving target point. Simple chaining permits the specification of compound actions such as "Make point a grab point b, move it to point c, then let go."

1.1 Background

A number of authors have investigated the use of articulated-body dynamics for animation [1, 17, 7, 9, 13]. Several of these employ highly efficient recursive dynamics formulations, but they are specific to articulated models composed of rigid bodies. The use of non-rigid dynamics for animation has been described in [15, 12, 16, 5, 10] The use of large finite-difference meshes, (or of mass-spring lattices, which are essentially equivalent,) poses a performance problem for two reasons: first, such systems possess many degrees of freedom, and second, they tend to lead to stiff equations which are expensive to solve. In [10], Pentland and Williams describe the use of modal analysis to create simplified dynamic models. By discarding high-frequency modes, the dimensionality and stiffness of the models are both drastically reduced. The models we develop here yield similar advantages, though we arrive there by a very different route.

The use of constraint methods for model creation and motion control has been extensively treated. [3, 18, 7, 9, 4, 12, 11, 20, 19, 13] A number of these entail the use of inverse dynamics to calculate constraint forces. The formulation employed here, which is based on the method of Lagrange multipliers, is described fully in [19], and is also closely related to that presented in [11].

2 Linear Deformations

2.1 The mechanics of global deformations

A global deformation [2, 14, 6] is a mathematical function that maps space to itself by assigning new, deformed coordinates to each point within some region of undeformed space. Applying a global deformation to an object entails deforming the space in which the object sits, allowing the points on the object's surface to be carried along. Any given deformation has associated with it some control parameters, such as bend or twist angles, that govern its behavior. Global deformations were originally introduced as a purely geometric modeling tool, allowing primitive shapes to be modified in stylized but interesting ways by manually adjusting the parameters.

Global deformations may also be made to serve as a substrate for simplified nonrigid dynamics by augmenting them in two respects: we embed masses in the space on which the deformation acts, and we define an energy of deformation that induces the desired behavior, such as elasticity or volume preservation. As with the simplified modal models of [10] the use of simple global deformations offers the twin advantages of reduced dimensionality and the elimination of the high-frequency components that lead to stiffness. In return for this large performance advantage, we naturally give up the ability to deform our objects in ways that cannot be represented using the chosen global deformation.

As an aid to understanding the derivation that follows, imagine a cloud of fixed point masses, all subjected to a global deformation. Although stationary in undeformed coordinates, the deformed points would be seen to move in response to a change of the deformation parameters. The addition of mass thus associates with any parameter displacement a *mass* displacement. This association allows us to express the body's inertial proprties, in particular its kinetic energy, as a function of the deformation parameters and their time derivatives. Once we have expressed the kinetic and potential energies as a function of the parameters, Lagrange's equations of motion(see [8] or any other classical mechanics text) provide a cookbook procedure for deriving the equations of motion, producing a generalization of the familiar $f = ma$. In the parlance of Lagrangian dynamics, the vector of parameters constitute the system's *generalized coordinates*, which we will denote by q.

To derive Lagrange's equations one must first express the kinetic energy T as a function of q and $\dot{q}$, and the potential energy V as a function of q. In terms of the *Lagrangian*, defined as $L = T - V$, Lagrange's equations are then

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} - Q = 0, \tag{1}$$

where Q is the force in q-space, known as *generalized force.*

The kinetic energy of a particle at position x with mass m is just $\frac{1}{2}m\dot{x}^2$, and the kinetic energy of a deformable body is the sum of its mass points' kinetic energy. For an

arbitrary deformation, the velocity of a particle is [1]

$$\dot{x}_i = J_{ij}\dot{q}_j,$$

where the Jacobian matrix J is defined by

$$J_{ij} = \frac{\partial x_i}{\partial q_j}.$$

If the mass at x is m, then the kinetic energy due to the point mass is

$$T = \frac{1}{2}m\dot{x}_i\dot{x}_i = \frac{1}{2}mJ_{ij}J_{ik}\dot{q}_j\dot{q}_k. \quad (2)$$

and the kinetic energy of the whole body is just the sum of this quantity over all the point masses. In the case of a continuous mass distribution, the sum becomes an integral and mass is replaced by mass density.

