Using Kinematic Clones to Control the Dynamic Simulation of Articulated Figures

Bill Westenhofer* and James K. Hahn†

*Rhythm & Hues Studios
Los Angeles, CA 90066
bill@rhythm.com

†The George Washington University
Department of Electrical Engineering and Computer Science
Washington, DC 20052
hahn@seas.gwu.edu

Abstract

A new paradigm is presented for the control of dynamic simulations involving articulated figures. A clone of a figure is manipulated by an animator using traditional kinematic control techniques. The realized dynamic simulation is influenced by this animation through dampened spring forces that connect each link to its corresponding link in the clone. Varying tensions dictate the tightness of the correlation between the animated actions and the computed movements of the simulation. This paradigm strikes a compromise between the absolute control of kinematics and the realistic automation of dynamics. Such a compromise is often sought by animators who want realistic motion but do not want to lose the ability to impart feeling and emotion in a character.

1. Introduction

Over the past century, classical animators have striven to endow characters with "life." Their challenge was to make a set of sequential drawings become a personality that an audience could identify with. "Character animation," as it has been so named, is the process of giving motion to a person, creature, or even inanimate object, in such a way that it appears to be alive, complete with autonomous control and evidence of thought and emotion. Animators at groundbreaking studios like Disney and Warner Brothers learned early on that such a lifelike quality could be reliably achieved by following a set of principles and guidelines [1]. Many of these principles like *proper staging*, *exaggeration*, and *appeal*, are artistic metrics that are important for the unambiguous portrayal of the intended actions of a character. Others, however, like *squash-and-stretch*, *follow-through*, and *overlapping action* are really formalizations of phenomena that occur naturally as a result of gravity, inertia, and the deformation of viscous material. Such physical behavior is well understood and capable of being calculated which is why it is tempting, in the light of the relatively recent use of the computer as an animation medium, to let a dynamics simulation handle their incorporation into character animation. Current techniques for doing 3D computer animation are often difficult and time consuming, which makes any such potentially time saving technique even more valuable. As computer graphics become more and more proliferated into movies, games, and advertising, the producers of such products must continually search for ways to cut production costs and timelines. At the same time, they must increase levels of quality and realism to suit the demands of an ever more sophisticated audience.

Animations of simple objects that have employed dynamic simulations have produced incredibly realistic motion of both rigid and deformable bodies [2][3]. The dynamic behaviors of motion for such objects have been known for some time. The use of dynamics in computer applications is not new either; robotics, industrial design, and even some advanced CAD/CAM systems have employed rigid body simulations. Computer graphics and animation applications using dynamic simulations have also been presented, and some of them, such as particle systems [4], have become widely used in the entertainment industry. There have even been methods suggested for introducing soft tissue motion behavior into computer generated creatures [5]. In general, however, most professional animation production, especially character

0-8186-7518-7/96 $5.00 © 1996 IEEE

animation, has yet to realize a benefit from dynamic simulations. The problem with their use lies in the difficulty of balancing the benefits of automation with an appropriate amount of control. In entertainment applications, the motion of a character, no matter how interesting, is not the end, but rather the means of telling a story. Professional animation therefore, tends to be less an act of discovery, and more the result of planning, execution, and continual refinement. A basic dynamic simulation, on the other hand, accepts a set of initial conditions and iterates forward, offering little opportunity for refinement along the way. An animator's control in this case is limited by his or her ability to predict the outcome from the supplied initial conditions. Dynamic motion is usually complex and difficult to predict which makes controlling it in this way impractical. To be useful, a system that incorporates dynamics has to give an animator the control during the course of a simulation to define the elements of an animation that are considered essential. The simulation's task is then to "fill in the gaps" with realistic motion.

This paper focuses on the control of dynamics simulations of *articulated figures* which are the most common construction for 3D animatable characters. An articulated figure is a hierarchical arrangement of rigid parts connected to each other by joints with varying degrees of freedom (DOF). It is not uncommon for a character represented by an articulated figure to have as many as 60 DOFs. An animator is ultimately responsible for the motion of each of these DOFs throughout an animation. The most common control technique used in studios today is "keyframing." With this technique, an animator defines "key" positions for each DOF which are then interpolated by the computer to create the final motion paths. In some ways, keyframing's biggest drawback is also its greatest strength. Keyframing is advantageous in that it grants total control to specify a motion. Such control is essential when an animator is establishing the subtle postures and timings that are needed to communicate things like a character's emotional state. The disadvantage of keyframing is that with this control comes the additional responsibility of accounting for the entire motion, including the more mechanical processes that a dynamics simulation would be able to compute. This price is paid through extra keyframes and more iterations to get the feel of motion right. Our technique seeks to take the "best of both worlds" by providing kinematic control where an animator needs it and a dynamics simulation to realistically calculate the rest.

2. Related Work

Considering the potential applications for physically-based simulations of articulated figures, its not surprising that there have been a wide range of strategies presented for their control. Strategies differ primarily in the nature in which control is granted to (and expected from) an animator.

