Animation of Dynamic Legged Locomotion

Marc H. Raibert
MIT Leg Laboratory

Jessica K. Hodgins
IBM Watson Research Center

1 Abstract

This paper is about the use of control algorithms to animate dynamic legged locomotion. Control could free the animator from specifying the details of joint and limb motion while producing both physically realistic and natural-looking results. We implemented computer animations of a biped robot, a quadruped robot, and a kangaroo. Each creature was modeled as a linked set of rigid bodies with compliant actuators at its joints. Control algorithms regulated the running speed, organized use of the legs, and maintained balance. All motions were generated by numerically integrating equations of motion derived from the physical models. The resulting behavior included running at various speeds, traveling with several gaits (run, trot, bound, gallop, and hop), jumping, and traversing simple paths. Whereas the use of control permitted a variety of physically realistic animated behavior to be generated with limited human intervention, the process of designing the control algorithms was not automated: the algorithms were "tweaked" and adjusted for each new creature.

Key Words and Phrases: computer animation, motion control, legged locomotion, robotics, dynamical simulation, physically realistic modeling.

2 Introduction

An important goal of computer graphics is to generate physically realistic animation of *actuated systems*. Actuated systems are those that use muscles, motors, or some other kind of actuator to convert stored energy into time-varying forces that act within the system's mechanical structure. Animals, robots, and vehicles are examples of actuated systems. Actuated systems can create their own motions when asked to perform a task, often without help from an outside agent. We distinguish actuated systems from passive physical objects: both can move with physical realism, but only actuated systems can power and regulate their own motions.

A key step in animating actuated systems is to formulate control algorithms that transform expressions of desired behavior into detailed actuator control signals that produce the necessary motion. This step can be quite challenging because the relationship between task and motion is usually indirect. Desired behavior is typically expressed at a time scale and in a coordinate system associated with the task, whereas actuator control signals operate in the coordinate system and at the time scale of the mechanical system. For example, the desired behavior "Run forward at 2 m/s using a trotting gait" does little to specify how the hip joint on leg 2 should move at various times throughout the locomotion cycle. In legged locomotion the transformation from task specification to actuator specification is central, in that motions of the legs and feet are only intermittently related to the basic functional goals of providing support, stability, and propulsion.

A second reason that control of actuated systems is challenging is the presence of significant system dynamics. In dynamic systems the forces and torques exerted by the actuators on the mechanism are just one of the factors that influence the movement. Energy stored, recovered, and exchanged among the various mechanical components of the system and external forces influence the present and future motion of the system. The control algorithms must anticipate the response to actuation in the context of the ongoing activity. In a fast-moving legged system, for example, kinetic energy stored in the rotation of the leg can be large compared to the energy immediately available from the hip actuators. If the control algorithms are to swing the leg forward soon enough to place the foot for the next step, they must begin reversing the leg's motion early in the cycle. Each mass, moment of inertia, and compliant element in the system stores energy that might influence behavior. In most cases it is not correct to think of the control as providing "commands" to the mechanism through the actuators. The control inputs are more like "suggestions" that must be reconciled with the dynamic state and structure of the system.

Whereas the difficulty of achieving control of dynamic systems poses certain problems, the system dynamics also present opportunities. For instance, the

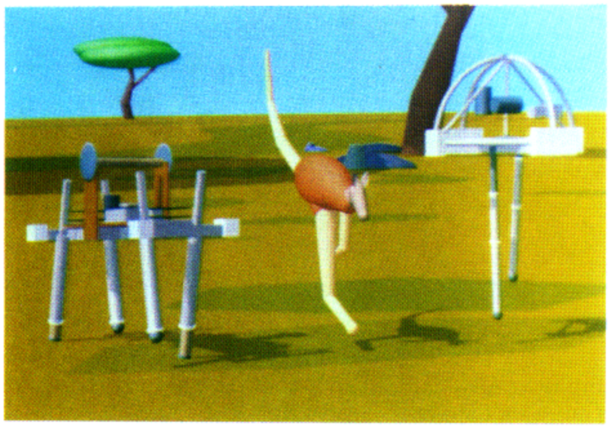


Figure 1: Biped, quadruped, and kangaroo models used to study control of running.

motions of dynamic systems are not limited by the instantaneous power available from the actuators. Stored energy can be used to generate motion. The difference achieved in the standing long jump vs. the running long jump is an example, (about 3.8 m vs. 8.5 m). The dynamics of a system can also contribute to energetic efficiency. Most running animals use some of the energy from one step to power the next, by temporarily storing it in stretched tendons and ligaments. The kangaroo achieves a significant energy savings with this trick [2].

