Parameterized Gait Synthesis

Michiel van de Panne
University of Toronto

This physics-based animation technique uses control mechanisms analogous to windup toys. The parameterized control yields common gaits and other useful motions for simulated creatures, despite the lack of active control over balance.

The study of gaits dates back to the earliest attempts to draw or sculpt animals in motion. Many important questions remained unresolved until photography made it possible to obtain clear images of humans and animals in motion.[1] Recent work in animation, robotics, and biomechanics has turned from the analysis of gaits to the problem of gait synthesis, using either physics-based simulations or real robotic mechanisms. Solutions to this problem hold promise as powerful tools for animation, enabling the creation of realistic motions with minimal effort. However, gait synthesis is still a challenge.

This article presents a method of producing gaits by using control mechanisms analogous to windup toys. The synthesis technique is based on optimization. One of the primary characteristics of "virtual windup toys" is that they are oblivious to their environment. This means that these creatures or simulated toys have no active control over balance. Nevertheless, "blind" parameterized control mechanisms can produce many common periodic gaits as well as aperiodic motions such as turns and leaps. The possibilities and limitations of this technique are presented in the context of example creatures having one, two, four, and six legs. Figures 1 and 2 show animations of two of these creatures.

An important attribute of the proposed synthesis method is that the motions produced can be parameterized. Thus, you can synthesize a family of motions instead of just a single fixed instance of a motion. The examples used here are

- a hopping gait parameterized with respect to speed,
- a turning walk parameterized with respect to the turning rate, and
- a leap parameterized with respect to the size of the leap.

The animator can thus interactively specify the hopping speed, turning rate, and leap size, respectively, for these physics-based motions.

Related work

Humans, robots, and animals together form an interesting and challenging class of objects to animate because they are active. Movement must be achieved through the coordination of actuators internal to the object—the muscles, motors, or other mechanisms that cause motion.

We can contrast active objects with passive physical systems whose motions are determined solely by external forces, as is the case for a marionette or for a rock rolling down a hill. The control problem for active objects to solve for the muscle or motor actions necessary to achieve a desired motion. In general, the problem takes the form depicted in Figure 3.

Over the past two decades, a variety of workable solutions to the control problem for motion synthesis have been proposed and implemented, with application to computer animation, robotics, biomechanics, and artificial life. The predominant methods for animation of legged locomotion are kinematic. Physical simulations are a more recent possibility. The early work of Girard[2] used kinematics and simplified dynamics to govern the behavior of legged creatures. Physical simulations guarantee that the laws of physics are obeyed and thus ensure realism. However, since the simulations do not produce desired or natural motions unless the appropriate control functions are applied, the methods for producing these control functions require investigation. The simulated creatures must somehow acquire basic motor skills.[3]

Hopping and running are presently among the most-studied classes of motions.[4,5] Research has been successfully applied to monopeds, bipeds, and quadrupeds, both on real robots and in physics-based simulations for computer animation. The controllers used for these motions are constructed by decomposing the motion into distinct

0272-17-16/96/$5.00 © 1996 IEEE

phases according to the legs in contact with the ground at any point in time. Appropriate control laws are then designed for each phase.

Researchers have developed several schemes for controlling the motion of hexapods.[6] The problem in this case is more one of coordination than balance because hexapods are statically stable in many configurations. Much of the previous work on hexapod control has focused on problems beyond simple locomotion. We will deal here with the locomotion problem for a simulated ant to show that it falls within the scope of the proposed control synthesis method.

Biped locomotion has fascinated people who build walking robots as well as those who simulate it for purposes of animation. Walking is most often treated as a separate problem from running because the two motions have different types and durations of ground contact. Biped control algorithms often do not generalize beyond walking motions for a specific physical model.

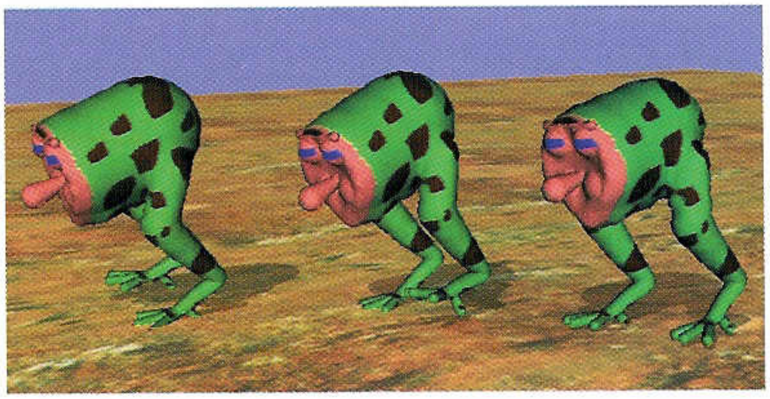
More recently, researchers have employed optimization methods to automatically synthesize locomotion controllers for simple articulated figures.[7-9] These approaches begin with a specific choice of architecture or control representation for calculating the actuator forces and torques from the sensory and state information. The most detailed approach models the neural activation as a function of time.[7] Ngo and Marks[8] use a control representation that can be described as a finite set of stimulus-response (SR) rules, where a particular actuator response is chosen based on the rule activated by the current stimuli. In the approach described by van de Panne and Fiume,[9] a weighted, nonlinear network with time delays is used to connect sensors indirectly to the actuators. This structure is referred to as a sensor-actuator network (SAN).

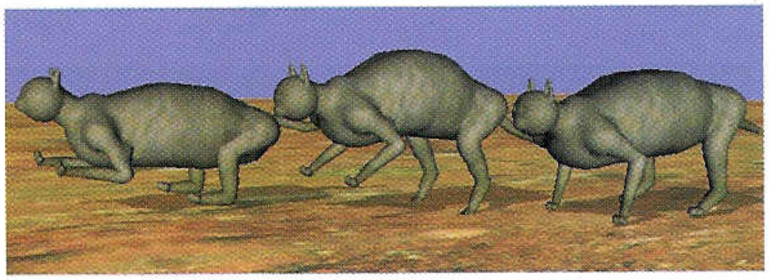
In both the latter cases, the architecture defines the form of the controller, but not its function. The function is determined by the values assigned to a set of free parameters that exist within the controller. The vector P_f will denote the set of free parameters for a controller. For the SR representation, P_f consists of the conditions necessary to trigger a particular response as well as the responses themselves. For SANs, P_f consists of the weights and time delays used in the network. In either case, P_f governs the behavior of the controller.