Both McKenna and Zeltzer [6] and Raibert and Hodgins [7] developed motor control programs for simulated actuators in their models that imitate the muscular activity used during walk and run cycles in real animals. An animator's task in this case is to write programs that drive the actuators with the appropriate forces at the appropriate times. Several researchers [8][9][10][11][12][13] have taken this technique a step farther by creating systems that automatically generate the motor control programs based on a set of user criteria. The user criteria is given in the form of objective functions that are used to evaluate the relative merit of the generated control programs. The techniques vary in terms of the nature by which the control programs accept input and effect motion, and in the way in which new control programs are generated. They are all similar, however, in that they require objective functions that can be formulated into equations. Such a task is relatively easy for functions like "move from point A to point B in the shortest time," or "jump over obstacle C using the least amount of energy." It is far more difficult, however, to write a function that evaluates "happiness" or a "stately walk." Until such time that artificial intelligence allows us to program personalities and emotions, we feel that objective functions cannot be made descriptive enough to match the capabilities of a trained animator. For this reason, we feel that kinematic control is essential.

Interestingly, it was some of the earlier papers on the control of dynamics simulations that presented kinematic strategies. Hahn [2] described how a system could calculate the forces and torques that a set of moving links would exhibit on a figure as a whole. The links themselves would be animated through kinematic means, but their motions would impart torque on the body much like a diver can impart a change in spin rate in the middle of a dive.

Isaacs and Cohen's [14] DYNAMO system allowed a subset of the links in a figure to be controlled kinematically while the rest were calculated in a dynamics simulation. Their system built a set of simultaneous dynamics equations and factored out the known accelerations of the kinematic links. The dynamically controlled links would react to external forces and to the motions of the kinematically controlled links. The limitation with their system is that links could not be dynamically controlled and still respond to some sort of kinematic influence.

Armstrong et. al. [15], and Wilhelms [16] presented systems that are similar in philosophy to the one presented here. All of the links of the articulated figures in their systems were under control of the dynamics simulation, but an animator could constrain the motion through kinematic means. Individual links

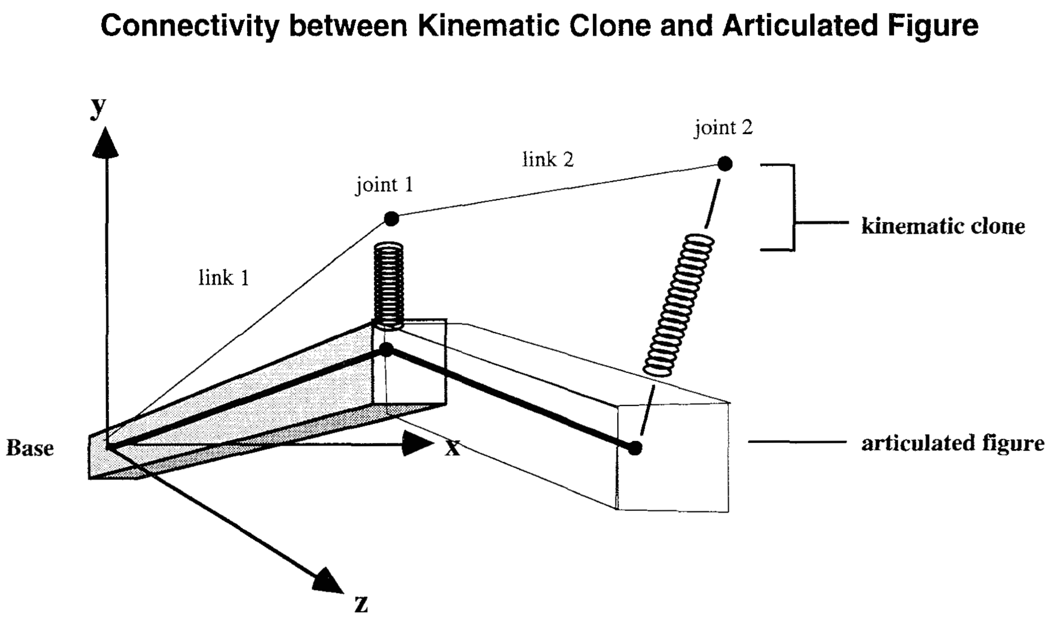

figure 1

could be assigned one of four control strategies. Links could be 1) free, meaning that no kinematic control would be applied, 2) fixed in space using constraint forces, 3) fixed in relation to a parent joint, or 4) forced to match the position and orientation of a kinematic motion. Their methods come close to providing the control we desire. The fault in their systems is that they generate forces that work to exactly match the kinematically defined motions. Our goal is to use dynamics to enhance kinematically created motion with realistic effects. To exactly match it would defeat our purpose.