This paper describes efforts to use control for animation of dynamic legged locomotion. We have restricted attention to fairly low-level desired behaviors, such as the speed and path of travel, posture, gait, and gait transitions. The starting point was our previous work on the control of one-, two-, and four-leg laboratory robots. Considered collectively, these robots ran at specified speeds, ran fast (13 mph), ran with several gaits, changed gait during running, jumped, climbed a flight of 3 stairs, and performed rudimentary gymnastic maneuvers [7, 8, 9, 12, 13]. We have adapted these robot control algorithms for animation of a biped that runs, gallops, and follows simple paths, a quadruped that trots, bounds, gallops, changes gait, and turns, and a planar one-legged kangaroo that hops and jumps. The creature models are shown in Figure 1.

Despite several variations, the control algorithms for the three models share a common set of basic elements. The common elements include a symmetry principle used for balance, decomposition of the control algorithms into separate parts for regulating hopping, speed, and posture, and the use of elastic energy storage in the legs. One might characterize these common elements of the control as a tool box for handcrafting control algorithms for new creatures with reasonable effort. At the moment, the control algorithms for each new creature or behavior require adjustment and tuning. For instance, the control algorithms that make the kangaroo run at a range of speeds had to be adjusted manually before they could maintain balance when the kangaroo took a big jump. We look forward to more automation of the control design process.

Once the control algorithms are implemented, behavior of each model is found by numerically integrating its equations of motion while the control algorithms monitor progress of the behavior and apply actuator forces. The animator specifies input to the control algorithms, but does not manipulate the model or its output directly. The resulting behavior was found to be qualitatively and quantitatively similar to that of the systems being modeled.

The next section describes previous work on the use of control in animation of legged locomotion. Then we describe the basic elements of the locomotion control algorithms and the models we used for testing. We close with results and a discussion.

3 Background

The opportunity to use control in computer animation has been recognized for about ten years. Progress has been slow because dynamical systems complex enough to be of interest are difficult to construct, computationally expensive to simulate, and difficult to stabilize and control. Researchers have taken a variety of approaches to simplifying the models. The trick is to simplify the model and the control problem sufficiently that animation is computationally and intellectually tractable, while retaining the underlying dynamics and realistic results.

Wilhelms and her colleagues implemented several systems that allowed the animator to explore various techniques for control of dynamic systems [16, 17]. The user selects which links are modeled kinematically and which have full dynamics. The system provides several low-level control options ranging from position control with springs and dampers to a mode which attempts to respond to gravitational forces with external forces to maintain balance. Joint limits and ground reaction forces are modeled with springs and dampers. One result of this work was the recognition that inverse dynamics is limited as a tool for computer graphics. Inverse dynamics is good for transforming detailed motion trajectories into force functions, once the detailed motions are known. However, the difficult part of the problem is finding the desired motions, and knowledge of the forces is not of vital interest in computer graphics, once the motion trajectories are known.

Bruderlin and Calvert used a dynamic model and control system to generate the leg motions for a human walking figure [3]. They used a telescoping leg with two degrees of freedom as the leg model for the stance phase and a compound pendulum model for the swing phase. They controlled walking using techniques adapted from biomechanics, which focused on synergy and a hierarchy of motor programs. Once the walking motion was calculated, a foot and upper body with arms were added to the model kinematically. These extra degrees of freedom were made to move in an oscillatory pattern similar to the pattern observed in humans. Bruderlin identified several key parameters of the walking motion and allowed the animator to change them. The details of the complete walking motion were generated automatically by the control system in concert with the dynamic model.

McKenna and Zeltzer's work on an animated cockroach fully embraced the idea that numerical integration of a dynamic model could be used to generate all motions of an animated creature, and that the control algorithms could influence behavior only through forces exerted by the actuators [10]. They implemented a dynamical model of the cockroach, and relied on a control system to pattern its motion. Their cockroach had springy legs, so the load of the body was distributed on the support legs. The walking algorithms they used were based on motion patterns that have been observed in insect locomotion. McKenna and Zeltzer's work is closely related to our own, in that we too rely on behavior of the dynamical model for all motion generation and restrict human intervention to specifying desired behavior to the control. Our work differs from theirs in the sort of locomotion studied and the nature of the control algorithms: they concentrated on statically stable multi-legged walking, while we focus on running and jumping with a ballistic flight phase, and on the role of the springy leg in generating the running cycle.

Optimization techniques and modern control theory offer the hope of automatically producing control systems by specifying task constraints or optimization functions. Witkin and Kass used their "spacetime" approach to produce a remarkable animation of a dynamic lamp [18]. Panne, Fiume, Vranesic used techniques from modern control theory to allow the lamp to perform a flip [15]. The potential generality of these approaches and their ability to deal with anticipation makes them among the most interesting new methods for animation of dynamic systems. The potential liability is the growth of the search spaces when applied to more complex systems.