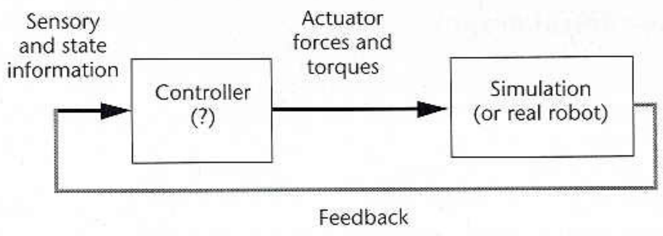
Given the control representation, the problem of finding a suitable controller for a desired motion is reduced to assigning a set of values to the free parameters. A desired motion is defined through the choice of a scalar-valued optimization function f, such as one that measures the distance traveled in a fixed amount of time or the maximal height achieved during a jump. We can search for the values of P_f that lead to an optimal value of f in several ways. The previous work with the SR representation used genetic algorithms, while separate global and local searches were used to optimize SANs.

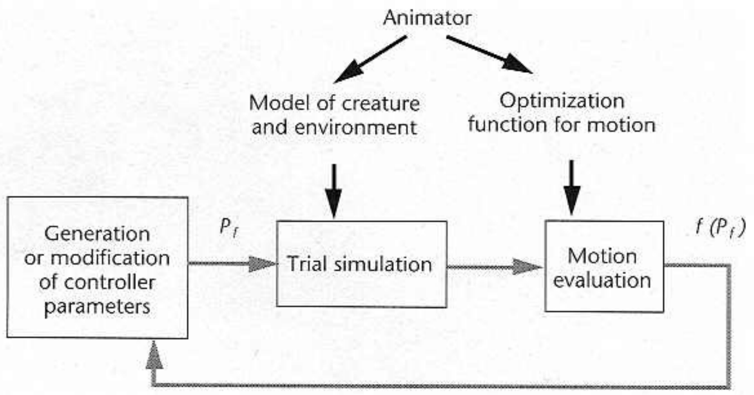
In either case, the optimization is stochastic, carried out by randomly generating or modifying the values assigned to P_f, and followed by a short trial simulation with the controller instance specified by P_f. Such a trial begins with the creature in a predefined initial state and then simulates its motion with the given controller for a fixed time duration. The optimization function is evaluated over the performed motion. The results are then used by the chosen optimization algorithm to make appropriate modifications to P_f for the next trial, as shown in Figure 4 on the next page.

More recent work[10] presented the use of pose-control graphs as a control representation and investigated several variations for optimization algorithms, as well as ways of describing the motion in terms of limit cycles, bifurcations, and potentially chaotic movements. The methods described here expand on the pose-control technique, showing that it can be extended to control more complex, 3D figures. In particular, I wish to demonstrate the ability to synthesize common gaits and to have parameterized control over these gaits.

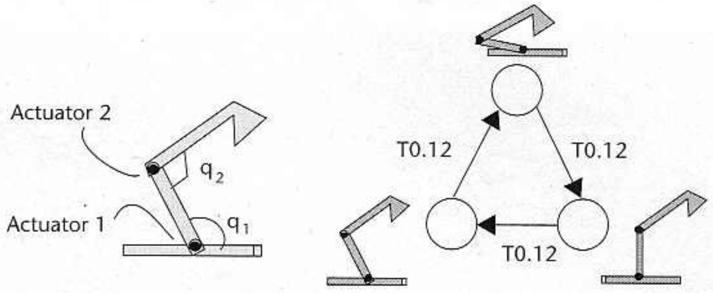

1 A physics-based walking monster.


2 A bounding fat cat.


Sensory and state information
Actuator forces and torques
Controller (?)
Simulation (or real robot)
Feedback
3 Control problem for motion synthesis.


Animator
Model of creature and environment
Optimization function for motion
Generation or modification of controller parameters
P_f
Trial simulation
Motion evaluation
$f(P_f)$

4 A generate-and-test motion synthesis system.



5 A pose-control graph for Luxo, an animated lamp. The arc labels indicate the duration of the timed state transitions.

Pose-control graphs

The control representation used here is a finite-state machine having timed transitions, as shown in Figure 5 for the animated lamp called Luxo. The labels on the arcs of the finite-state machine give the durations that the machine remains in a given state before proceeding on to the next state. The output of each state is referred to as a *pose*, which is the desired position to be attained by each of the actuators. Finite-state machines have been extensively used in many types of control problems.[11]

A pose bears a superficial resemblance to a keyframe, although the pose only specifies the desired shape for the creature and not its global position or orientation. More importantly, the object might never actually reach the configuration specified by a pose, and the animator never deals directly with poses. Only the synthesis algorithm generates and modifies poses in attempting to produce a desired motion. Poses are held constant for the duration of time spent in a state. The pose-control graph in Figure 5 consists of a cycle of three poses, each specifying the desired joint angles for Luxo's two joints.

Given a desired position for an actuator, we can use a proportional derivative (PD) controller to calculate an actuating torque:

$$\tau = k_p(\theta_d - \theta) - k_d\omega$$

where τ is the applied torque, θ_d corresponds to the desired position as specified by a pose, θ and ω are the actual angular position and velocity of the joint, and k_p and k_d are gain constants. The PD controllers cause the creature to work toward taking the shape specified by the poses in the pose-control graph. However, the creature's actual shape will rarely match that specified by the controlling pose because of external forces, namely gravity and ground reaction forces.

Although PD controllers work toward making individual joints achieve desired behaviors, it is worth remembering that the final motion of a creature is the product of all its joints working together to cause a specific interaction with the environment. Simulated land creatures must ultimately use ground reaction forces to propel themselves forward, just as an aquatic creature must be propelled forward by the reaction forces of water.

The controller shown in Figure 5 is completely described by the set of parameters

$$P = \{n, k_{pj}, k_{dj}, t_i, \theta_{ij}\},$$
$$i = 1, \ldots, n; j = 1, \ldots, m$$

where n is the number of states, m is the number of actuators for a creature, t_i is the duration of state i, θ_{ij} is the desired angle of actuator j in state i, and k_{pj} and k_{dj} are the PD constants associated with actuator j.