3. Kinematic Clones

We chose to base our framework for kinematic control on keyframing systems due to their popularity. In order to most effectively combine dynamics with this type of system, there are some important considerations for us to make regarding the manner in which we integrate the simulation. First, it would be beneficial if the model that an animator manipulates in the keyframe system is similar in construction to the one that will be run through the simulation. Similar construction makes it easier to relate poses specified kinematically with those output from the dynamics simulation. Second, the nature in which the dynamics simulation differs from the purely kinematic motion should be intuitive; an animator should be able to have some sense of what the motion will be before a simulation is run. As was suggested earlier, surprises are rarely a welcome thing in a production environment. Third, the influence of the dynamic simulation should be refineable. Often, one has a motion that is close to what is desired, but that needs minor adjustments to be perfect. Small changes to the dynamic parameters should result in small changes to the final motion. Finally, the system must be flexible. An animator should have some control over how closely the dynamic motion matches the keyframed motion. He or she should also have the ability to exempt certain parts of a figure from the simulation so that a critically timed action is guaranteed to stay intact.

In keeping with these criteria we introduce the concept of a *kinematic clone* (the name betrays a slight favoritism on our part; the "clone" could just as easily have been the dynamically controlled figure). A kinematic clone is a link-for-link duplicate of the articulated figure that will be run through a dynamics simulation (see fig. 1). An animator manipulates the kinematic clone directly to position the keyframes necessary for a particular action. Unlike traditional keyframe planning, however, dynamic effects can be ignored since they will be introduced by the simulation. Through this construction we are able to utilize existing keyframe systems directly and are able to take advantage of their installed kinematic time saving techniques like inverse kinematics and motion capture.

In order to allow the keyframed motion of a kinematic clone to effect the motion of the articulated figure, we introduce virtual spring connections between corresponding joints. Spring behavior is intuitive which makes it easy for an animator to predict the kind of dynamic motion that will be realized in the simulation. Spring forces are also simple to calculate which results in easier coding and fast response times. Flexibility is provided by allowing an animator to adjust the tightness of the springs throughout the figure and over time. Certain connections may need to be tighter than others; a wrist needs to reach its target more accurately than an elbow, for example. Varying tightness over time can be useful for portraying periods of relaxation.

One might observe that by adding adjustable spring parameters, we have increased the number of DOFs in a figure. We feel this cost is justified by a reduction in keyframes and through the benefit of having complex motion generated automatically. Additionally, these new DOFs will probably be modified less often than those for position and orientation.

4. Dynamics Formulation

Robotics research provides several methods for calculating the dynamics of articulated figures that have been optimized over those used for simple rigid bodies. We chose Roy Featherstone's Articulated Body Method (ABM) [17] based on its reputation for being a very fast algorithm for figures with large numbers of DOFs [6]. In general, dynamics algorithms for articulated figures fall into two categories: 1) those that involve a set of simultaneous equations, and 2) those that use recursive formulations. Simultaneous equation solvers, like the Walker-Orin method typically have $O(n^3)$ time complexity (or worse) but are very efficient for small numbers of DOFs. Most industrial robots have six DOFs or less which is why such algorithms are popular for designing robotic motor control programs [18]. The ABM is a recursive method which proves itself to be more efficient than the Walker-Orin method with 9 DOFs or more. The ABM is similar to Armstrong's approach [15], but is superior in that it is not limited to spherical joints.

4.1 Spatial Algebra

Featherstone's work relies heavily on a mathematical system called *spatial algebra.* Spatial algebra uses six-dimensional *spatial vectors* that simultaneously account for the linear and angular components of position, velocity, and acceleration. Most dynamics formulations handle linear and angular quantities separately using two sets of equations. Spatial algebra eliminates this restriction, thus halving the number of equations required. Spatial vectors are given a common basis by transforming the linear velocity of a body from its center of mass to the origin of the current coordinate system. That transformation is:

$$\hat{r} = \dot{r} + r \times \omega \tag{1}$$

(the "^" denotes spatial quantities). Using this convention, we can define *spatial velocity* and *spatial acceleration* as:

$$\hat{v} = \begin{bmatrix} \omega \\ \dot{r} + r \times \omega \end{bmatrix} \tag{2}$$

$$\hat{a} = \begin{bmatrix} \dot{\omega} \\ \ddot{r} + \dot{r} \times \omega + r \times \dot{\omega} \end{bmatrix} \tag{3}$$

Spatial rigid body inertia is:

$$\hat{I} = \begin{bmatrix} m\vec{r}\times & m1 \\ I^* + \vec{r} \times m\ (-\vec{r}\,)\times & \vec{r} \times m \end{bmatrix} \tag{4}$$

I^* is regular rigid body inertia about center of mass

$\vec{r}$ is the vector from the origin to center of mass

The cross operator, $\times$, is defined as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{5}$$

Another important construction, which is used to provide joints with varying DOFs, is a *vector sub-space.* This is a mapping from a one space to another with equal or more dimensions. We use sub-space matrices to map joint parameters to the linear and angular components of a joint's spatial velocity.