Girard and Maciejewski do not use numerical integration of physical models, but rely instead on rules associated with dynamics [4, 5]. For instance, they programmed a sinusoidal vertical motion of the body to approximate the motion of a massful body bouncing on springy legs. They coordinated the joints of their human figures to keep the center of mass over the support feet, as required for balance. These techniques resulted in some of the best looking animation of legged locomotion that we have seen.

4 Animation, Control, and Modeling

Figure 2 shows the general process we use for animation. The user provides the control system with information about the desired animated behavior, such as speed, gait, path, etc. The user also initializes the legged model by placing it in a particular state. Once the animation is started, the control algorithms are responsible for stabilizing posture, maintaining the locomotion cycle, controlling speed and direction of travel, and regulating the behavior of the joints. Because the control is able to coordinate the lower levels of behavior for a task, the animator is free from direct involvement in specifying the joint torques or the details of the actual movements.

The three legged models are shown in Figure 1. Two of the models are patterned after physical robots that we built and use for laboratory experiments. One robot model is of a biped with telescoping legs and ball-joint hips. The other robot model is of a quadruped with telescoping legs and gimbal hips. The third model is a simplified version of a kangaroo. It is simplified in that it is planar, has one leg and arm instead of two, and it has fewer links in the tail than the animal. A total of six gaits were implemented and tested: biped running and galloping, quadruped trotting, bounding, and galloping, and kangaroo hopping. All of these gaits are technically classified as running, because they include at least one flight phase per cycle, a period when all feet leave the ground at the same time.

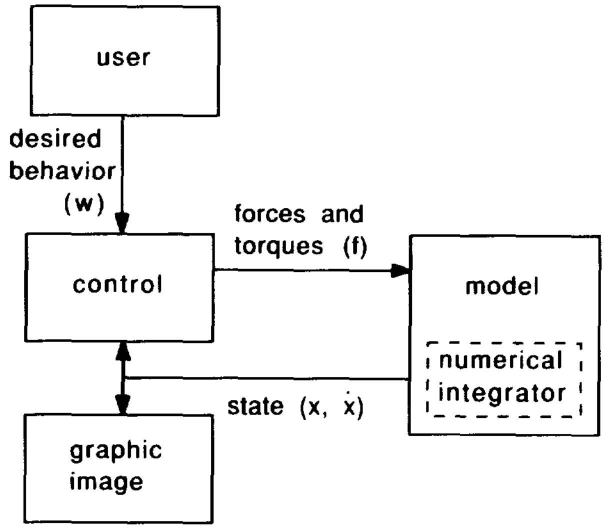


Figure 2: Block diagram of animation process. The model consists of equations of motion for the rigid bodies of the legged system, actuator and sensor models, force equations for ground-interaction, and a numerical integrator that produces motion as a function of time. The model calculates its behavior once every integration interval, 0.0004 s. The control calculates the force or torque to be exerted by each actuator based on the current state of the model, $\mathbf{X}, \dot{\mathbf{X}}$ and the user input $\mathbf{W}$. The control calculation is done every control interval, which is usually a few ms. The human animator specifies desired behavior $\mathbf{W}$, which consists of desired running speed, the path along which to travel, and any event information, such as when and how high to jump. The animator specifies his or her input before the animation process begins. In the current implementation, the animator must also initialize the state of the model.

Control

In this section we describe the control algorithms used for animation of running. A control system for running must perform three primary functions

- cause the legs to step, exchanging support,
- provide balance to regulate the running speed, and
- maintain the body in an upright posture.

These three functions can be called *hopping*, *speed control*, and *posture control*.

Hopping Control

An idea that developed in biomechanics over the last fifteen years is that animals use elastic structures in their limbs to improve the energetic efficiency of their locomotion. Tendons and ligaments in the legs and feet stretch during each collision with the ground, converting some the system's kinetic energy into elastic strain energy. The stored energy is returned during the next step, when the elastic structures rebound. A significant fraction of the total running energy, perhaps 20% to 40%, recirculates from one step to the next, without needing resupply from the muscles. Kangaroos use their substantial Achilles tendons to perform this energy recovery function whereas Alexander argues that humans store energy in their Achilles tendons and the ligaments that support the arch of the foot [1]. Compliant legs and feet also reduce peak loads that occur in running when the feet strike the ground at the end of each flight phase [11, 1].

We use compliance in the legs to produce the vertical oscillations needed in running. The control algorithms allow the mass of the body to rebound on the springy leg during ground collisions and to be drawn back to earth by gravity during the flight phase. The biped and quadruped legs were made springy with spring-damper actuator models for the telescoping joint. The kangaroo leg was made springy by modeling the ankle actuator as a torsional spring-damper with adjustable rest length. Both actuator models have the form

$$f = k(x - x_r) + b\dot{x} \qquad (1)$$

where f is the actuator force, k is the spring constant, b the damping constant, x the spring length, and x_r the spring rest length.