The form of the potential energy V depends on the desired behavior. For example, in a global bend deformation with bend angle θ, an energy term of the form $V = (\theta - \theta_r)^2$ would attract θ elastically to the rest value θ_r. The generalized force Q is typically due to point forces applied to the object. The generalized force due to a point force f applied at x is $Q = f_i J_{ij}$, with the Jacobian J evaluated at x.

2.2 Linear deformations

The preceding discussion applies to any deformation function. However, the Lagrangian equations of motion simplify greatly for deformations that are linear functions of the state. Such deformations may be expressed in the form

$$x_i = R_{ij}p_j,$$

where x is a world-space point, the components of matrix R are the generalized coordinates, and p is a function of the undeformed point, but not of R or of time. While deformations of this form are linear in the state R, x may depend nonlinearly on the undeformed coordinates through the function p. For example, p might be defined by $p(x, y, z) = \left[1, x, y, z, xy, xz, yz, x^2, y^2, z^2\right]$, which is 2nd order. In fact, any polynomial deformation may be cast in this form.

Since p is not a function of time, we then have point velocity

$$\dot{x}_i = \dot{R}_{ij}p_j$$

and kinetic energy

$$T = \frac{1}{2}\dot{R}_{ij}\dot{R}_{ik}M_{jk},$$

where M is a constant symmetric *mass matrix* defined by

$$M_{jk} = \sum \left(mp_jp_k\right),$$

with summation performed over all the mass points in the body.

For the sake of readability, we will omit the potential energy V, noting that the force due to V is $-(\partial V/\partial q)$, which may be subsumed in the generalized force Q. Therefore the Lagrangian is just $L = T$. To obtain Lagrange's equations, we observe that

$$\frac{\partial L}{\partial \dot{R}_{rs}} = \frac{1}{2}\left(\delta_{ir}\delta_{js}\dot{R}_{ik} + \delta_{ir}\delta_{ks}\dot{R}_{ij}\right)M_{jk}, \quad (3)$$

where the δ's are *Kroeneker deltas*, defined by $\delta_{ij} = 1$ if $i = j$, and zero otherwise; that is, the identity matrix. Using the identity $a_i\delta_{ij} = a_j$, and also the symmetry of M, we obtain

$$\frac{\partial L}{\partial \dot{R}_{rs}} = \dot{R}_{rk}M_{ks}.$$

It follows straightforwardly that

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{R}_{rs}}\right) = \ddot{R}_{rk}M_{ks}$$

and also that

$$\frac{\partial L}{\partial R_{rs}} = 0.$$

Combining these pieces, Lagrange's equations are

$$\ddot{R}_{ij}M_{jk} - Q_{ik} = 0, \quad (4)$$

and since M is constant, its inverse W may be precomputed, giving

$$\ddot{R}_{ij} = Q_{ik}W_{kj}. \quad (5)$$

Finally, the generalized force Q due to a force f applied at world-space point $x_i = R_{ij}p_j$ is

$$Q_{rs} = f_i\frac{\partial x_i}{\partial R_{rs}} = f_i\delta_{ir}\delta_{js}p_j = f_rp_s.$$

The inverse mass matrix W represents the linear function that maps forces into accelerations. The fact that W is constant affords some gain in efficiency in solving equation 5 because the matrix need not be inverted anew at each step. This is not a large advantage because the matrix is relatively small. The major benefit of a constant W will emerge later in the discussion of constraints. Figure 1 illustrates a second-order deformable object, with $p(x, y, z) = \left[1, x, y, z, xy, xz, yz, x^2, y^2, z^2\right]$, and with R a 3×10 matrix.

[1] In index notation, an unsubscripted quantity is a scalar, one subscript denotes a vector, and two denote a matrix. Under the *summation convention*, the appearance of any index twice in a term implies summation, so that $M_{ij}v_j$ means $\sum_j M_{ij}v_j$, which is matrix M times vector v.

2.3 Affine deformable bodies

A particularly simple linear deformation has

$$p_i(r_j) = [r_1, r_2, r_3, 1],$$

where r is a point in body coordinates, and

$$R_{ij} = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \\ t_1 & t_2 & t_3 \end{bmatrix},$$

which resembles a homogeneous transformation matrix with the rightmost column deleted. The 3×3 submatrix m is an ordinary 3D transformation matrix, and t is a translation vector. A body defined by subjecting mass points to this deformation is permitted to undergo affine transformations—translation, rotation, stretch, and shear, with equations of motion as given in equation 5. Despite the limited range of deformations they afford, affine deformable bodies offer two significant advantages: first, graphics workstations perform affine deformations very quickly; and second, the potential energy terms that yield elastic and volume preserving behavior become extremely simple.