Finally, there are two important spatial functions that must be defined in order to understand the upcoming dynamics equations. A *spatial transpose,* denoted a^S is defined as:

$$\hat{a}^S = \begin{bmatrix} a \\ b \end{bmatrix}^S = [\, b^T \, a^T \,] \quad \text{and,} \tag{6}$$

$$\hat{A}^S = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^S = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix} \tag{7}$$

The other is the *spatial cross operator*:

$$\begin{bmatrix} a \\ b \end{bmatrix} \hat{\times} = \begin{bmatrix} a\times & 0 \\ b\times & b\times \end{bmatrix} \tag{8}$$

4.2 The Articulated Body Method

To establish a recurrence relation, a quantity known as an *Articulated Body Inertia* (ABI) is established. An ABI is the apparent instantaneous inertia of a body comprised of a set of flexible joints. Once the ABI of an articulated figure is known, the entire figure can be treated as a single rigid body with a rigid body inertia equal to the ABI and with a fixed point of rotation about the first joint of the figure. Assuming that an articulated figure has a non-looping hierarchical structure, we can recursively define any branch or sub-branch as a single rigid body with an ABI built from all its constituent links. The Articulated Body Method works by finding the ABI of each branch and sub-branch of a figure and uses them to propagate accelerations from the base to all the links.

4.2.1 Calculating Articulated Body Inertias. The process of finding all of the ABIs for each sub-tree requires a recursive pre-step that starts from the root node and propagates out to the leaves. The purpose of this step is to propagate the velocity of the base combined with the cumulative velocities of each of the interior joints outwards in order to calculate the total momentum of all the links. This equation for the velocity of a link *i* (expressed in spatial coordinates) is:

$$\hat{v}_i = \hat{v}_{i-1} + \hat{s}_i \dot{q}_i \tag{9}$$

$\hat{s}_i$ is the subspace matrix of joint i

Both the ABIs and the momenta of the sub-trees are calculated on a return loop back from the leaves. The ABI of a leaf node is simply the rigid body inertia of that link. From there, the following equations are employed for the ABI and momenta:

$$\hat{I}_i^A = \hat{I}_i + \hat{I}_{i+1}^A - \frac{\hat{I}_{i+1}^A \hat{s}_{i+1} \hat{s}_{i+1}^S \hat{I}_{i+1}^A}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \tag{10}$$

$$\hat{p}_i = \hat{v}_i \hat{\times} \hat{I}_i \hat{v}_i + \hat{p}_{i+1} + \hat{I}_{i+1}^A \hat{v}_{i+1} \hat{\times} \hat{s}_{i+1} \dot{q}_{i+1} + \dots$$

$$\frac{\hat{I}_{i+1}^A \hat{s}_{i+1} (Q_{i+1} - \hat{s}_{i+1}^S (\hat{I}_{i+1}^A \hat{v}_{i+1} \hat{\times} \hat{s}_{i+1} \dot{q}_{i+1} + \hat{p}_{i+1}))}{\hat{s}_{i+1}^S \hat{I}_{i+1}^A \hat{s}_{i+1}} \tag{11}$$

where Q_{i+1} is the active joint forces of the next link. $\hat{v}_i \hat{\times} \hat{I}_i \hat{v}_i$ is the bias force on link i due to velocity product forces.

4.2.2 Propagating Accelerations. Once the ABIs and momenta are known, the spatial accelerations and resultant joint accelerations can be found. This last step propagates back out from the base to the leaves. These equations are respectively:

$$\hat{a}_i = \hat{a}_{i-1} + \hat{v}_i \hat{\times} \hat{s}_i \dot{q}_i + \hat{s}_i \ddot{q}_i \tag{12}$$

$$\ddot{q}_i = \frac{Q_i - \hat{s}_i^S (\hat{I}_i^A (\hat{a}_{i-1} + \hat{v}_i \hat{\times} \hat{s}_i \dot{q}_i) + \hat{p}_i)}{\hat{s}_i^S \hat{I}_i^A \hat{s}_i} \tag{13}$$

4.3 Algorithm Implementation

Our current implementation uses all spherical joints. The ABM method, through sub-space matrices, allows for joints of varying DOFs, but we decided to use spherical joints so that we could test the spring connections' performance with the widest degree of latitude possible. The use of sub-matrices also makes it easy to use a number of joint space parameterizations. We store the spherical orientation of individual joints using quaternions which are free of the singularity and associated numerical instability problems of Euler angles. With quaternions, we can also use the angular velocity vectors themselves to parameterize joint velocity. By also expressing all of our quantities in joint local coordinate systems, we get a very simple sub-space matrix:

$$\hat{S} = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \\ 0\,0\,0 \\ 0\,0\,0 \\ 0\,0\,0 \end{bmatrix} \tag{14}$$