This parameter set is further partitioned into two subsets, P_c and P_f, where P_c contains constant parameters to be supplied by the animator and P_f contains the free parameters to be determined by the synthesis technique. In the work presented here, this partition is as follows:

$$P_c = \{n, k_{pj}, k_{dj}, t_i\}, P_f = \{\theta_{ij}\}$$

The user-supplied parameters in P_c contain important information that could potentially be synthesized automatically, but they are presently included as part of the problem specification. The number of states is an indicator of the complexity of the motion. For all the motions considered here, $n = 3$ or $n = 4$ yields the desired gaits. All transition times are initially assumed to be equal and are given by $t_i = T/n$, where T is the period of the desired gait. T is typically estimated from the creature's size. For example, we can expect the duration of an elephant's stride to be much longer than that of an ant's. Lastly, we can consider k_p and k_d to be part of the creature's design, as they specify the strength and response of its actuators. Reasonable values can be determined by calculating the necessary torques for a creature to support its own weight when standing.

The components of P_f define the search space that must be explored to synthesize desired motions. For the example of Figure 5, P_f has six parameters (two joints times three states) and thus represents a six-dimensional search space. Any point in this search space defines a controller and can be evaluated by carrying out a simulation of the creature. An optimization function is evaluated over the duration of the simulation, typically measuring quantities such as distance traveled, angle turned, or energy consumed. Simulation trials of six to eight cycles of locomotion are usually sufficient, corresponding to 3 to 4 seconds of simulated time for creatures having the scale of a large cat or human. The synthesis process must explore the multidimensional search space using repeated trials in looking for a global optimum of the optimization function.

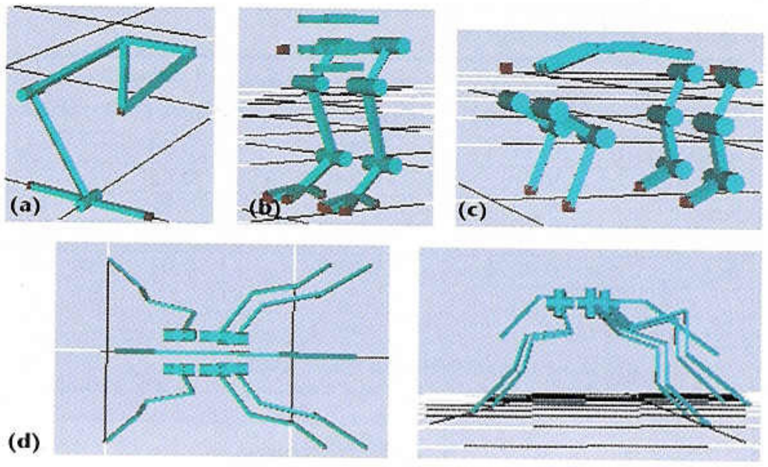
Creature modeling

The current modeling system gives creatures an articulated skeleton composed of rigid links and joints having one or two degrees of freedom. Figure 6 shows the skeletons for four creatures considered here. The skeletons are then "dressed" using a deformable, elastic skin, as shown in Figure 7. The actuators at each joint require specification of the joint range as well as the actuator "strength" implied by its PD constants, k_p and k_d. The system monitors a fixed number of prespecified points for contact with the ground, which is simulated using a spring-and-damper model. The ground contact model also approximates friction and slippage. No hard joint limits are included. Instead, the system uses joint ranges to bound the desired angles for the control of joints.

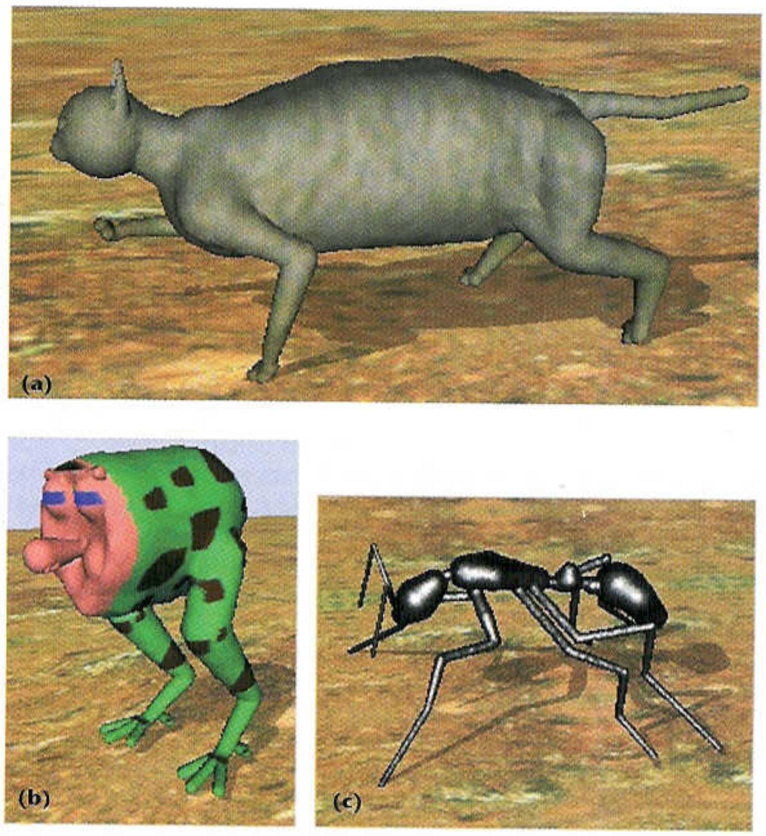
The Monster (Figure 6b) has the skeletal construction of a bird. The Cat skeleton (Figure 6c) was originally constructed using a set of realistic dimensions, but it subsequently had to be widened and lengthened to obtain stable gaits. The Ant (Figure 6d) is modeled after *Euponera Sikorae*.

We can derive the masses and moments of inertia for a model's links from either the skeleton or the volume enclosed by the skin. In the former case, the "bones" belonging to each link are assigned a uniform linear density used to determine masses and inertia tensors for the links. If the skin is chosen as the basis for calculating physical properties, we can assign a uniform volumetric density to the creature and apply a voxel flood-fill to estimate the interior volume of the creature. Each voxel is then considered to be a constituent mass of the closest articulated link, as defined by the distance from the voxel center to the closest bone.