2.4 Potential energy functions for affine bodies

An affine deformable body is in its undeformed state exactly when the submatrix m is an *orthogonal* matrix, defined by $m_{ij}m_{ik} = \delta_{jk}$. To see why, note that the squared magnitude of a transformed vector x_j is $m_{ij}m_{ik}x_jx_k$. This is equal to the squared magnitude of x for all x exactly when m is orthogonal. An energy function whose minimum lies at the undeformed state is

$$V_e = k_e \left| m_{ij}m_{ik} - \delta_{jk} \right|^2,$$

where k_e is a stiffness constant. This is a "rigidity" term, giving elastic behavior.

The affine deformation is volume preserving exactly when $\det(m) = 1$, so that an energy function that resists compression and dilation is $V_c = k_c \left| \det(m) - 1 \right|^2$, where k_c is also a stiffness constant. The imposition of this term gives a body that, when stretched along one dimension, reacts by squashing along the others, and *vice versa.*

The required gradients that give the forces due to both these energy terms are readily derived. Velocity dependent damping forces may be imposed on these and other potential functions using the form

$$Q_{ij} = -k_d \dot{V} \frac{\partial V}{\partial R_{ij}},$$

where k_d is a positive drag constant. Adjusting the stiffness and drag constants yields a wide range of physical behavior, from a bouncy jello-like response to near rigidity.

3 Constrained Dynamics

In [19] we present a general formulation for constrained dynamics, similar to that of [11], and more loosely related to [4]. Here we briefly summarize the general solution, then develop the simplified form for deformable bodies defined by linear deformations.

3.1 The general form

If vector q represents the state of a physical system, then a *holonomic constraint* may be defined implicitly by a function $c(q, t)$, where the states consistent with the constraint are those that satisfy $c(q, t) = 0$. For example, if a and b are points whose world-space coordinates depend on state, then $a(q) - b(q) = 0$ is a constraint that requires the points to coincide. If multiple constraints are to be met simultaneously, then c is a vector all of whose components must vanish. If the system begins in a legal state, with $c = 0$ and $\dot{c} = 0$, then requiring that $\ddot{c} = 0$ thereafter suffices in principle to hold the constraints in force. $\ddot{c}$ depends on the acceleration $\ddot{q}$, so requiring that $\ddot{c} = 0$ defines a subspace of legal accelerations. Because $\ddot{q}$ in turn depends on force, the problem of constrained dynamics is to calculate a *constraint force* that projects the acceleration into the legal subspace. Because it is expressed differentially, this turns out to be a linear problem even when the constraints depend nonlinearly on state.

To obtain the constraint equations we first express $\ddot{c}$ as a function of $\ddot{q}$. Applying the chain rule twice gives

$$\ddot{c}_i = \frac{\partial c_i}{\partial q_j}\ddot{q}_j + \frac{\partial \dot{c}_i}{\partial q_j}\dot{q}_j + \frac{\partial^2 c_i}{\partial t^2}, \tag{6}$$

noting that

$$\frac{\partial \dot{c}_i}{\partial q_j} = \frac{\partial^2 c_i}{\partial q_j \partial q_k}\dot{q}_k.$$

The term $\partial^2 c_i / \partial t^2$ reflects the direct dependence, if any, of c on time, in contrast to its indirect time dependence through q.

The equations of motion in turn give $\ddot{q}$ as a function of the known applied force Q and an as yet undetermined constraint force C, according to

$$\ddot{q}_j = W_{jk}(C_k + Q_k),$$

where W is an inverse mass matrix. Substituting into equation 6 and setting $\ddot{c}$ to zero gives

$$\frac{\partial c_i}{\partial q_j} W_{jk}(C_k + Q_k) + \frac{\partial \dot{c}_i}{\partial q_j}\dot{q}_j + \frac{d^2 c_i}{dt^2} = 0, \tag{7}$$

where only the constraint force C is unknown. Equation 7 gives a set of linear conditions that C must satisfy, but in general there are fewer equations than unknowns. This deficiency is rectified by requiring that the constraint force does not add or remove energy from the system, which leads to the requirement, known as the *principle of virtual work,* that the constraint force be a linear combination of the constraint gradients. This in turn means that C must satisfy