This makes sense since the spatial velocity vector at a joint reduces to the angular velocity vector for spherical joints. The relationship between angular velocity and the change in the positional quaternion is calculated using:

$$\dot{q}(t) = \frac{1}{2}\,\omega(t) q(t) \tag{15}$$

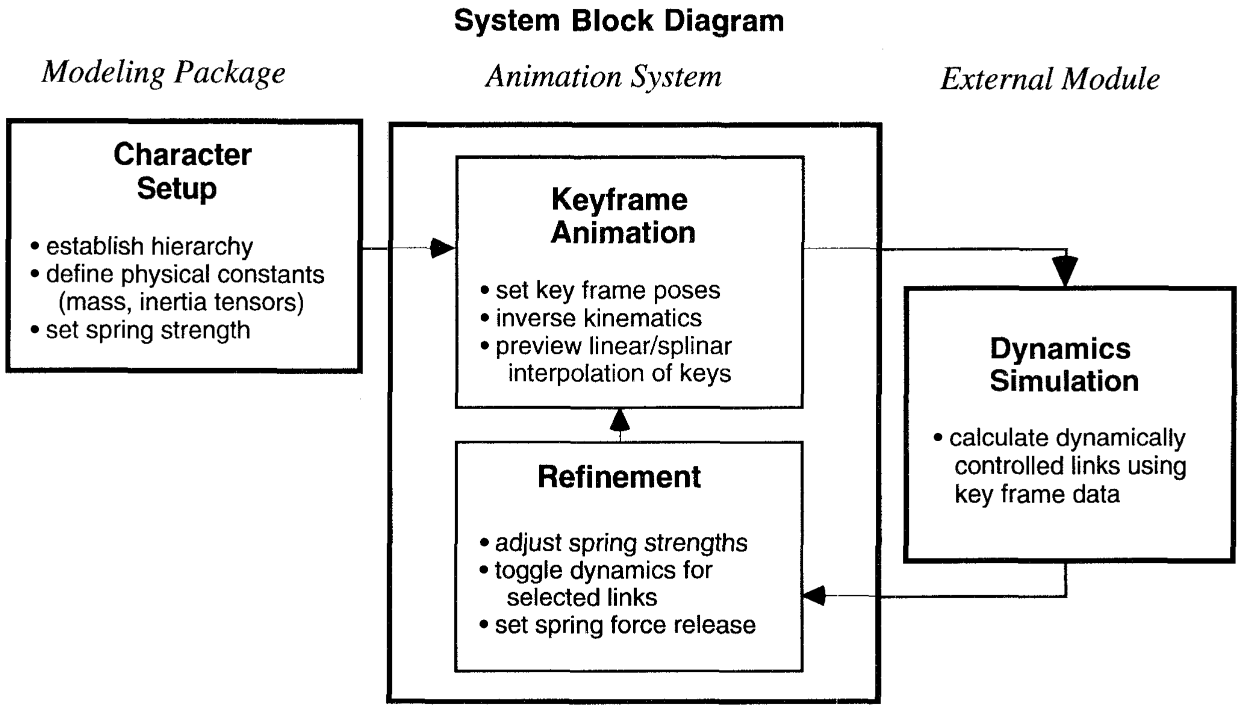
System Block Diagram
Modeling Package
Animation System
External Module
Character Setup
• establish hierarchy
• define physical constants (mass, inertia tensors)
• set spring strength
Keyframe Animation
• set key frame poses
• inverse kinematics
• preview linear/splinar interpolation of keys
Refinement
• adjust spring strengths
• toggle dynamics for selected links
• set spring force release
Dynamics Simulation
• calculate dynamically controlled links using key frame data

figure 2

Fortunately, the animation system we use already stores angular positions with quaternions which makes our setup even easier.

4.4 Numerical Integrations

Finally, to numerically integrate the dynamics equations, a fifth order Runge-Kutta method is used with adaptive step-size. A simpler, Euler-step method was tested but proved to be very unstable, as one would expect. The particular Runge-Kutta method used is an embedded Runge-Kutta formula originally invented by Fehlberg [18], which has error estimation built into the function evaluation. The formulation is fifth-order with an error estimation that bounds the fourth order calculation. The extra accuracy makes up for its cost by allowing larger step sizes. It proved to be quite stable in testing.

5. System Implementation

5.1 Architecture

The system we have implemented follows the block diagram depicted in figure 2. Both the modeling package and the animation system were existing tools that were modified only to accept and process the new parameters for the dynamics simulation.

5.1.1 Character Setup. As part of the normal setup process, an animator plans the articulation of a character in the modeling/setup package. The number of joints are determined and a hierarchy is established. Added to the modeling package is the ability to set masses for individual links and provide an inertia tensor as a nine-dimensional array. The inertia tensors are calculated in link local coordinates, so we can generally reduce the computation to the three diagonal elements. Also new to the package are spring strength values for each link that determine how closely a dynamically controlled link's base joint will follow the corresponding kinematic position specified in the keyframes. Finally, an animator can, at this stage, limit the links which are to be effected by the simulation. Decisions to exclude links can be made to limit computation and/or to insure a precise performance from certain elements like a hand catching a ball.

5.1.2 Keyframe Animation. The keyframe system remains virtually untouched with the incorporation of the dynamics system. The animator directly animates the kinematic clone of a figure and sets key frames as appropriate to portray a character's personality and actions.

5.1.3 Dynamics Simulation. Prior to running this module, the keyframes are interpolated to produce a frame by frame kinematic position and orientation for each link that is used as a reference to measure spring offsets during the course of the simulation. The simulation is run and the position of the articulated figure is found for each frame based on external forces like gravity, wind, and the joint-to-joint spring forces, and on internal forces from the articulated body inertias. The final positions are stored and passed back to the animation package for refinement.

