

Automatic Motion Synthesis for 3D Mass-Spring Models

Jon Christensen, Joe Marks, J. Thomas Ngo

TR95-01 December 1995

Abstract

We describe how to automatically synthesize motion controllers for locomotive tasks involving animated characters modeled as 3D mass-spring lattices. The motion controllers determine an actuation sequence based on elapsed time, not physical state; actuation is represented economically using a canonical set of global lattice deformations; and stochastic search is used to determine effective values for the controller parameters. Our algorithm generates controllers that produce stylistic, visually plausible motion for simple locomotive tasks in under an hour on a standard workstation, which is more than an order of magnitude faster than comparable approaches to motion synthesis for 3D articulated-linkage models. Key Words: Spacetime constraints, controller synthesis, stochastic optimization.

The Visual Computer, Vol. 13, No. 1, 1997, pp. 20-28

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

1. First printing, TR95-1, January 1995.

1 Introduction

Motion synthesis is the task of automatically generating visually plausible movement of an animated character that conforms to the animator’s script [2]. (In previous work and in this paper ‘movement’ is taken to mean locomotion, though other kinds of motion are also important for animation and should be included under the general rubric of motion synthesis [3, 8].) One of the most promising approaches to the motion-synthesis problem is *controller synthesis*: instead of computing a character’s motion directly, a motion controller is computed; this controller actuates a physical model, the simulation of which produces the final motion. The controller-synthesis idea has its origins in AI and robotics [4, 5, 9, 12] and has had several incarnations in the recent computer-graphics literature [14, 17, 18, 20].

The three most general treatments of the controller-synthesis approach for animation purposes [14, 18, 20] have concentrated exclusively on articulated linkages, with which humanoid and many animal-like characters can be modeled easily. However, there are three significant problems with articulated-linkage models. First, their physical simulation comes at a considerable price: the simulator code is difficult to write, hard to debug, and high physical fidelity is computationally very demanding. Second, the synthesis of effective motion controllers for articulated linkages can be extremely costly. Very little data has been reported for motion synthesis in 3D (the most general and useful context for this problem, and the one we consider here), but 10 hours on a powerful workstation is a very conservative lower bound on the time required to compute a simple motion controller—one that can regulate walking or jumping or some similar locomotive task—for a reasonable 3D articulated-linkage model of a person or animal.¹

The third problem is motion quality: not surprisingly, simulated articulated linkages that are regulated by simple motion controllers often move with robotic rigidity and stiffness. (This is especially true when faster, less-accurate 3D simulation algorithms are used, but these are often the only kind are fast enough for motion synthesis.) Moreover, this could be an intrinsic limitation: the combination of rigid-body dynamics and low-complexity motion controllers may just be too incompatible with many of the principles of traditional animation, such as “squash and stretch,” exaggeration, and anticipation [10].

So unless and until more efficient algorithms are developed, controller synthesis for 3D articulated linkages appears to be a computationally daunting task; moreover, the results may never be visually compelling from an artistic perspective. These conclusions led us to consider other, simpler physical models that might have greater visual appeal and be more amenable to efficient motion synthesis. In this paper we describe the application of controller

¹Sims reported that computing both the structure of a 3D articulated linkage and a relatively simple neural-network controller for that linkage takes “around three hours ... on a 32 processor CM-5” [18]. Auslander et al. reported a time of “just under five hours” on a workstation to compute an “acceptable” time-based banked stimulus-response (BSR) controller that causes a 3D dog-like character to walk; a refined controller took an additional five hours [1]. Both of these times require some further qualification: Sims’s characters all appear to move in a low-gravity environment, which can lead to simpler motion-synthesis problems, especially for unstable characters, because it is easier to avoid falling over in low gravity; and Auslander et al. use a simplified approach to physical simulation [7] that is much faster but less realistic than standard robotics techniques for simulating articulated linkages [11]. It is therefore difficult to draw any but the coarsest conclusions from these initial accounts.

synthesis to 3D mass-spring lattices that are subject to nonholonomic constraints (*e.g.*, collision with a floor). For these physical models, motion controllers that are comparable to previously reported controllers for 3D articulated linkages can be computed in tens of minutes, not tens of hours. The key characteristics of our approach are:

- *A time-based controller*: the actuation of the physical model is a function of elapsed time, not physical state [1, 21].
- *Economical representation of actuation*: although actuation of the mass-spring lattice is ultimately accomplished by varying the rest lengths of the springs, actuation is represented in terms of a small number of canonical global lattice deformations.
- *Stochastic search*: effective values for the controller parameters are found by parallel hill climbing [1, 6].