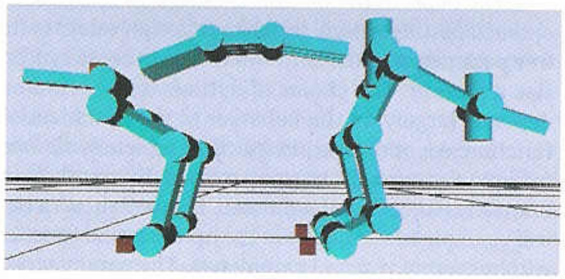
Table 1 (next page) gives the basic properties of the modeled creatures. The equations of motion necessary to physically simulate the models are produced using a commercially available simulation package.[12] In general, it can be useful to construct skeletons at various levels of detail. The control can be synthesized using the simpler skeleton, with the detailed skeleton being used to produce the final motions. A simpler skeleton has the advantage of reducing the number of actuators and hence the size of the search space, as well as speeding the simulation computations. Figure 8 shows the detailed skeleton used for the Cat.



6 Creature skeletons with one, two, four, and six legs: (a) Luxo, (b) Monster, (c) Cat, and (d) two views of Ant.



7 Creatures in motion: (a) Cat walk, (b) Monster walk, and (c) Ant tripod gait.

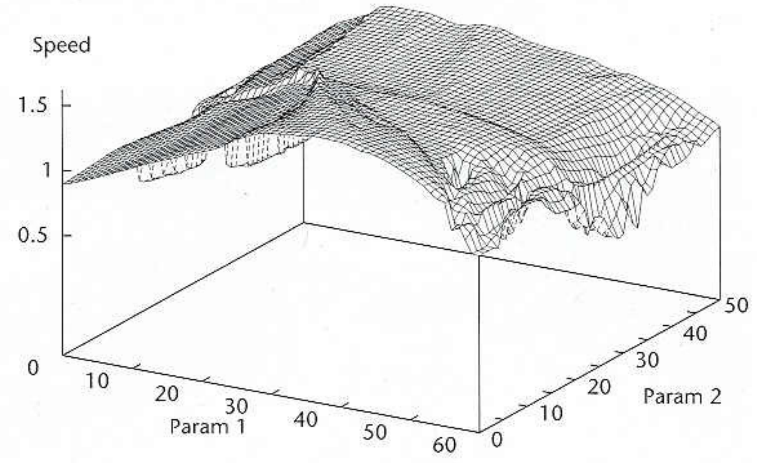


8 A refined Cat skeleton.

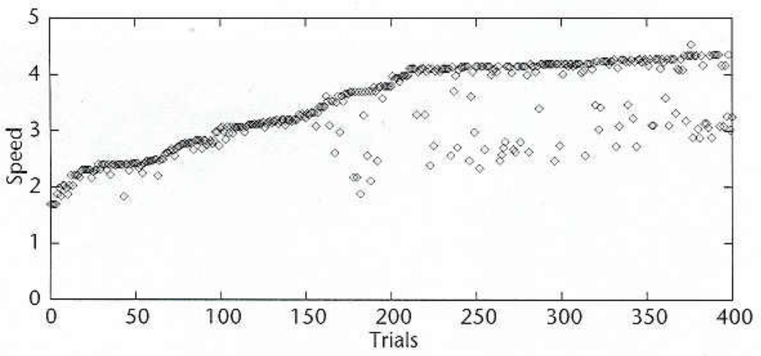
Table 1. Creature properties.

Creature	Mass (kg)	Length (m)	Height (m)	Links	Actuators	Total Degrees of Freedom
Luxo	0.55	0.20	0.60	3	2	5*
Monster	9.81	0.25	0.45	7	6	12
Simple Cat	5.77	0.355	0.30	11	10	16
Detailed Cat	6.47	0.78**	0.30	20	21***	27
Ant	0.0076	0.019	0.01	7	12	19

* Luxo model is planar.
** Includes the tail.
*** The extra 11 actuators provide only passive control.



9 Optimization space for the Cat trot.



10 Local optimization for a Cat walk.

Optimization

At the heart of the motion-synthesis process lies an optimization algorithm, which must assign values to the free parameters of the controller, P_f, in order to synthesize a motion. The choice of optimization technique depends largely on the behavior of the optimization function. An optimization function replete with local maxima (or minima) must be treated differently from one that is smooth and unimodal. Figure 9 shows a typical example of variations in the optimization function with respect to two free parameters. The parameters in this case correspond to desired angles for a single joint in the Cat for two states of a four-state pose-control graph. The graph represents the distance traveled by the cat for different values of the two chosen controller parameters. The kind of regular, exhaustive sampling necessary to create this graph is clearly prohibitively expensive for the high-dimensional space of the full optimization problem. Thus a better search method is required.

The selected optimization technique is based on a global search followed by a separate local search.[9,10] The global search carries out a fixed number of evaluations at randomly chosen points in the parameter space and retains the best results. While most trials do not produce much useful motion, there are nevertheless a small but consistent number of trials (from 1 to 5 percent) that do produce a reasonable first attempt at forward motion of various types.

The subsequent local search begins at one of the best results found by the global search and carries out a greedy modify-and-test strategy. The local search also includes the transition times between states, t_i, as part of the parameter set to be optimized. This lets us increase or decrease the time period of a stride, as is often necessary to change the speed of a gait. The algorithm makes a small change of fixed magnitude and random sign to a randomly selected parameter and evaluates the resulting modified controller to determine if the change improved performance. If it did, the change is retained. Otherwise, it is rejected. This could be considered analogous to an athlete experimenting with small changes in his or her technique, although our "athlete" is not systematic in selecting the change to be attempted.