5.1.4 Refinement. After the dynamics simulation is run, the positions of the dynamic figure are displayed along with the kinematic clone. At this point, the animator can adjust spring strengths for individual links, potentially varying them over time if he or she wants to protect a successful earlier section, for example. We also add an interesting visualization technique for cases where initial key positions need to be altered. When an animator adjusts a key frame on the kinematic clone, the identical transformations are carried out on the dynamic figure. There is certainly no guarantee that the dynamic figure will assume this new pose when the simulation is rerun, but it provides a ballpark guess as to where the link might be, which, as we said earlier, is a desirable quality.

5.2 Spring Connections

As mentioned earlier, the connection between the kinematic clone and its dynamic counterpart is made through spring connections that join the corresponding joints of the two structures. For control purposes, the components of the spring force between a pair of joints can be broken out along the coordinate axes of the dynamically controlled link on the posterior side of the joint. This allows an animator to specify different spring coefficients along certain axes to impart directional bias. Such a setup can be used to simulate the joint of a hip which has more resistance from side to side than from front to back.

The spring forces obey an exponential version of Hooke's Law using the following formula:

$$f = \beta(k\,|\,d\,|^{\alpha})m \qquad (16)$$

k *spring constant specified by animator*
d *spring displacement*
m *mass of link*

The two constants α and β determine the exponential behavior of the springs. α determines how steeply the force rises with respect to distance while β determines the width of a "valley" about zero where exponential springs exhibit less force than linear ones. Exponential springs were used exclusively in test cases and exhibit better numerical stability than the alternative of stiffer spring constants and linear response.

Since the spring forces connect corresponding joints, we must make a special case for the last link in a chain. We add an additional null joint at the end of the last link and provide it with its own variable spring constant. We began with a configuration where the spring constant of the last joint was reused for this extra connection, but we found it desirable to let the last link (a hand for example) respond differently than the joint. One need only picture the action of snapping a wrist to see why this would be needed.

5.3 Damping Forces

Simple spring forces without damping tend to present numerical instabilities and they generate oscillations and vibrations that appear unnatural in a creature that is supposed to move through autonomous power in its muscles. Tests without damping had a very "marionette"-like feel. As with the spring constants, an animator can supply varying degrees of damping for each link. For simplicity, a damping force proportional to the mass and velocity of a link is used:

$$\tau = -\,\omega p m \qquad (17)$$

p *damping coefficient*
ω *angular velocity*
m *mass of the link*

5.4 Animating Parameters

We also found it useful to allow an animator to change the spring and damping coefficients as a function of time. Overall muscle exertion varies depending on the attitude of a character. At one time during an animation, a character may be lethargic which is generally depicted with limp, slumped-over movements. During this stage, both the spring and the damping coefficients would be drastically lowered, approaching the free swing of an unpowered limb. However, when that same character becomes excited, those forces would be increased to produce the quick and snappy movements associated with that alternative frame of mind. Additionally, animating values can help in the refinement process. Values that work in one section of an animation may, for one reason or another, need to be changed elsewhere.

5.5 Spring Force Release

During testing, it was often found desirable to lower the spring coefficients at the end of a rapid movement to exaggerate a "follow through." As an additional aid to the animator, the system allows the spring forces to be reduced in the presence of large resistance. A threshold is introduced which is compared against the force a spring is being asked to exert above which the spring's coefficient is relaxed. The coefficients are reduced by the following formula:

$$k' = k \left(\frac{\beta - f}{\beta - \alpha} \right)^{\gamma} \qquad (18)$$

k *normal coefficient value*
f *current spring force*
α *force threshold*
β *maximum force allowed*
γ *exponent of decay*

When a force is below α, there is no change. When it is above β, k is set to zero. In between, k is ramped down through the formula. Thus for extremely large forces, the link swings freely. As the force is removed it gradually regains control.

5.6 Kinematic Links

In practice, there are generally situations in which an animator would like to limit the effects of the dynamics simulation to a certain subset of the entire figure. An animator may require precise control of a limb for some action. Additionally, computational considerations come into play when dealing with figures with larger numbers of DOFs. Considerable savings can be made when an animator isolates the links he or she wants to have processed in the dynamics equations. Kinematic links, while still accounted for in the equations, require significantly less calculation. To incorporate the effects of their motion into the dynamics equations, we need only use their accelerations and calculate an ABI for the sub-tree in which they exist. But the more costly equations that calculate joint acceleration and velocities are not necessary.

In the current implementation, only the joint angle relationship between a kinematic link and its parent is preserved. Any interior links that are under the influence of the dynamics simulation can cause a distal link to vary from its world space position specified in a keyframed pose. Solving this problem is a topic for further research.