$$C_j = \lambda_i \frac{\partial c_i}{\partial q_j},$$

for some vector λ. The λ's are known as *Lagrange multipliers.* Substituting for C in equation 7, after some rearrangement, gives

$$-\left[\frac{\partial c_i}{\partial q_j} W_{jk} \frac{\partial c_r}{\partial q_k}\right] \lambda_r = \frac{\partial c_i}{\partial q_j} W_{jk} Q_k + \frac{\partial \dot{c}_i}{\partial q_j} \dot{q}_j + \frac{\partial^2 c_i}{\partial t^2}, \tag{8}$$

in which the matrix on the left hand side is square, with dimensions of the constraints, and only λ is unknown. The constraints are enforced by solving equation 8 for λ, using λ to compute C, adding C to the applied force, and computing the constrained acceleration $\ddot{q}$. In practice, an additional feedback term must be added to the force to inhibit drift, and also to bring the system to a legal state initially. Including the damped feedback term, the total force becomes

$$Q_j + (\lambda_i + \alpha c_i + \beta \dot{c}_i) \frac{\partial c_i}{\partial q_j},$$

where α and β are constants.

Equation 8 refers to the entire constrained system. In a system comprising a number of distinct objects, the global state vector q is formed by concatenating those of the original objects, and the constraint vector c is formed by concatenating the constraints. The inverse mass matrix W is block diagonal, receiving a block from each object. The constraint Jacobian receives a non-zero block for each constraint/object pair for which the constraint depends on the object.

Although we have written the constraints as direct functions of state, in practice there is usually an intermediate quantity to which constraints are applied. For example, a constraint that nails a pair of points together depends on the points, and the points' coordinates depend in turn on the respective objects' state. See [19] for a general partitioning scheme that exploits this kind of structure. In brief, if c is a constraint on one or more points, x is a point on which c depends, and q is the state of the object to which x is attached, then the chain rule gives

$$\frac{\partial c_i}{\partial q_j} = \frac{\partial c_i}{\partial x_k} \frac{\partial x_k}{\partial q_j},$$

as the Jacobian block representing c's dependence on q through the point x.

3.2 Impulses

When very large forces act for very short times, producing large transient accelerations, it is often useful to describe the behavior of the system in terms of the integral over the short time interval, neglecting the internal dynamics of the event, and to treat the duration of the event as zero. The force integral is known as an *impulse.* Instead of accelerations, impulses produce instantaneous changes in velocity. The calculation of an impulse response closely resembles that of an acceleration. The equation $\ddot{q}_i = w_{ij} Q_j$ becomes $\Delta \dot{q}_i = w_{ij} I_j$ where I is the impulse, and $\Delta \dot{q}$ is the change in velocity. In computing an impulse all non-impulsive forces, such as gravity, are neglected.

Impulses are most frequently encountered in the analysis of collisions. Here, we are interested in impulses in connection with motion control, where we may wish to allow the prescribed velocity of a controlled point to undergo a discontinuous change. The desired discontinuity appears in the direct derivative of the constraint with respect to time, leading to an equation similar to equation 8:

$$-\left[\frac{\partial c_i}{\partial q_j} W_{jk} \frac{\partial c_r}{\partial q_k}\right] \lambda_r = \Delta \frac{\partial c_i}{\partial t}, \tag{9}$$

where the right hand side gives the constraint discontinuities. Once λ is obtained, the constraint impulse is

$$I = \lambda_i \frac{\partial c_i}{\partial q_j}.$$

3.3 Linearly deformable bodies

Now we recast equation 8 for the special case of a system of linearly deformable bodies, subject to constraints each of which depends linearly on one or more points on the bodies. This restricted class of constraints includes point-to-point attachments, and constraints that nail points in place or require them to follow arbitrary known trajectories. This set is thus sufficient for building models by attaching deformable pieces together, and controlling the models by controlling the motions of specified points. The benefit of imposing this restriction is a large one: the constraint matrix on the left hand side of equation 8 remains constant, except when constraints are added or deleted. The matrix is inverted whenever the constraint structure changes, after which evaluating the constraint force requires only a matrix multiply rather than the solution of a linear system. Figure 2 illustrates the behavior of linearly deformable bodies subjected to attachment constraints.