Figure 10 shows the performance of the Cat during a series of 400 local optimization trials to obtain a walking gait. In this case the optimization function is speed, and the walk eventually becomes a trot. The effectiveness of the search procedure depends in part on the parameters associated with the described algorithm. For the simulations here, the modifications applied to desired joint angles during the local search typically correspond to 4 to 10 percent of the actuator's range. Many refinements are possible for the local search process.[10]

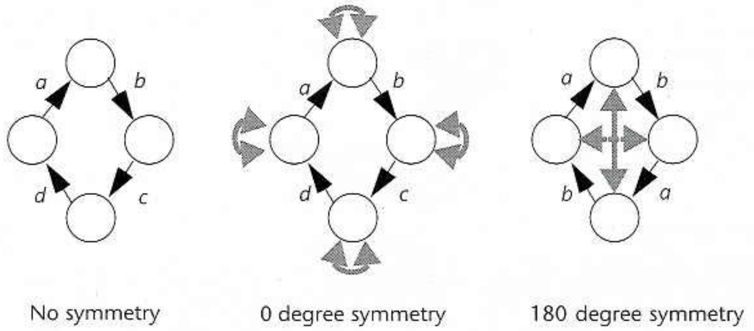
Table 2. Summary of synthesized gaits.

Gait	Creature	Symmetry Constraints	Global Selection Criteria	Local Optimization Function
1-legged hop	Luxo	None	Speed	Speed
2-legged hop	Monster	0	Speed	Speed
2-legged walk	Monster	180	Speed	Speed
4-legged trot	Cat	180	Speed	Speed
4-legged rack	Cat	180	Visual	Speed
4-legged walk	Cat	180	Visual	Lateral rock
4-legged bound	Cat	0	Speed	Speed
6-legged tripod	Ant	180	Speed	Speed
6-legged wave	Ant	0	Speed	Speed

Gait synthesis

In these experiments, we can arrive at the most common gaits by choosing speed as an optimization metric, imposing suitable symmetry conditions on the controller, and being selective about the gait obtained from the global search phase. Table 2 summarizes the gaits and their synthesis criteria.

To reduce the size of the parameter space in searching for useful gaits, it helps to take advantage of symmetries exhibited in the desired gaits. In these cases, we can take advantage of left-right symmetry, as the creatures here and their desired gaits exhibit such lateral symmetry.



11 Symmetry conditions for a four-state pose-control graph (shaded arrows indicate lateral symmetry).

Three possibilities emerge with respect to using lateral symmetry, as shown in Figure 11. The first is to impose no lateral symmetry constraints at all. This is useful in synthesizing turning motions, for example. The second is to enforce a constraint that left and right limbs operate synchronously. The last possibility is to have left and right operate out of phase, as occurs when limbs take alternating steps. This requires an even number of states in the pose-control graph so that each state can have its out-of-phase counterpart perform the same action one half stride later. The symmetry constraints are imposed directly on the poses of the pose-control graph. Both types of lateral symmetry constraints reduce the dimensionality of the parameter space by two. The animator fixes the choice of symmetry constraint as part of the desired motion specification, before initiating the synthesis.

In general, symmetry constraints can be implemented between arbitrary pairs of legs, for example, diagonal pairs to obtain a trot.[4,5] However, this can present a problem for cases where the legs involved do not have identical construction. Many creatures exhibit only lateral symmetry, including our quadruped, the Cat. As a result, the specification of the desired lateral symmetry alone is still insufficient for automatic synthesis of all the gaits we might wish. For example, quadruped walks, trots, and racks all exhibit out-of-phase lateral symmetry. We solve this problem by relying on the well-defined local maxima that common quadruped gaits exhibit when speed is chosen as an optimization function.

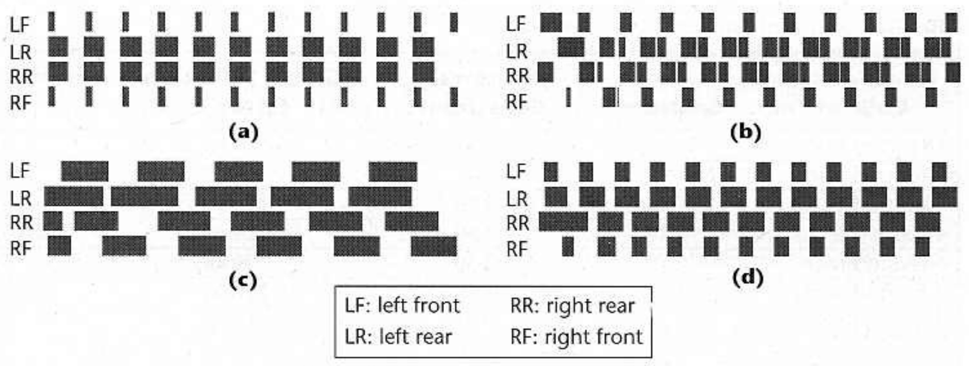
To automatically synthesize quadruped walks, trots, and racks, this gait-synthesis system first carries out a single initial global search using a lateral symmetry constraint. The best solutions in terms of speed are then visually examined and classified as resembling trots, walks, or racks. At present, it is unclear how this visual classification could be best automated. For the Cat, the speediest gait found in a global search is consistently a trot, so this particular case requires no visual classification.

The best selected trot and rack gaits from the global search retain their basic characteristics (that is, relative phasing of the limbs) when they are individually further optimized for speed using the local optimization. This indicates that these gaits represent well-established local maxima in terms of speed. Unfortunately, the same cannot be said for quadruped walks, which tend to turn into trots when further optimized for speed. Improved walking gaits were obtained by minimizing the lateral rock, which is defined as

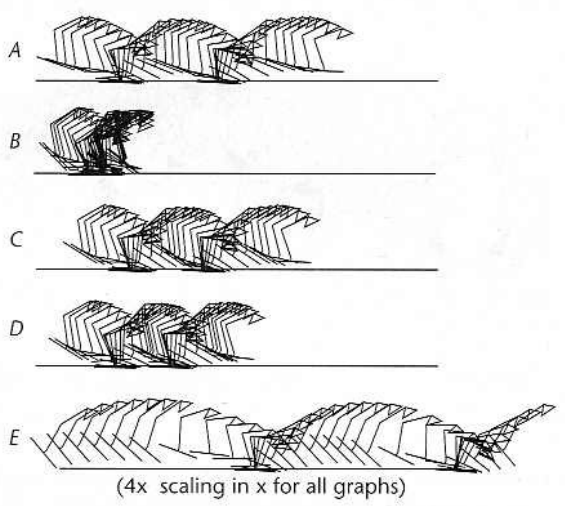
$$\int \omega^2(t)dt$$

where ω is the angular velocity about the body's longitudinal axis. Figure 12 (next page) shows the footfall patterns obtained for the various synthesized quadruped gaits.