6. Results

The kinematic clone system has proven to be very successful at generating realistic motion that is related to the kinematic motion designed by an animator. Figure 3 shows three frames from two different animations. In each sequence, the kinematic clone is portrayed with white lines representing the links. The first shows a three link chain being swung through an arc defined with two keyframes. In these frames, one can see the inertial lag and follow through. The second three frames show a runner taking a leap. In this case, the legs and feet were excluded from the dynamics equations. Careful observation shows realistic follow-throughs of the torso, head and arms as a result of the leap itself and the ensuing impact. The torso and head were held static in the kinematic clone and thus required only a single keyframe. A similar effect was animated by hand and required eight keyframes to complete.

7. Summary

The kinematic clone technique proves to be a viable method of combining a dynamic simulation with kinematic control for character animation. Without the need for additional keyframes, movements automatically exhibit mass and momentum, especially around rapid movements and quick changes in velocity. The motions also exhibit subtle reactionary effects that are difficult to animate directly and help create the sense of realism. These subtle motions replicate what animators call a "moving hold" [1], where even at rest, small motions continue to keep a character alive.

8. Future Work

Several areas exist that merit future research. One such area is the use of alternative spring configurations. The first configuration to consider are torsion springs. Torsion springs could be used to match the *orientation* of kinematic links as well as position. Currently, orientation is passively preserved since springs are connected to both ends of a link. Under periods of extreme stress, however, joints can twist abnormally which might be remedied through torsion springs which would work against these rotations. Additionally, intra-link forces could be added that drive a figure to a rest position. Currently, as the spring forces relax, the dynamic figure starts to droop like an unpowered chain. Perhaps a spring force that works to restore a rest position would provide more natural looking results.

Joint constraints should also be incorporated in our model. The ABM accommodates single DOF joints which would be appropriate for knee and elbow joints. Joints should also have angular restrictions to prevent unnatural bending as well. When limbs were jerked violently around, violations were often seen that would be corrected with such restrictions.

Also worth considering is the calculation of the net force on a body from the movements of its limbs. Our current implementation imitates that of a "fixed base" where the movements of the joints have no effect on the position or orientation of the figure as a whole. Both Featherstone [17] and Hahn[2] present methods for determining the appropriate forces and torques. What is not clear is exactly when and how to involve their