Our results consist of controllers that produce stylistic, visually plausible motion for several simple locomotive tasks involving mass-spring lattices of modest complexity (*i.e.*, 5-10 point masses and 10-35 damped linear springs), computed in 40 minutes on average using a Digital 3000/400 AXP workstation.

2 Technical Approach

In this section we describe the details of the mass-spring-lattice simulation, the form and function of the motion controllers, and the search process whereby near-optimal controllers are found.

2.1 Physical simulation of 3D mass-spring lattices

A simple mass-spring lattice that is representative of the ones we consider here is shown in Figure 1.² It comprises eight point masses connected by 28 damped linear springs: 12 run along the edges of the cube, 12 traverse face diagonals, and the remaining four lie along the body diagonals.

Each spring exerts along its length an equal and opposite force on the two point masses it connects. The magnitude of this force

$$f = -k(L - x) - D\dot{x}$$

where k is the spring constant, L is the rest length of the spring, and D is a damping constant. The point masses are also acted upon by gravity, forces due to inelastic collision with the floor, and friction and normal forces due to resting contact with the floor. The simple Coulomb friction model is used, with the one significant modification that friction is permitted to be stronger in one preferred direction if desired. The resulting equations of

²Note that the visual appearance of an animated character can differ significantly from the physical model used to generate the character's motion. Side-by-side examples of mass-spring lattices and their associated visual appearances are shown in Figure 5.

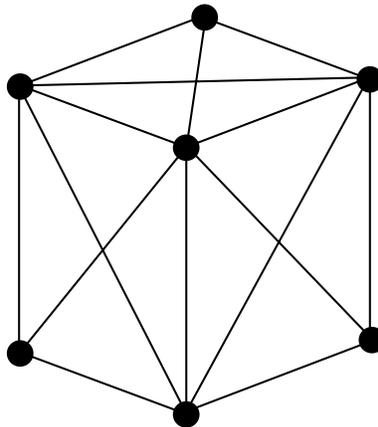


Figure 1: A simple 3D mass-spring lattice. Hidden masses and springs have been removed for clarity of illustration.

motion are solved numerically using a variable time-step, fifth-order Runge-Kutta integration procedure [16]. Actuation of the mass-spring lattices is achieved by varying the rest lengths of the springs.

2.2 Motion controllers for 3D mass-spring lattices

Initial work on controller synthesis used motion controllers that select actuation “reflexes” as a function of the physical state of the simulated model [14, 20]. More recent work has shown how controllers that select actuation reflexes as a function of elapsed time can be equally effective (at least for stable physical models) and are easier to compute [1, 21]. In our kind of time-based controller, a time interval with user-defined length t_{per} is broken into a small number C of mutually exclusive regimes. During each regime, a particular actuation reflex is performed. The sequence of regimes is repeated to fill the length of the simulation, T , which is also chosen by the user. Thus, if $t_{\text{per}} < T$, a specific sequence of reflexes will be repeated periodically; if $t_{\text{per}} \geq T$, the reflex sequence is not constrained to be periodic.

More specifically, each regime is specified by the following parameters:

- The *regime time* $t, 0 \leq t \leq t_{\text{per}}$: all points on the time line that are closer to t than to any other regime time t' belong to t 's regime.
- A *reflex number* $R, 0 \leq R \leq 11$ that indicates which one of 12 possible actuation reflexes (defined below) is associated with this regime.
- A *reflex coefficient* $\alpha, -1.0 \leq \alpha \leq 1.0$ that indicates the magnitude of the actuation reflex.