Control of the spring rest length is used to inject or remove energy from the system in order to initiate the oscillation, modulate it, or stop it. For vertical hopping with a massless leg, the altitude of a particular hop is predicted by the sum of the potential strain energy in the leg spring, the potential energy of elevation of the system mass, and the kinetic energy due to motion of the body

$$h = (PE_{strain} + PE_{elevation} + KE)/Mg \qquad (2)$$

where h is the expected altitude of the hop, M is the system mass, and g is the acceleration of gravity. The control system can inject or remove energy to influence this outcome. This hopping control mechanism takes advantage of the dynamic interaction between the mechanical system and the control to generate the motion. No trajectory is specified.

Speed Control

Legged systems are like inverted pendulums: they tip and accelerate whenever the point of support is displaced from the projection of the center of mass [6]. If the average point of support is kept under the average location of the center of mass, the system may tip for short periods, without tipping over entirely. One way to achieve such a balancing relationship between the feet and the center of mass is to move the body in a symmetric fashion over the supporting feet during each support period. When the control system places the foot to obtain a symmetric sweeping pattern, the forward speed will remain the same at liftoff as it was at touchdown. We call this position of the foot the *neutral point*. When the control system displaces the foot from the neutral point, the body accelerates, with the magnitude and direction of acceleration related to the magnitude and direction of the displacement, as shown in figure 3. The control system displaces the foot from the neutral point by a distance proportional to the difference between the actual speed and the desired speed. The control system computes the desired foot position as:

$$x_{fh,d} = \frac{\dot{x}T_s}{2} + k_{\dot{x}}(\dot{x} - \dot{x}_d) \qquad (3)$$

where $x_{fh,d}$ is the forward displacement of the foot from the projection of the center of gravity, $\dot{x}$ is the forward speed, $\dot{x}_d$ is the desired forward speed, T_s is the predicted duration of the next support period, and $k_{\dot{x}}$ is a gain. The first term of equation 3 is an estimate of the neutral point and the second term is a correction for any error in forward speed or for a desired acceleration. The duration of the next support period is predicted to be the same as the measured duration of the previous support period. After the control system finds $x_{fh,d}$, a kinematic transformation determines the joint angles that will position the foot as specified.

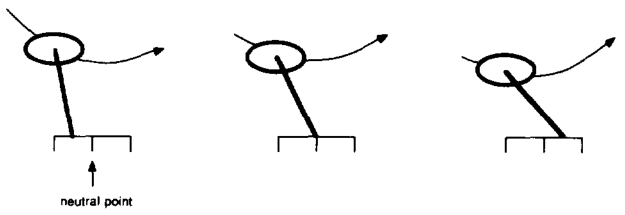
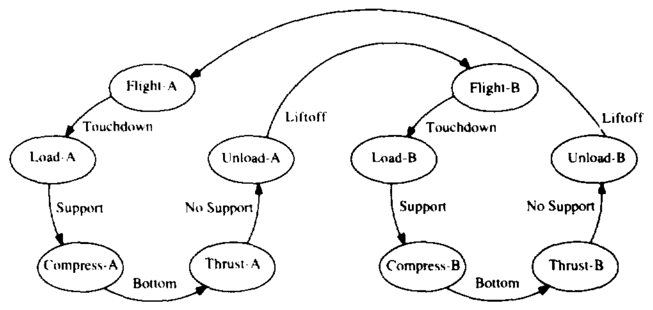


Figure 3: When the foot is positioned at the neutral point, the body travels along a symmetric path that leaves the system unaccelerated in the forward direction. Displacement of the foot from the neutral point accelerates the body by skewing the symmetry of the body's trajectory. When the foot is placed closer to the hip than the neutral point, the body accelerates forward during stance and the forward speed at liftoff is higher than the forward speed at touchdown (left). When the foot is placed further from the hip than the neutral point, the body decelerates during stance and the forward speed at liftoff is slower than the forward speed at touchdown (right). Horizontal lines under each figure indicate the distance the body travels during stance, and the curved lines indicate the path of the body.



State	Actions
FLIGHT	
Active leg leaves ground	Interchange active, idle legs Lengthen active leg for landing Position active leg for landing Shorten idle leg
LOADING	
Active leg touches ground	Zero active hip torque Keep idle leg short
COMPRESSION	
Active leg spring shortens	Servo pitch with active hip Keep idle leg short
THRUST	
Active leg spring lengthens	Extend active leg Servo pitch with active hip Keep idle leg short
UNLOADING	
Active leg spring approaches full length	Shorten active leg Zero hip torques active leg Keep idle leg short

Figure 4: Finite state machine that coordinates running. The state shown in the left column is entered when the sensory event, listed just below the state name, occurs. Actions are listed on the right. The controller advances through the states in sequence. The diagram is for a two-legged gait.