If x is a point on a linearly deformable body then its derivative with respect to the body's state is

$$\frac{\partial x_i}{\partial R_{rs}} = \frac{\partial}{\partial R_{rs}} \left(R_{ij} p_j\right) = \delta_{ir} p_s,$$

which is a constant. Because each constraint is a linear function of one or more points, the derivative of any constraint c with respect to a point x is a constant as well. Hence the global constraint Jacobian is composed of constant blocks each having the form $\delta_{ir}p_s$, possibly times a constant. For example, a two-point attachment constraint of the form $a_i - b_i = 0$ yields two such blocks, one positive and the other negative.

Because x is a vector and R is a matrix, the derivative has rank 3. In practice, though, the R's are flattened and concatenated to form the global state, so that in an expression like $\delta_{ir}p_s$, the r and s are combined to form a linearized state index, and the quantity may then be viewed as a block in the global matrix. The bookkeeping involved in performing these index calculations is greatly simplified by the use of a block-sparse matrix data structure, in which a matrix is composed of a collection of rectangular blocks. Operations such as matrix-times-vector and matrix-times-matrix are readily implemented in terms of this structure.

Once the constant Jacobian matrix has been computed, the left-hand-side matrix of equation 8 can be calculated and inverted. Because $\dot{c}$ does not depend on q, one term of the equation vanishes, giving

$$\lambda_r = -Y_{ri}\left(\frac{\partial c_i}{\partial q_j}W_{jk}Q_k + \frac{\partial^2 c_i}{\partial t^2}\right), \qquad (10)$$

where Y is the inverted constraint matrix.

4 Motion Control

The preceeding sections provide the machinery required to animate a collection of connected objects—a puppet, for example—by moving control points on the puppet as arbitrary functions of time. As the control points follow their assigned paths, the rest of the puppet moves with correct passive dynamics. We begin this section by considering the generic problem of constraining a point to follow a known trajectory. Given this capability, we proceed to treat the issue of generating motion paths both by interactive keyframing and by the specification of motion goals.

A constraint that nails a point p at a fixed location n may be written $R_{ij}p_j - n_i = 0$. Such a nail constraint depends on time only indirectly, through R, so that the constraint's contribution to the direct time derivative term of equation 10 is zero. Suppose that the nail position n is not constant, but is instead a known, twice-differentiable function of time, $n(t)$.[2] By saying that $n(t)$ is known, we mean that we have a way to evaluate n, $\dot{n}$, and $\ddot{n}$ at the current time. We need not know anything further about the form of n or the manner in which it is computed. The control constraint then becomes

$$R_{ij}p_j - n_i(t) = 0,$$

with direct second time derivative

$$\frac{\partial^2 c}{\partial t^2} = -\frac{\partial^2 n}{\partial t^2}.$$

Inserting this term into equation 10 induces a constraint force that causes point p to move with the desired acceleration. The feedback term

$$(\alpha c_i + \beta \dot{c}_i)\frac{\partial c_i}{\partial q_j}$$

inhibits drift from the desired trajectory. Note that

$$\dot{c}_i = \dot{R}_{ij}p_j - \frac{\partial n_i}{\partial t},$$

so that the feedback term as well as the constraint force take account of the desired motion.

Depending on the acceleration supplied at each instant in time, the control point can be made to accurately follow any piecewise twice-differentiable trajectory. This form of control is analogous to attaching a jet engine at the control point and continuously adjusting its thrust to drive the point along the desired path. Note that it is not necessary that the path be completely specified in advance—only values at the current time need be known.

Having the general ability to control the motion of points on an object makes it possible to separate the problem of motion specification from that of enforcing the specified motion. We now consider two approaches to motion specification: keyframing, and goal-directed motion.

4.1 Keyframed Motion Paths

A direct extension of standard animation techniques is to specify control point trajectories by interpolating between keyframes. If a user is allowed to interactively position control points on arbitrary frames, piecewise cubic splines passing through the keyframed points suffice to provide the required values for n, $\dot{n}$, and $\ddot{n}$ at each instant in time. Provided that position and velocity are matched at the beginning of a keyframed motion, our models are able to track the keyframed paths accurately and stably at interactive speeds.