A sample three-regime controller is shown in Figure 2.

The novel feature of the controller representation and the key to its economy is the use of a canonical set of a dozen actuation reflexes that can be applied to all 3D mass-spring

$$C = 3, t_{\text{per}} = T = 5.0$$

<i>Regime Time</i>	<i>Reflex Number</i>	<i>Reflex Coefficient</i>
$t_1 = 1.5$	$R_1 = 2$	$\alpha_1 = 0.2$
$t_2 = 2.5$	$R_2 = 5$	$\alpha_2 = -0.7$
$t_3 = 4.2$	$R_3 = 7$	$\alpha_3 = 0.5$

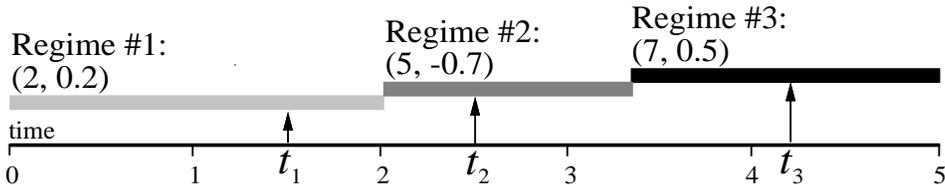


Figure 2: Illustration of a time-based motion controller.

lattices. (An earlier, less-compact representation in which an actuation reflex could be an arbitrary combination of spring rest lengths produced less coordinated motions, and the search process (§ 2.3) was much slower.) These reflexes are based on global deformations of the lattice: scaling, shearing, splaying, and twisting (see Figure 3).³ A deformation is applied about the center of mass of the lattice and along one of the three axes in a lattice-centric coordinate system. The rest lengths of the springs for the current regime are then computed from the post-deformation positions of the masses. Thus in the case of the lattice shown in Figure 1, the reflex number and the reflex coefficient alone uniquely specify the rest lengths of all 28 springs. Furthermore, the only combinations of rest lengths that can be represented are ones that deform the lattice in a coordinated and useful way.

2.3 The search process

Synthesizing a useful motion controller is formulated as an optimization problem: the task is to find values for the controller parameters that are near optimal with respect to some objective function. The functions we consider here are relatively simple. For example, the distance traveled by the center of mass is an objective function that typically elicits a walking or leaping motion. Usually the primary term in the objective function must be complemented by one or more secondary terms to obtain a particular kind of motion [1]. Examples of useful primary and secondary terms are described in Table 1.⁴

The process whereby optimal controller-parameter values are discovered is one of stochastic search. The search algorithm is a form of parallel hill climbing [1, 6], outlined in Figure 4.

³These deformations are very similar to the deformation modes described by Pentland and Williams [15]. However, we use the deformation concept in the controller representation only, and not in the physical simulation (though doing so may be a good idea).

⁴In addition to specifying the objective function, the animator can also influence the motion generated by choosing values for t_{per} , T , the gravitational force, and the coefficients of friction in the preferred and nonpreferred directions.

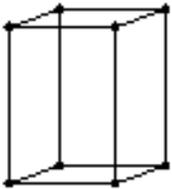
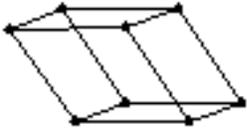
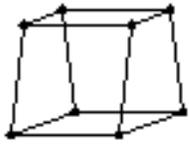
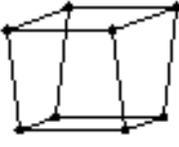
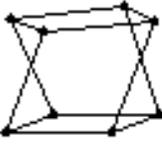
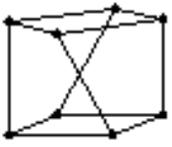
<i>Deformation</i>	$\alpha = -1.0$	$\alpha = 1.0$
Scale		
Shear		
Splay		
Twist		

Figure 3: Deformations of a cube-shaped lattice along one axis (some springs have been omitted for clarity). Analogous deformations along the other two axes bring the total number of different possible deformations to 12.