Posture Control

Depending on the number of legs, the gait, and whether there is a tail, the trunk may pitch and roll during running. The long-term attitude of the trunk must be stabilized if the system is to remain upright. The control system we implemented regulates the orientation of the trunk by applying torques to the body during the support phase. In the biped and quadruped models, the hip actuators are used to apply the torques required for attitude control. In the kangaroo model, the knee is used to perform this function. Vertical loading on the feet keeps the leg from slipping when the torque is applied. The posture control torques are generated by a linear servo:

$$\tau = -k_p(\phi - \phi_d) - k_v(\dot{\phi}) \qquad (4)$$

where τ is the leg torque, ϕ is the angle of the body, ϕ_d is the desired angle of the body, $\dot{\phi}$ is the angular rate of the body, and k_p, k_v are gains.

The control systems for running use separate algorithms for stabilizing hopping, forward speed, and posture of the trunk. Each of these parts of the control acts independently, as though it influences just one component of the behavior. Interactions due to imperfect decoupling are treated as disturbances. This decoupling simplifies the control implementation.

In addition to the control algorithms described so far, each implementation uses a finite state machine to track the ongoing behavior of the model, to synchronize the control actions to the running behavior, and to do some bookkeeping. Figure 4 shows a state machine for the biped.

Gaits

We implemented a total of six gaits: biped running and galloping; quadruped trotting, bounding, and galloping; and kangaroo hopping. The running algorithms for all six gaits are based on control originally developed for one-legged hopping. For each gait we tailored the state machine to cycle through the legs in the correct order and to invoke suitable versions of the algorithms that distribute the load among the support legs.

Bipedal running is like one-legged hopping, except there is an extra *idle* leg, in addition to the active leg. The idle leg is kept short and out of the way while the active leg performs the functions described earlier to control forward speed, hopping height, and balance. The state machine for bipedal running is shown in figure 4.

In quadruped trotting and bounding, the legs are coordinated to work together in pairs. The coordination we used makes each pair of legs act collectively like a single leg, called a *virtual leg*. The members of each pair strike the ground in unison and leave the ground in unison. Diagonal legs form pairs in trotting and front legs and rear legs form pairs in bounding. One can think of these quadruped gaits as *virtual biped gaits*, with the active pair of legs providing support while the idle pair swings forward in preparation for the next step. The higher levels of the control system ignore the individual physical legs, pretending to do biped control on the two virtual legs as described earlier.

Quadruped galloping is similar to bounding except that the legs of the front and rear pairs no longer strike and leave the ground in perfect unison. The stance phase is composed of a single support phase, a double support phase, and then a second single support phase. The legs are positioned on the ground with a separation both in time and space. The stance phase is extended and the legs of each pair share the work of rebounding the body. Biped galloping is similar to quadruped galloping in that the two stance legs share a single support phase. We implemented two styles of biped galloping. In one style the legs swing forward together during the flight phase. In the other style they swing forward independently during the other leg's single support phase. The first style produces a motion that is similar to the front half of a galloping horse while the second is closer to the pattern used by galloping humans. Pitching of the body in response to swinging of the legs is greatly reduced in the second form of galloping.

Kangaroo control

Control algorithms for the kangaroo were essentially the same as for the robot models, with additional provisions for coordinating the joints of the articulated leg and for moving the tail. Kangaroos have legs with rotary joints. In the kangaroo model we eliminated the toe joint, leaving an ankle, a knee, and a hip, all of which have axes perpendicular to the sagittal plane. We made several decisions that constrained the behavior of the leg and allowed us to program it using methods originally developed for robot telescoping legs.

We decided to use the ankle joint as the primary energy storage element in the leg. We assumed that the ankle actuator consisted of a spring-damper mechanism with an adjustable zero spring length. This mechanism models a muscle acting in series with a springy Achilles tendon. We adjusted the spring and damper characteristics so that a significant fraction of the energy stored in the spring during leg compression was returned during leg extension.

We decided to configure the leg so the ground reaction force generated during hopping passes approximately through the knee. This configuration minimizes the moment required at the knee to resist support and thrust forces. In balanced running, the ground reaction forces act along a line passing from the toe through the system center of mass. The control system servoes the hip joint to keep the knee on this thrust line during stance.

Because torque about the knee is not needed to support the body, we use the knee to maintain the body in a level posture. The linear servo given in equation (4) operates at the knee during the stance phase to eliminate errors in body orientation and orientation rate.

The tail is made to counteroscillate with the leg, keeping the angular momentum of the entire system near zero throughout the running cycle. When the leg strokes backward during the stance phase, the tail strokes downward. The tail motion is produced by making a step change in the spring rest length for the actuator at the base joint of the tail. The spring damper characteristics of the joint is tuned to oscillate in period with the running motion. The two peripheral joints in the tail are actuated by a spring damper with fixed rest length. The head was servoed to stay level throughout the running motion.