Experiments quickly showed us that physical keyframe control differs fundamentally from standard direct control of object parameters. First, the quantities that we are able to control are more likely to be the ones that we *want* to control. Second, we are permitted to employ far fewer degrees of control than degrees of freedom, the rest of the motion being determined by physics. This is a property

[2] We may relax this requirement to piecewise differentiability by inserting impulses at velocity discontinuities.

with no counterpart in conventional keyframing, where nothing moves unless we move it.

To fully exploit the ability to *refrain* from controlling all aspects of the motion, control points ought not to be regarded as persistent entities whose keyframed trajectories span an entire scene. The style of keyframe control that we believe will be most effective is based on the ability to freely turn control points on and off during an animation, establishing and relinquishing control as required. For example, in animating a walk it is necessary to control heel, toe, knees, hips, shoulders, etc., but not all at the same time. The heel position must be accurately controlled just before and during the support phase, but during the swing the heel can simply be allowed to follow the toe. It would be an unnecessary burden to specify the heel's position all the time.

The ability to turn control points on and off raises technical issues. Turning a path constraint off is simply a matter of letting the point "go ballistic," eliminating the relevant blocks from the constraint matrix, and turning off the restoring forces. Turning a path constraint on in the midst of an ongoing motion is more difficult: at the moment that control is initiated, the position and velocity of the point being controlled must match those specified by the splined trajectory. However, the point's state cannot generally be predicted in advance. We handle this problem using what we call *constraint preroll*, by analogy to the video term. A short time before the nominal onset of control we dynamically compute a spline segment that smoothly joins the point's current position and velocity to those at the start of the pre-specified path. During the preroll interval, this segment serves to bring the point smoothly from its uncontrolled state to the required initial state.

Impulses provide additional keyframing possibilities by allowing control points to undergo arbitrary velocity discontinuities. For example, at the end of a motion path, it is possible to insert an impulse that makes the control point stop dead or "bounce," simulating collisions. Although not generally physical, starting a motion impulsively may also produce interesting effects. We also use impulses as a graceful way to start and stop animation runs. Before starting the run, we use an impulse to install the initial control point velocities, and at the end, an impulse is used to bring them to an instant but well behaved halt.

4.2 Goal-Directed Motion

Keyframing of point trajectories, though offering real advantages over object-parameter keyframing, can still be a frustrating process, primarily because motion must be specified in such a literal way. For instance, the fact that a reaching motion is intended to bring the hand into contact with an object to be grasped is entirely lost in the translation to spline curves. If the object's position is changed, the hand will happily grab the empty space where it used to be, unless the hand's motion is manually changed as well.

An alternative to keyframing is to specify the goals of actions directly, dynamically calculating the motion required to satisfy them. In contrast to the first-principles approach to motion synthesis described in [20], our objective here is to develop a minimal vocabulary of simple behaviors that are chained to produce motion. With the ability to control the motions of individual object points already in hand, it is comparatively simple to develop a variety of useful atomic actions. To implement an action that governs the behavior of one or more points, we must provide a way to compute the desired positions, velocities, and accelerations, as a function of state and of time.

An example will illustrate the approach. Suppose we want to make one point chase another, making contact at a specified time, and with specified final velocity. If the target point is stationary, then a cubic spline segment can be constructed, taking the chaser's current position and velocity as initial conditions, and the position and velocity of the target point as final conditions. As the motion progresses, the spline and its derivatives are evaluated to supply the desired position, velocity, and acceleration. This is equivalent to the "preroll" segment described earlier. To chase a moving target, we use its current position and velocity to make a linear prediction of its position at the desired time of contact, build a spline to that point, and continuously update the estimate as things change. The action is complete when the appointed contact time is reached.

Figure 3 shows some frames of a very simple animation in which a sky hook swoops down, grabs a pyramid, flys off with it, then hangs itself up, leaving the pyramid dangling. It was created using two primitive behaviors, "Chase," as described above, and "Connect," which applies a velocity-matching impulse, then creates an attachment constraint. For convenience, we define a behavior "Grab" which performs a Chase followed by a Connect. Having first created the geometry and defined some named control points at strategic locations, we used the following script to produce the motion:

```
Grab(hook_tip,pyr_tip, dt1, vx1, vy1)
Grab(hook_eye,sky_point, dt2, vx2, vy2)
```

where the dt's and v's are durations and final velocities for the chase segments.