<i>Term</i>	<i>Effect</i>
Final horizontal displacement of the lattice's center of mass	A primary term that is used for eliciting walking or bounding motions
Maximum vertical displacement of the center of mass	A primary term that is used for eliciting jumping motions
Minimum number of contact points with the floor	A secondary term that can penalize motions in which the lattice leaves the ground
Total time in which the lattice is in contact with the floor	A secondary term that is used to promote motions that keep the lattice airborne as much as possible
Total rotation of a triplet of point masses about the center of mass	A secondary term that can promote rotary motions
A boolean function indicating loss of uprightness, left-right orientation, or front-facing direction	A secondary term that can penalize motions in which the lattice falls over
Average asymmetry of the lattice about a specified axis	A secondary term that is used to promote asymmetric movement

Table 1: Example objective-function terms.

The details unspecified in the figure concern the initialization and perturbation steps. For both of these steps, regime times and reflex numbers are selected randomly from uniform distributions over the ranges $[0.0, t_{\text{per}}]$ and $[0, 11]$, respectively. Reflex coefficients, however, are selected from a nonuniform distribution: values near the extremes of the range $[-1.0, 1.0]$ are more likely to be selected than values near zero. This promotes the use of exaggerated, purposeful movements over minor, twitch-like movements. In addition to randomly picking new values for regime times and reflex coefficients, a controller can also be perturbed by adjusting existing values slightly, (*i.e.*, by a random factor less than 10%). The different kinds of perturbation and their respective probabilities of application are given in Table 2.

The particular values given here for the parameters of the search algorithm (*e.g.*, the size of the search set, the number of iterations, and the probabilities of the various perturbations) are not essential to its correct operation. In informal testing the algorithm has also worked well with a range of different search-parameter values.

3 Results

We have tested our approach on the mass-spring lattices shown in Figure 5. Although the lattices are simple, the visual appearances we have associated with them are more complex and interesting. For example, the spider's visual appearance is related to the pyramidal lattice as follows: the feet of the spider are coincident with the point masses at the corners of the pyramid; the top of the spider's tubal head is located at the pyramid's apex; the spider's knees are positioned a little above the midpoints of the springs connecting the corner base

```

Initialize search set to contain 100 random controllers
Rank order the controllers in the search set
for  $i = 1$  to 25,000
  Randomly select a controller from the search set
  Perturb the controller and re-evaluate it
  Insert the new controller into the search set by rank
  Delete the lowest-ranked controller from the search set
end for
Return the best controller in the search set

```

Figure 4: Parallel hill climbing.

<i>Probability</i>	<i>Perturbation</i>
0.32	Pick a new reflex number for one regime
0.08	Pick a new reflex number for every regime
0.10	Pick a new reflex coefficient for one regime
0.02	Pick a new reflex coefficient for every regime
0.22	Adjust a reflex coefficient for one regime
0.06	Adjust a reflex coefficient for every regime
0.05	Pick a new regime time for one regime
0.01	Pick a new regime time for every regime
0.11	Adjust a regime time for one regime
0.03	Adjust a regime time for every regime

Table 2: Different ways of perturbing a controller.

masses to the mass at the apex; and the spider’s thighs connect his knees to the bottom of his head, which is a little less than midway between the base and apex. A consequence of this relationship is that the limbs and head of the spider can change length as the pyramid deforms. However, we regard this as a feature and not a bug, because it adds to the visual appeal of the characters and their movements.

Sample motions for the different characters are shown in Figures 7–9. In Figure 7, a spider leans back and then leaps forward. This movement illustrates well the kind of exaggerated anticipation that is characteristic of many of the motions generated by this approach. The motion is actually periodic (the sequence shown in the figure is repeated four times during time $T = 4.5t_{\text{per}}$), and was elicited using a four-regime controller (*i.e.*, $C = 4$) and an objective function that rewards net horizontal movement in a specific direction and penalizes motions in which the pyramid fails to remain upright.⁵

In Figure 8, a poodle modeled as a cube-shaped lattice (the lattice encompasses only the lower half of his body) performs a perfect backward somersault while jumping forward in a low-gravity environment. This aperiodic motion begins with a small jump to build up momentum, after which a perfectly timed leap propels the poodle forward with greater speed while also imparting the necessary angular momentum for the reverse somersault. The objective function that led to this motion rewards forward motion, maximum height, rotation, and being on the ground at time T ; it also penalizes inappropriate rolling and turning, and contact time with the ground. The controller has eight regimes.