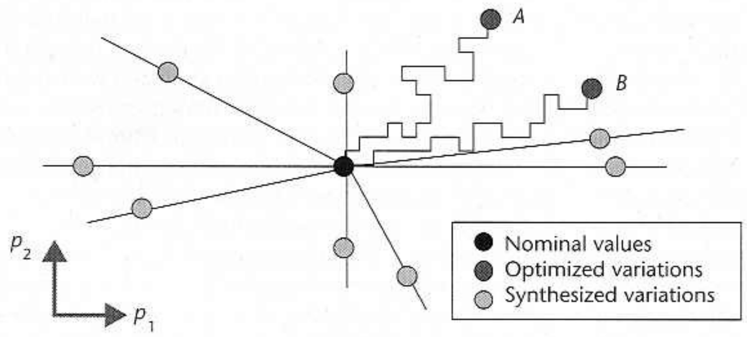
In summary, the synthesis of both walking and running gaits can take place in a framework that does not require dividing the motion into distinct phases based on the different contact configurations with the ground that may occur. A clear deficiency of the pose-control graphs is the lack of active balance in the control they provide, especially for the walking and hopping motions of the



12 Footfall patterns for various gaits for the Cat. The horizontal axis indicates the passage of time. Shaded areas indicate that the given foot is in contact with the ground. The gaits shown are (a) bound, (b) trot, (c) walk, and (d) rack.



13 Interpolating between gaits. (a) Nominal gait for this example. (b) A variation of the gait in (a), which tilts the front of the base further down during a jump. (c) A gait determined by a controller calculated to lie midway between that of (a) and (b), and then simulating the result. (d) Result of using the equivalent kinematic interpolation between (a) and (b). (e) The gait in (c) further mixed with a fast gait.



14 Obtaining variations of a motion, as viewed in the controller parameter space.

Monster. All the creatures retain some measure of dynamic stability, but this is because contact with the ground drives the system toward a limit cycle. The attraction to this limit cycle must be enhanced by active feedback to improve the simulated creatures' sense of balance.

Parameterizing motions

Parameterized motions are an effective way of dealing with the enormous space of all possible motions. They provide a useful substrate for high-level motion planning and for constructing more abstract motion representations. They also serve to amortize the computational cost of controller synthesis if controllers suitable for producing an entire family of motions are produced,

tion method described here is based on interpolation between existing, synthesized controllers to yield controllers providing the appropriate in-between motions.

Note that interpolating between two controllers to produce an in-between motion is different from interpolating directly between the two motions. The motion obtained using the interpolated controller has the desired characteristics of an interpolated gait while at the same time being faithful to the laws of physics. In general, kinematic interpolation between motions can violate physical constraints. Figure 13 shows an example illustrating the difference between these two types of interpolation.

The parameterized motions are built on the same assumption underlying the local optimization phase of the synthesis process, namely that a small change to the parameters of a synthesized pose-control graph, P_s, usually leads to a small change in the resulting motion. As a result of this property, we can interpolate between similar motions by interpolating between their controller parameters. We shall define similar motions as any in a set of motions originally derived from the same pose-control graph.

Before further discussing interpolation, we must know how to create a set of similar controllers to use as the basis for interpolation. One method already discussed is to optimize a motion with respect to a chosen function. These typically lead to controllers such as those marked *a* and *b* in Figure 11. The graph illustrates how two controller parameters typically change during an iterative optimization process. The current implementation makes changes to one parameter at a time, resulting in Manhattan walks in the parameter space.

Another method of automatically generating variations of a motion is to randomly choose a direction in the parameter space and explore changes in this direcexploration results in synthesized variations of the type also shown in Figure 14. This approach requires specifying a single similarity metric. To date, only speed has been used as a similarity criterion. Any gait having a speed within ±30 percent of the nominal gait is classified as being similar.

After we have created a set of controller variants, we can produce parameterized motions by interpolating between them. A parameterized controller is given by

$$P = \sum w_i P_i,\ \sum w_i = 1,\quad w_i \in [0,1]$$

where the parameter vectors $P_1 \ldots P_n$ define the controller variants.

Linear weightings of control parameters generally result in predictable motions, but not always. Convex combinations of controllers are generally the most predictable, although extrapolation can be used to produce exaggerated motions, which themselves could prove useful for producing entertaining animation. In all cases, the motion instance is produced by carrying out a simulation using controller P.

Figure 15 shows the results of interpolating between a slow gait and a fast one to produce a speed parameterization for the walking Cat. The fast gait was obtained by optimizing the slow gait for speed. Although the speed is not completely linear with respect to the interpolation parameter k, it is in general unimodal and well behaved. Figure 16 shows a turn parameterization for the Monster, achieved by interpolating between a forward walking gait and one optimized for turning. In this case the parameterization is clearly not as well behaved, although it proves sufficient to build a working (but not very stable) path-following controller for the Monster.

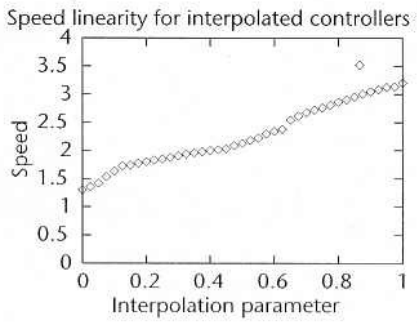
It is also possible to experiment interactively with linear combinations of existing controllers. Figure 17 shows an example of the current interface. The large window on the left displays an ongoing physical simulation using an interpolated controller, while the nine smaller windows play back animations of gait variations used as the basis for the interpolation. The sliders associated with the nine smaller windows can increase the relative weight of the current controller parameters toward those of the chosen motion variation. The changes are reflected immediately in the ongoing simulation displayed on the left.

Aperiodic motions

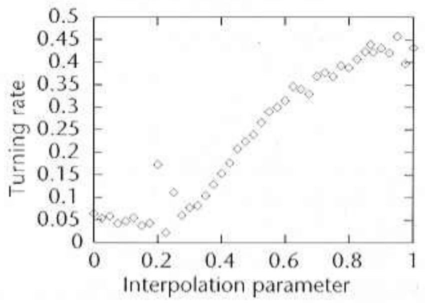
For aperiodic motions that are variations of periodic motions, we can "unwind" a synthesized cyclic pose-control graph to yield a linear chain of poses. Here we consider the example of Luxo, the hopping lamp, performing a large leap halfway through a series of hops. We begin with a synthesized cyclic pose control graph having poses A, B, and C, as shown in the top left of Figure 18. We now unwind two cycles of the motion so that a linear chain of poses exists at the time of the desired leap. Four states in the chain are marked as being modifiable, as is the timing of the transitions between these states. The poses of these modifiable states and their timed transitions become the parameter set to be optimized.