Figure 4 shows a more complex sequence in which a hinge-horned monster skewers and ingests a small humanoid. The fine grained action is as follows: the monster waits until a humanoid comes in range, skewers it, moves its "elbow" to a suitable spot, and accelerates the humanoid to a point just outside its mouth. At just that moment, the monster lets go, simultaneously stopping its claw, and the

humanoid pops neatly into the gaping jaws. Here is the script, including interactively chosen values for timing and velocity:

```
Wait_For_Humanoid()
Grab(claw_tip,humanoid, 1.0)
Grab(elbow,feeding_position, 1.0, 0, 0)
Grab(claw_tip,mouth_point, 1.0, -1.5,
   -1.5)
Disconnect(claw_tip,humanoid)
Grab(humanoid,gullett, .5, -.07, -.07)
```

When no final velocity is specified, "Grab" computes a default based on the current position and velocity, duration and distance to the target. The predicate "Wait_For_Humanoid" serves to trigger the action when suitable prey comes within range. Unlike a conventional animation script, this sequence defines what amounts to a reflex. The monster responds to its environment, successfully capturing its prey over a wide range of initial conditions.

References

[1] William W. Armstrong and Mark W. Green. The dynamics of articulated rigid bodies for purposes of animation. In *Visual Computer*, pages 231–240. Springer-Verlag, 1985.

[2] Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18:21–29, 1984. Proc. SIGGRAPH 1984.

[3] Ronen Barzel and Alan H. Barr. *Topics in Physically Based Modeling, Course Notes*, volume 16, chapter Dynamic Constraints. SIGGRAPH, 1987.

[4] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constaints. *Computer Graphics*, 22:179–188, 1988.

[5] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):243–252, 1989. Proc. SIGGRAPH 1989.

[6] Kurt Fleischer and Andrew Witkin. A modeling testbed. In *Proc .Graphics Interface*, pages 127–137, 1988.

[7] Michael Girard and Anthony A. Maciejewski. Computational Modeling for the Computer Animation of Legged Figures. *Proc. SIGGRAPH*, pages 263–270, 1985.

[8] Herbert Goldstein. *Classical Mechanics*. Addision Wesley, Reading, MA, 1950.

[9] Paul Issacs and Michael Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987. Proc. SIGGRAPH '87.

[10] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, 1989. Proc. SIGGRAPH 1989.

[11] John Platt. *Constraint Methods for Neural Networks and Computer Graphics*. PhD thesis, Caltech, 1989.

[12] John Platt and Alan Barr. Constraint methods for flexible models. *Computer Graphics*, 22:279–288, 1988.

[13] Peter Schroeder and David Zeltzer. Dynamic simulation with linear recurive constraint propogation. *Computer Graphics*, 24(2):23–32, March 1990.

[14] Thomas Sederberg and Scott Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986. Proc. SIGGRAPH 1986.

[15] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4), July 1987. Proc. SIGGRAPH '87.

[16] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. In *Proc. Graphics Interface*, pages 146–154, Edmonton, Alberta, Canada, June 1988.

[17] Jane Wilhelms and Brian Barsky. Using dynamic analysis to animate articulated bodies such as humans and robots. *Graphics Interface*, 1985.

[18] Andrew Witkin, Kurt Fleischer, and Alan Barr. Energy constraints on parameterized models. *Computer Graphics*, 21(4):225–232, July 1987. Proc. SIGGRAPH '87.

[19] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *Computer Graphics*, 24(2):11–22, March 1990.

[20] Andrew Witkin and Michael Kass. Spacetime constraints. *Computer Graphics*, 22:159–168, 1988.

Figure 1: A three-dimensional second-order deformable object. The object is shown above in its undeformed state. Below it is shown deforming in response to a leftward pull, while its right-hand corners are held in place. This object can bend, whereas an affine body may only stretch and shear.

Figure 2: A two-dimensional triple pendulum moving under the influence of gravity. The pendulum, which is nailed in place at the top, is composed of affine-deformable parts connected using attachment constraints. The frames for all sequences are ordered top-to-bottom, then left-to-right.

Figure 3: Animation created by chaining motion goals. A sky hook swoops down to grab a pyrmid, then hangs itself up.

Figure 4: A hinge-horned monster skewers and ingests a small humanoid. This compound behavior is in effect a reflex, triggered by the arrival of suitable prey, which executes successfully over a wide range of initial conditions.