A cube-shaped lattice was also used as the physical model for the table featured in Figure 9. Our goal was to have the table perform a pirouette in a low-gravity environment. In order to achieve maximum rotation, the table first twists one way, then the other, then finally jumps vertically while unwinding from the second twist. To elicit this aperiodic motion we used a four-regime controller and an objective function that rewards maximum height, net rotation about a vertical axis, and being on the ground at time T ; it penalizes loss of uprightness, contact time with the floor, net horizontal displacement, and the number of “hops” taken (*i.e.*, the number of times the model loses contact with the ground).

The performance of the search algorithm (§ 2.3) is illustrated in Figure 6. A “learning plot” is a scatterplot that indicates the objective-function value returned at each iteration of the search. Thus it is possible to determine not only the best solution found so far, but also how much time the algorithm is spending on the investigation of good, medium, and bad solutions. The plot in the figure is for the search process that found the controller illustrated in Figure 8. The 25,000 iterations took 66 minutes on a Digital 3000/400 AXP workstation. The value of the best controller found so far increased quickly through the first 5,000 iterations, but the rate of progress decreased significantly after that. This behavior is fairly typical of the other learning plots we have looked at, so we recommend running the search process for no more than 15,000 iterations, which would take about 40 minutes for the cube-shaped lattice, somewhat more for the chair-like lattice, and somewhat less for the pyramidal lattice. Note that even at the end of the search, many evaluated controllers are not very good: this is due to the disruptive nature of some of the perturbations, which may take a good controller and wreck it. However, the benefit of disruptive perturbations is that

⁵There are a surprising number of very effective locomotive strategies for the lattices in Figure 5 that involve falling over, so this secondary term is usually quite necessary to guarantee upright movement.