Modeling

Each of the legged systems was modeled as a tree of rigid bodies, each connected to its parent by rotary or sliding joints. The mass, mass center, and moment of inertia

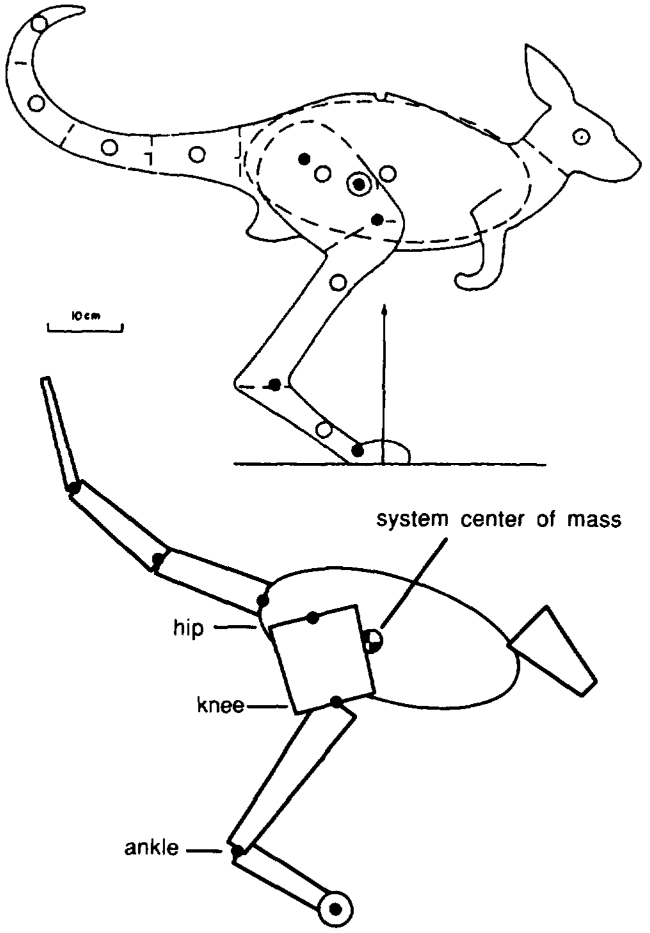


Figure 5: Top) Drawing of real kangaroo used as the basis of the kangaroo model. It was a juvenile red kangaroo weighing 6.6 kg. Drawing is from [2]. Bottom) Diagram of kangaroo model. Except for the trunk, each link was modeled as the frustum of a cone, with pivots at the base and tip. The two legs of the real kangaroo were combined into one model leg. All dimensions were chosen to match the real kangaroo in link length and link mass, assuming the kangaroo was the density of water. Mass centers and moments of inertia were calculated from the geometric model.

for each body were determined in one of two ways. For the robot models we used actual measurements of the mass properties. For the kangaroo model we used the link length, mass, and mass center data given in [2]. We calculated the moments of inertia of the links from the geometry used in the graphics, typically assuming the density of water.

An actuator capable of exerting forces or torques was located at each joint. The lowest level of control used linear servo mechanisms to specify actuator forces:

$$f = -k_p(\theta - \theta_d) - k_v\dot{\theta} \qquad (5)$$

where f is the force or torque acting on the joint, θ is the joint angle or length, and k_p and k_v are position and velocity feedback gains. These joint servos have the same dynamical behavior as a spring-damper mechanism with programmable rest length. Depending on one's point of view and the design of the control system, these joint servos can be regarded as part of the control system or as part of the model.

Environmental interaction was restricted to gravitational forces and ground contact forces. A single point on each foot could make contact with the ground. The ground contact model for each foot consisted of four spring and damper sets: one vertical, two tangent to the surface, and one torsional about the ground surface normal. The rest length of each spring was reset when a foot first touched the ground during a support period. The ground contact compliance represents the compliance of the "paw pad", the elastic elements on the bottom of the feet, plus any compliance provided by the support surface. The kangaroo model used a non-linear paw pad spring in the vertical direction:

$$f_z = k_{g_z} \frac{k_r}{k_r + z - 1} \qquad \text{for } z < 0 \qquad (6)$$

where f_z is the vertical spring force, k_{g_z} is the paw pad stiffness, k_r is the paw pad thickness, and z is the altitude of the foot contact point above the ground. We chose a non-linear spring for this part of the model because it is consistent with compression of an elastic material between two surfaces, it reduced the maximum deflection during the support period, and it allowed vertical forces to develop more slowly at initial impact.