The leap itself is specified by using the distance traveled over all the hops as an optimization function. The periodic motion serves as a point of departure for the motion synthesis, thus the leap requires only the local optimization procedure. As the modify-and-test trials proceed, the middle hop becomes a full-fledged leap. Proper anticipation and recovery is ensured because the entire sequence of hops is simulated during each trial. All changes resulting in a fall are rejected.

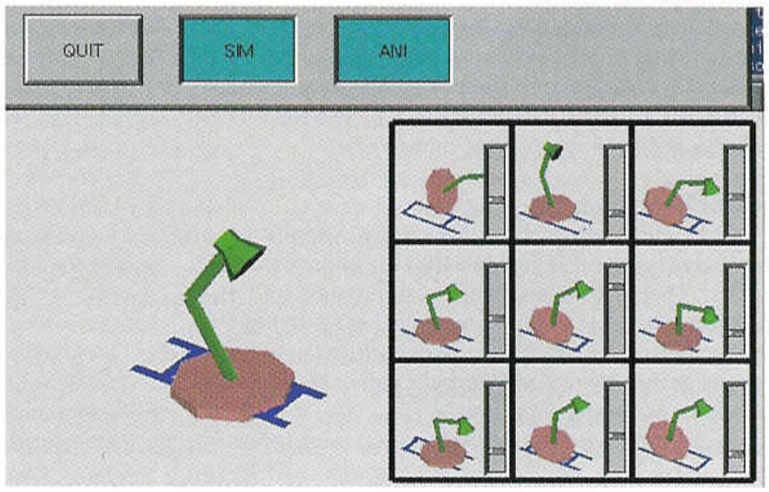
Figure 19 shows the results of synthesizing a leap. The synthesis of such a motion typically requires on the order



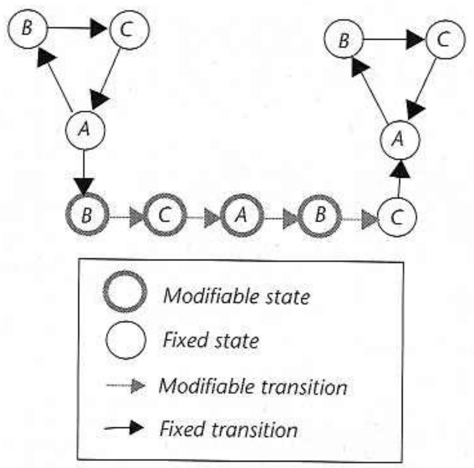
15 Speed parameterization for a Cat walk using linear interpolation between two controllers.



16 Turn parameterization for the Monster using linear interpolation between two controllers.



17 The display for interactive gait design.



18 Unwinding a pose-control graph. A periodic motion can serve as the basis of an aperiodic motion by unrolling a portion of the cyclic pose-control graph.

of 300 simulation trials, which take approximately one hour to compute on a modern workstation (~60 SPECfp92).

The result can also be parameterized, because in determining the control for as large a leap as possible, we have also determined the control for intermediate-size leaps. Figure 19d shows the result of a medium-sized leap, obtained by interpolating between the original and final pose-control graphs of the synthesis process. A creature could directly use the parameterized control in conjunction with an "obstacle detector" to always generate appropriate jumps.

Conclusions

This work shows that pose-control graphs are a sufficient control representation for many common gaits. Their synthesis can be automated and the resulting control (and hence the resulting gaits) can be parameterized with respect to quantities such as the speed or turning rate of a periodic gait or the jumping distance of a leaping motion.

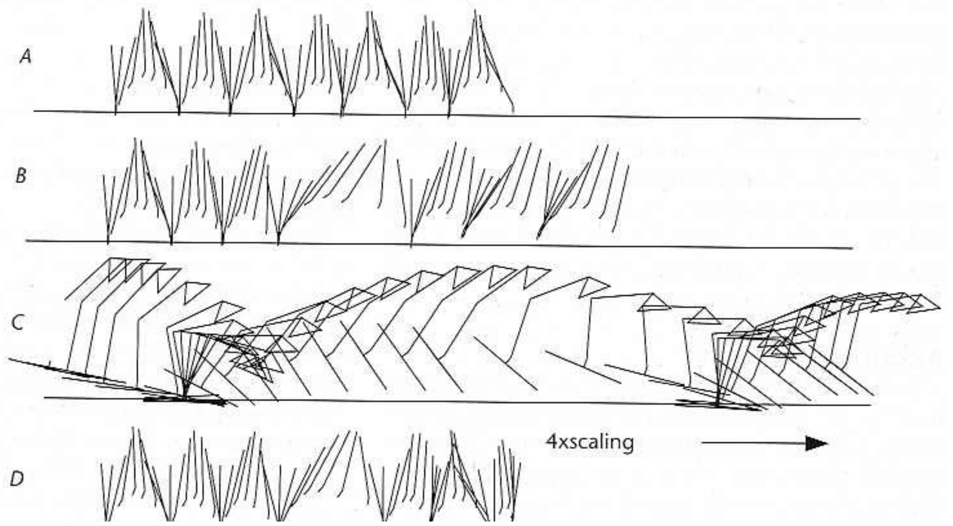
Techniques for computer animation must typically strike a compromise between detailed control over th motion and an interface requiring minimal effort t specify a motion. Within this spectrum, the parameter ized controllers discussed here can be used at two di ferent points: as building blocks for constructing highl autonomous locomotion behaviors or as a means t return some measure of control to the animator in a otherwise automated system.

Many interesting problems related to gait synthesi remain to be explored. The lack of active feedback in th control means that dynamically stable controllers can not always be found. The question of how best to auto matically synthesize controllers that make use o sensory information is both interesting and challeng ing. Quadruped gallops have not yet been addressed nor has the problem of automatically synthesizing arbi trary gait transitions. Lastly, better modeling tools ar required to build skeletal models that more closely rep resent real animal (or monster) skeletons. ∎

Many of the animations in this article are on the Web a http://www.dgp.utoronto/ca/DGP/Animations.html.