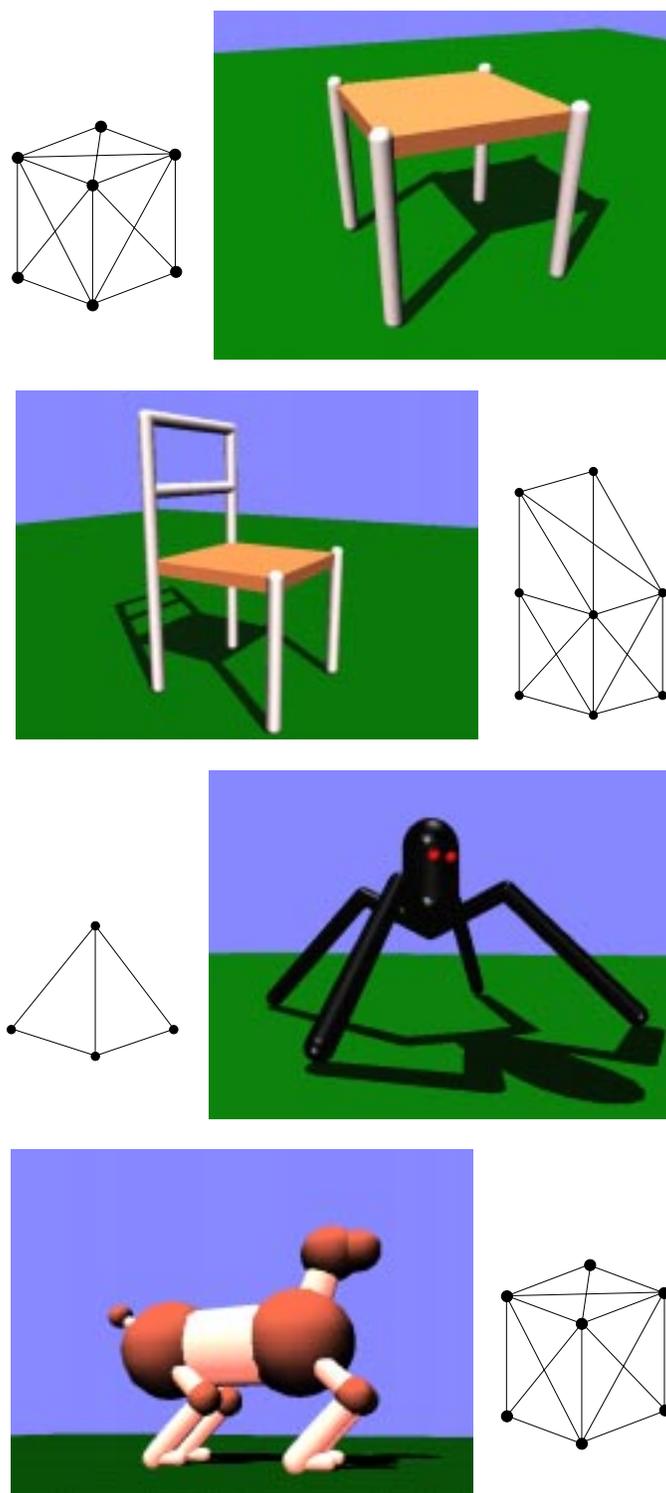


Figure 5: Some mass-spring lattices (with hidden masses and springs removed) and their associated visual appearances.

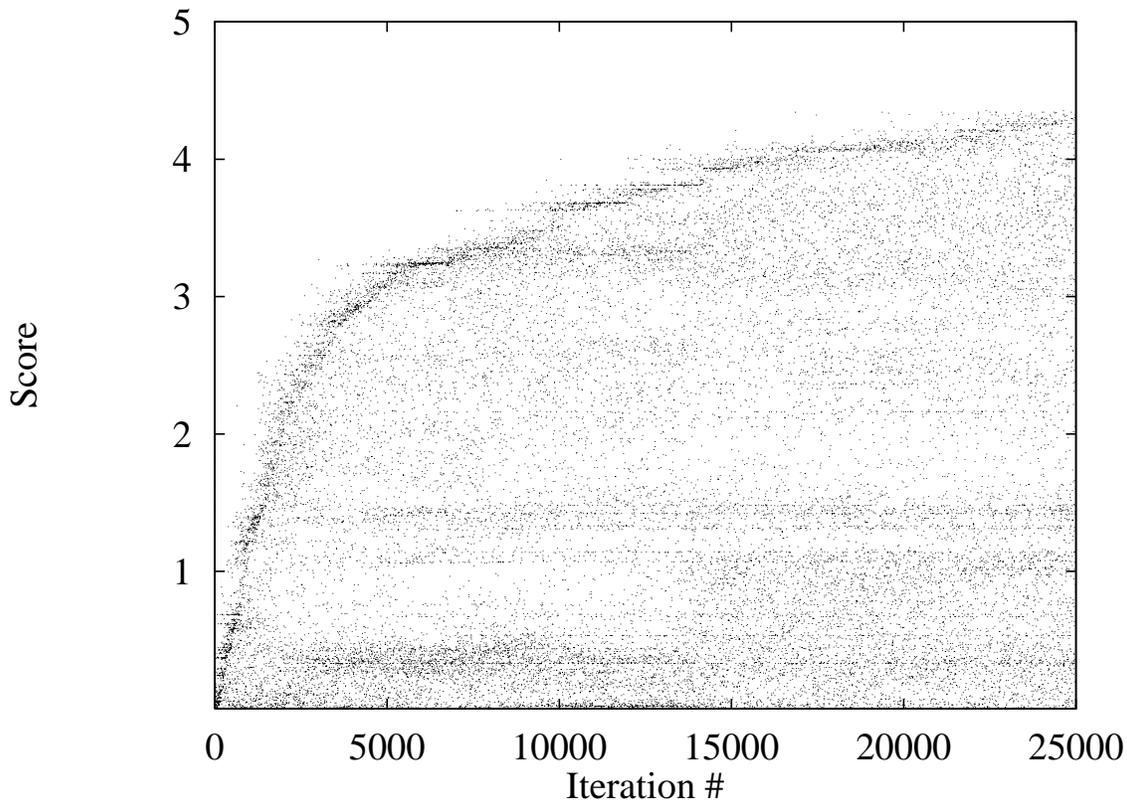


Figure 6: A learning plot.

they reduce the tendency for the search to get stuck in local minima of the search space.