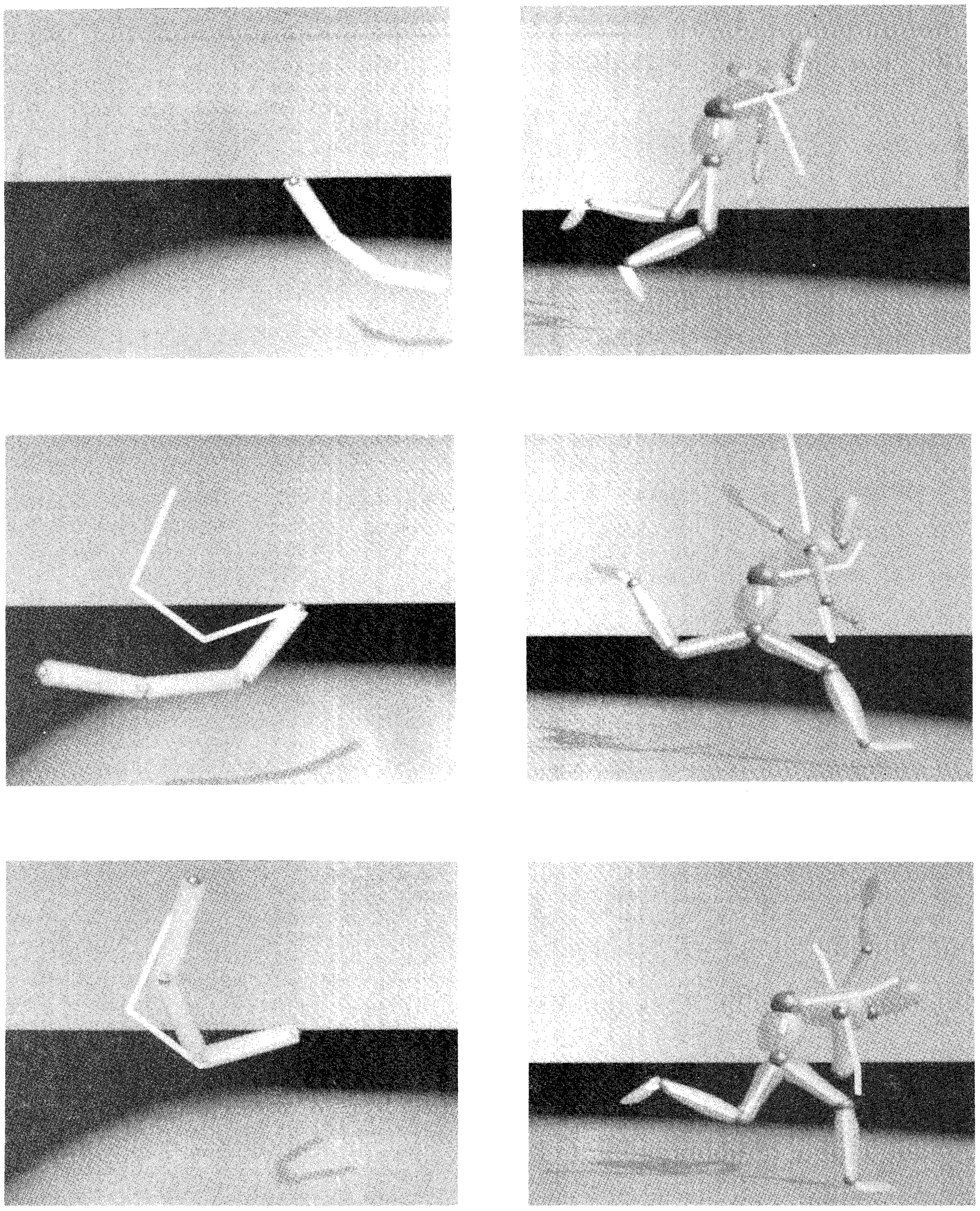

figure 3

For color plate see page 244

influence, e.g. when *should* it override the positions set forth by an animator?

Finally there is an issue regarding the degree to which the keyframed positions and timings are interpolated. Our experiments often showed considerable variances between the positions and overall timing of events. This can occur when large forces have accumulated on a link to which the springs have not yet been able to counter. Where certain events are time critical, e.g. the arrival of a hand at a position to catch a falling vase, an animator should be able to indicate that and have the system accommodate it. Perhaps in those cases a blending function could be used to gradually phase out dynamic positions in favor of the kinematic ones. The dynamics simulation could still incorporate this influence by adding the additional accelerations to the joint parameters.

9. Acknowledgments

The authors wish to thank Rhythm & Hues Studios for the supply of rendering and compositing hardware and software and for the use of their video and film editing facilities. We especially thank Steve Ziolokowski and Ian Hulbert for their modeling talents. Finally we thank Chris Perry his help in proofreading this paper.

References

[1] Thomas, F., and Johnston, O. *Disney Animation: The Illusion of Life.* Abbeville Press Publishers, New York, 1981.

[2] Hahn, J. Realistic animation of rigid bodies. *Computer Graphics (Proceedings of SIGGRAPH '88).* 22, no. 4 (August 1988) 299-308.

[3] Terzopoulos, D., Platt, J., Barr, A., Fleischer, K. Elastically deformable models. *Computer Graphics (Proceedings of SIGGRAPH '87).* 21, no. 4 (July 1987) 205-214.

[4] Reeves, W. Particle systems - A technique for modeling a class of fuzzy objects. *ACM Transactions On Graphics.* 2, no. 2 (April 1983) 359-376.

[5] Chadwick, J., Haumann, D., and Parent, E. Layered construction for deformable animated characters. *Computer Graphics (Proceedings of SIGGRAPH '89).* 23, no. 3 (July 1989) 243-252.

[6] McKenna, M., and Zeltzer, D. Dynamics simulation of autonomous legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH '90).* 24, no. 4 (August 1990) 29-38.

[7] Raibert, M., and Hodgins, J. Animation of dynamic legged locomotion. *Computer Graphics (Proceedings of SIGGRAPH '91).* 25, no. 4 (July 1991) 349-358.

[8] van de Panne, M., Fiume, E., and Vranesic, Z. Reusable motion synthesis using state-space controllers. *Computer Graphics (Proceedings of SIGGRAPH '90).* 24, no. 4 (August 1990) 225-234.

[9] van de Panne, M., and Fiume, E. Sensor actuator networks. *Computer Graphics (Proceedings of SIGGRAPH '93) Annual Conference Series.* (August 1993) 335-342.

[10] Witkin, A., and Kass, M. Spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH '88).* 22, no. 4 (August 1988) 159-168.

[11] Ngo, J. and Marks, J. Spacetime constraints revisited. *Computer Graphics (Proceedings of SIGGRAPH '93) Annual Conference Series.* (August 1993) 343-350.

[12] Cohen, M. Interactive spacetime control for animation. *Computer Graphics (Proceedings of SIGGRAPH '92).* 26, no. 2 (July 1992) 293-302.

[13] Gritz, L. and Hahn, J. Genetic programming for articulated figure motion. *Journal of Visualization and Computer Animation.* 6, No. 3. (July 1995)129-142.

[14] Isaacs, P., and Cohen, M. Controlling dynamics simulations with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics (Proceedings of SIGGRAPH '87).* 21, no. 4 (July 1987) 215-224.

[15] Armstrong, W., Green, M., and Lake, R. Near real-time control of human figure models. *IEEE Computer Graphics and Applications.* 7, no. 6 (June 1987) 52-61.

[16] Wilhelms, J. Using dynamic analysis for realistic animation of articulated bodies. *IEEE Computer Graphics and Applications.* 7, no. 6 (June 1987) 12-27.

[17] Featherstone, R. *Robot Dynamics Algorithms.* Kluwer Academic Publishers, Boston, 1987.

[18] Craig, J. *Introduction to Robotics Mechanics and Control.* Addison-Wesley Publishing Co., Reading, MA, 1989.

[19] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. *Numerical Recipies in C.* Cambridge University Press, New York, 1992.