We assume that once ground contact is made, there is no slipping between the foot and the ground. This is equivalent to an infinite coefficient of friction. To test this assumption we used data from typical runs to calculate the coefficient of friction that would have prevented slipping. For the biped, this value was always less than 1. With the exception of the very beginning and end of the support period, a coefficient of friction of about 0.5 would have prevented slipping for the quadruped and kangaroo. At the very beginning of the support period, when the feet begin to make contact with the ground, however, there is a period of up to 10 ms during which the coefficient of friction would have had to be almost 2.0 to prevent slipping. A similar period occurred at the very end of the stance phase. On a day without oil leaks, the coefficient of friction between a robot foot and the floor of our laboratory is about 1.0.

Equations of motion were generated for the structure with a commercially available program [14]. The program generates efficient subroutines ($O(n)$ where n is the number of links) that implement the equations of motion using a variant of Kane's method and a symbolic simplification phase. The equations of motion were numerically integrated using Euler's method, with time steps of about 0.0004 s. Simulations of a single creature ran between 7 and 10 times slower than real time on a SUN Sparc2.

Dynamic Scaling

We used the basic principles of allometry to scale the size of a model, along with its control system and movements. Adult animals of a single species generally scale uniformly in all linear dimensions, and thereby maintain their proportions [11]. For a system scaled in this fashion, Table 1 gives rules for scaling the control system and the motions. Suppose we want to animate a kangaroo that is L times bigger than normal. Using the table we see that it will hop L times as high, travel $\sqrt{L}$ as fast, and have a cadence $1/\sqrt{L}$ of the original cadence.

There are two ways to use these scaling rules. One way is to generate a new model of the creature and a new control system, based on the scaled mechanical parameters available from the table. Behavior of the scaled model can be used directly. An alternative is to implement a single model and control system at scale 1, but to scale all input to the animation as a function of $1/L$, and all output as a function of L. For example, if $L = 2$, initial positions would be scaled by $L^{-1} = 1/2$, initial times and the integration time step would be scaled by $L^{-1/2} = 1/\sqrt{2}$, and desired running speed would be scaled by $L^{-1/2} = 1/\sqrt{2}$. The animation could then be run at scale 1. The outputs are scaled before displayed: positions and geometry by $L = 2$ and time by $L^{1/2} = \sqrt{2}$. We used the latter approach, but both give identical results.

Quantity	Units	Scale Factor
Basic variables		
length	L	L
time	T	$L^{1/2}$
force	F	L^3
torque	FL	L^4
Motion variables		
displacement	L	L
velocity	LT^{-1}	$L^{1/2}$
acceleration	LT^{-2}	1
angular displacement	-	1
angular velocity	T^{-1}	$L^{-1/2}$
angular acceleration	T^{-2}	L^{-1}
Mechanical parameters		
mass	$FL^{-1}T^2$	L^3
stiffness	FL^{-1}	L^2
damping	$FL^{-1}T$	$L^{5/2}$
moment of inertia	FLT^2	L^5
torsional stiffness	FL	L^4
torsional damping	FLT	$L^{9/2}$

Table 1: Scaling rules that preserve geometric similarity. If a system is scaled in size by a factor L and its mechanical parameters are each scaled according to the table, then the motion of the scaled system can be found from the motion of the original unscaled system. The table is derived assuming uniform scaling in all dimensions (geometric similarity), and that the acceleration of gravity is invariant to scale.