4 Conclusions and Future Work

Mass-spring lattices have been used before for various niche applications in physically based computer animation, such as snake locomotion [13], fish locomotion [19], and facial animation [22]. The major difficulty preventing their more widespread use, especially for animations in which the use of articulated linkages might appear more natural, is the control problem: crafting a controller by hand for some specific locomotive task and a given mass-spring lattice is often a feat in itself.

In this paper we have shown how automatic motion-synthesis techniques can be applied effectively to animation problems involving mass-spring lattices. Indeed, the combination of motion synthesis and 3D mass-spring lattices may be more potent than the combination of motion synthesis and 3D articulated linkages because of the relative ease of implementation of the underlying physical simulation, the greater efficiency of the motion-synthesis process, and the distinctive stylistic motions that are generated.

In future work we hope to use the principles of traditional animation [10] to guide the further modification of our physical models and motion-controller representation (§ 2.2). For

example, the exclusive use of affine deformations for the actuation reflexes would ensure that a lattice's volume remains constant, an important rule for effective "squash and stretch," and requiring that the deformation used for the first actuation reflex be preceded by a small inverse deformation would guarantee some degree of anticipation in the resulting motion.

Although we have acquired considerable facility in devising useful primary and secondary terms for inclusion in objective functions, this aspect of automatic motion synthesis will not appeal to everyone. We plan to explore the use of gestural interfaces for expressing desired motion characteristics as an alternative to the mathematical specification of objective functions.

References

- [1] J. Auslander, A. Fukunaga, H. Partovi, J. Christensen, L. Hsu, P. Reiss, A. Shuman, J. Marks, and J. T. Ngo. Towards practical automated motion synthesis. Unpublished manuscript, 1994.
- [2] N. I. Badler, B. A. Barsky, and D. Zeltzer, editors. *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*. Morgan Kaufmann, San Mateo, CA, 1991.
- [3] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, Oxford, England, 1993.
- [4] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, Summer 1992.
- [5] H. de Garis. Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 132–139, Austin, Texas, June 1990.
- [6] A. Fukunaga, J. Marks, and J. T. Ngo. Automatic control of physically realistic animated figures using evolutionary programming. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming (EP94)*, pages 76–83, San Diego, CA, February 1994. World Scientific, Singapore.
- [7] J. K. Hahn. Realistic animation of rigid bodies. *Computer Graphics (Proc. of SIGGRAPH '88)*, 22(4):299–308, August 1988.
- [8] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *SIGGRAPH '94 Conference Proceedings*, pages 395–408, Orlando, FL, July 1994. ACM SIGGRAPH.
- [9] J. R. Koza and J. P. Rice. Automatic programming of robots using genetic programming. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 194–201, Menlo Park, California, 1992. American Association for Artificial Intelligence.