References

1. E. Muybridge, *Animals in Motion*, L.S. Brown, ed., Dove Publications, New York, 1957.
2. M. Girard, "Interactive Design of 3D Computer Animate Legged Animal Motion," *IEEE Computer Graphics and Appli cations*, Vol. 7, No. 6, June 1987, pp. 39-51.
3. N.I. Badler, B. Barsky, and D. Zeltzer, *Making Them Move* Morgan Kaufmann, San Francisco, Calif., 1991.
4. M.H. Raibert, "Trotting, Pacing, and Bounding by a Quadru ped Robot," *J. Biomechanics*, Vol. 23, No. 1, 1990, pp. 79-58
5. M.H. Raibert and J.K. Hodgins, "Animation of Dynami Legged Locomotion," *Computer Graphics* (Proc. Siggraph) Vol. 25, No. 4, July 1991, pp. 349-358.
6. M. McKenna and D. Zeltzer, "Dynamic Simulation o Autonomous Legged Locomotion," *Computer Graphic*



19 Synthesis of a leap. (a) Regular hopping gait for Luxo—for clarity, only the motion of the middle link is shown. (b) Result of a leaping motion derived from (a) through optimization. (c) Details of the anticipation and recovery involved. (d) A smaller leap, obtained by interpolating between the control used for (a) and (c).

(Proc. Siggraph), Vol. 22, No. 4, Aug. 1990, pp. 29-38.
7. M.G. Pandy, F.C. Anderson, and D.G. Hull, "A Parameter Optimization Approach for the Optimal Control of Large-Scale Musculoskeletal Systems," *Trans. ASME*, Vol. 114, Nov. 1992, pp. 450-460.
8. J.T. Ngo and J. Marks, "Spacetime Constraints Revisited," *Proc. Siggraph 93*, ACM, New York, 1993, pp. 343-350.
9. M. van de Panne and E. Fiume, "Sensor-Actuator Networks, *Proc. Siggraph 93*, ACM, New York, 1993, pp. 335-342.
10. M. van de Panne, R. Kim, and E. Fiume, "Virtual Windup Toys for Animation," *Proc. Graphics Interface 94*, Morgan Kaufmann Publishers, Palo Alto, Calif., 1994, pp. 208-215.
11. J.K. Hodgins, P.K. Sweeney, and D.G. Lawrence, "Generating Natural-looking Motion for Computer Animation," *Proc. Graphics Interface 92*, Morgan Kaufmann Publishers, Palo Alto, Calif., 1992, pp. 265-272.
12. *SD/FAST Reference Manual*, Symbolic Dynamics, Mountain View, Calif.

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

Readers may contact the author at the Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada, M5S 1A4, e-mail van@dgp.utoronto. ca.

CG Applications

Animating Gaits and Locomotion

Linda World, *Associate Editor*

Gaits and locomotion are the subject of interdisciplinary research, ranging from kinesiology (the study of muscles and their movements) to robotics.

Gait analysis plays a particularly important role in the investigation of human movement disabilities. Readers with Web access can visit an elegant site on gait analysis at the University of Virginia's Motion Analysis Laboratory in the Kluge Children's Rehabilitation Center (http://www.med. Virginia.EDU:80/medcntr/gaitlab). KCRC specializes in ameliorating the effects of cerebral palsy. The lab uses a variety of 3D systems to study walking patterns and to gather data on joint kinematics and kinetics, muscle activity and timing patterns, and muscle strength. Researchers, surgeons, and clinicians use the data to document deviations from normal gait patterns; to plan surgery, therapy, or bracing; and to evaluate the effects of intervention.

Robotics and computer graphics approach the study of gaits and locomotion from a different direction, namely, the *simulation* of these phenomena in dynamic system models. Students in Cornell University's Human Power, Biomechanics, and Robotics Laboratory (http://tam.cornell.edu/ programs/humanpower/) have developed a "passive dynamic walking" simulation and robot, driven entirely by gravity down a shallow slope. However, most work in simulation addresses the control algorithms that generate stable gaits, as does the work of both van de Panne and Ko and Badler in this issue.

In graphics, the simulation of gaits and locomotion is part of a larger project to simulate living creatures. As Ko and Badler point out, physically accurate gaits are not the equivalent of realistic looking gaits. Creating interactive human agents that behave realistically is one of the research goals of the University of Pennsylvania's Center for Human Modeling and Simulation (http://www.cis.upenn.edu/ ~hms/). The Center's Jack software, under development since the late 1970s, is a commercially available, 3D interactive environment for controlling articulated figures. Jack is "the virtual employee of choice" in human factors applications, according to Norman Badler, Jack's originator and the Center's director. In fact, Jack appeared on the centerfold of the 1991 annual report for heavy-equipment manufacturer John Deere.

The Life Forms package resulted from research at Simon Fraser University (http://fas.sfu.ca/css/groups/lifeforms. html). Also commercially available, Life Forms emphasizes the precise choreography of human motion and employs existing motion libraries as well as more complex parameterized motions.

Human simulation is the major focus of the Thalmann research group at the Ecole Polytechnique Fédérale de Lausanne (http://ligwww.epfl.ch/~thalmann/research. html), whose animation of Marilyn Monroe is almost as famous in computer graphics as the Utah teapot.

For an interesting comparison of simulated motions to real motions, visit the animation lab at Georgia Tech (http://www.cc.gatech.edu/gvu/animation/) and take an informal "Turing Test," where you must attempt to distinguish between real and simulated motions.

The tools of motion capture/analysis and motion simulation work together in the proprietary software of Biomechanics, an R&D company in Marietta, Georgia, that provides technology used in the production of video games, as well as medical, sports, and robotics applications (http://www.crl.com/~biomech/). Although the company's original product focused on 3D motion analysis, they also developed physics-based tools to enhance the visualization of the rapid transitions from one situation to another that occur in interactive games. More recently, Biomechanics applied these tools in a collaboration with Acclaim Entertainment for special effects in the movie *Batman Forever*. They wanted to keep the subtle complexities of human movement while allowing Batman to jump from a tall building and land gracefully—not something easily picked up with a motion-capture system. ■