- [10] J. Lasseter. Principles of traditional animation applied to 3D computer animation. *Computer Graphics (Proc. of SIGGRAPH '87)*, 21(4):35–44, July 1987.
- [11] J. Luh, M. Walker, and R. Paul. On-line computational scheme for mechanical manipulators. *Trans. ASME, J. Dynamic Systems, Measurement, and Control*, 102:69–76, 1980.
- [12] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 796–802, Menlo Park, California, 1990. American Association for Artificial Intelligence.
- [13] G. S. Miller. The motion dynamics of snakes and worms. *Computer Graphics (Proc. of SIGGRAPH '88)*, 21(4):169–173, August 1988.
- [14] J. T. Ngo and J. Marks. Spacetime constraints revisited. In *SIGGRAPH '93 Conference Proceedings*, pages 343–350, Anaheim, CA, August 1993. ACM SIGGRAPH.
- [15] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proc. of SIGGRAPH '89)*, 23(3):215–222, July 1989.
- [16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, second edition, 1992.
- [17] G. Ridsdale. Connectionist modelling of skill dynamics. *The Journal of Visualization and Computer Animation*, 1:66–72, 1990.
- [18] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94 Conference Proceedings*, pages 15–22, Orlando, FL, July 1994. ACM SIGGRAPH.
- [19] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *SIGGRAPH '94 Conference Proceedings*, pages 43–50, Orlando, FL, July 1994. ACM SIGGRAPH.
- [20] M. van de Panne and E. Fiume. Sensor-actuator networks. In *SIGGRAPH '93 Conference Proceedings*, pages 335–342, Anaheim, CA, August 1993. ACM SIGGRAPH.
- [21] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94*, pages 208–215, Banff, Alberta, May 1994.
- [22] K. Waters. A muscle model for animating three-dimensional facial expression. *Computer Graphics (Proc. of SIGGRAPH '87)*, 21(4):17–24, July 1987.

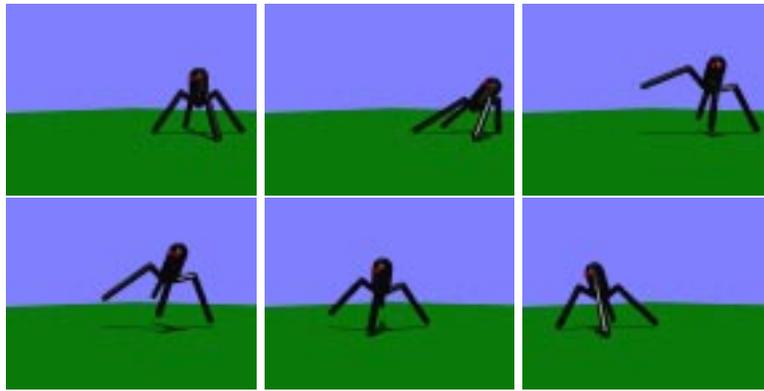


Figure 7: A leaping spider.

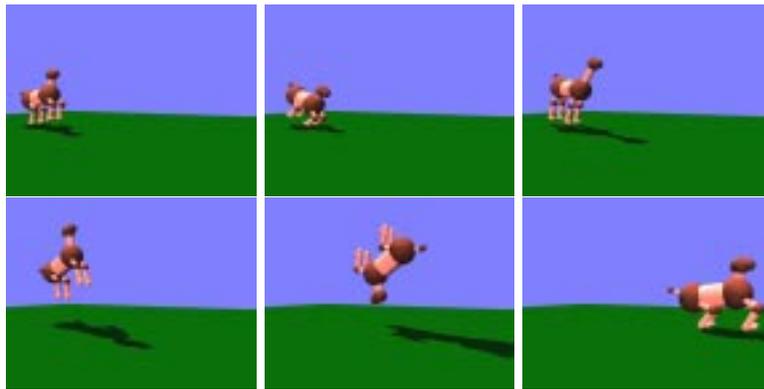


Figure 8: A poodle performing a somersault.

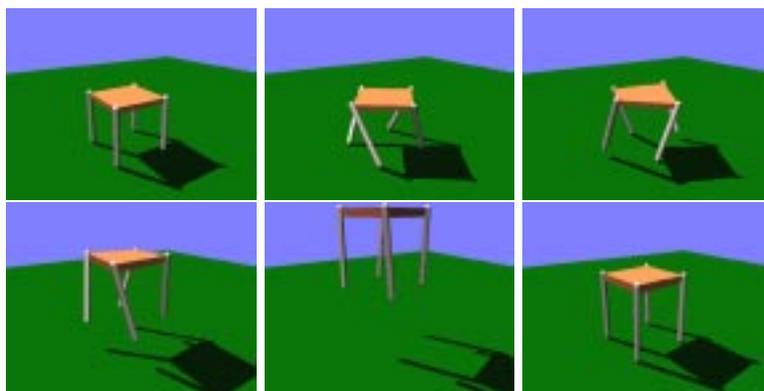


Figure 9: A table performs a pirouette.