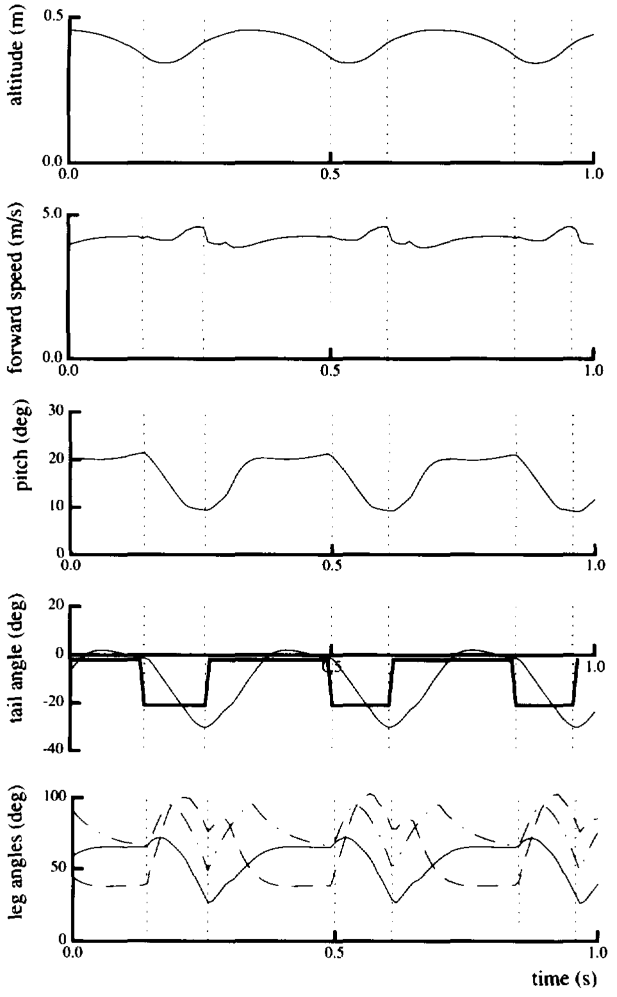


Figure 6: Data recorded from planar kangaroo model during three steps of running with a desired speed of 5 m/s. The vertical dashed lines bracket the stance phase. The leg joint angles are defined in Figure 5. Key to joints in leg angle plot: (solid) hip, (dashed) knee, (dot-dashed) ankle.

5 Results

Table 2 compares the behavior of each animated runner to the physical system it is designed to model. The data indicate that there are many similarities. For example, the extended flight phase is shorter than the gathered flight phase for both the physical and the animated quadruped during bounding. The kangaroo's body and tail oscillate as it runs and the magnitude of the oscillations are similar. The table also illustrates a number of differences. For instance, the gathered flight phase of the animated quadruped is twice that of the robot quadruped. The animated biped spent a great deal more time in flight than the robot.

Another difference between the animated and real kangaroo is in the behavior of the feet at impact. The real kangaroo accelerates its feet to the speed of the ground just before they touch, so they do not scuff or have a tangential impact. We call this *ground speed*

Quantity	Robot/ Animal	Animation
Quadruped Bound		
pitch magnitude (deg)	26.4	25.8
stride length (m)	1.15	1.22
stride duration (s)	0.37	0.43
gathered flight phase duration (s)	0.11	0.22
extended flight phase duration (s)	0.04	0.03
stance duration, front legs (s)	0.10	0.10
stance duration, rear legs (s)	0.12	0.08
change of leg length during support (m)	0.085	0.092
running speed (m/s)	2.9	2.8
error in running speed (m/s)	1.25	1.2
Quadruped Trot		
pitch magnitude (deg)	2.2	6.2
stride length (m)	1.84	1.49
stride duration (s)	0.80	0.54
flight duration (s)	0.27	0.18
stance duration (s)	0.12	0.09
change of leg length during support (m)	0.04	0.03
desired running speed (m/s)	2.3	2.3
error in running speed (m/s)	0.7	0.2
Biped Run		
pitch magnitude (deg)	2.0	3.4
roll magnitude (deg)	6.7	11.5
stride length (m)	0.57	0.74
stride duration (s)	0.36	0.47
flight duration (s)	0.18	0.29
stance duration (s)	0.18	0.18
change of leg length during support (m)	0.01	0.04
error in running speed (m/s)	-0.3	0.2
Kangaroo Hop		
peak vertical acceleration (g)	5	4.6
pitch magnitude (deg)	10	12
magnitude of tail wag relative to trunk (deg)	30	31
stride length (m)	2.2	1.5
stride duration (s)	0.35	0.35
flight duration (s)	0.25	0.24
stance duration (s)	0.10	0.11
desired running speed (m/s)	6.2	5.0

Table 2: Comparison between behavior of physical robot and animation, and between real and animated kangaroo. The runs were selected to match running speed as closely as possible, so running speed should not be used for comparison. The kangaroo data are from [2], and the quadruped robot data are from [13].

matching. The animated kangaroo does not do ground speed matching.

The control algorithms were successful in providing balanced running, regulating the speed of travel to within about 10% of the desired value, and in steering the creatures along specified paths. To get each new creature or gait working required some adjustment of the control parameters. For example, to improve the appearance of the quadruped trotting and bounding motions, we reduced the standing length of the legs. To get the kangaroo tail to oscillate in rhythm with the running motion, we adjusted the spring and damping constants of the tail joint servo until the natural frequency was about equal to the hopping frequency. To make the kangaroo jump over an obstacle, a number of additional states were added to the state machine. These states allowed stiffer operation of the legs for the jump, more dissipation in the legs during landing after the jump, and a number of cosmetic changes. Once adjusted, the locomotion proceeded without adjustment.

6 Discussion

The motions described in this paper are physically realistic in that they were generated by applying forces and torques to physical models of a mechanical system. The degree of physical realism depends on the degree to which the system is accurately modeled. For instance the mass parameters of the links, structural strength of the links, torque available from the actuators, actuator bandwidth, stiffness of the feet, and external friction are parameters that help determine the overall appearance of a motion.

There is no guarantee, however, that physically realistic motion will be "natural looking motion". It seems that animals move with a smoothness and coordination that is not required by physical realism alone. Constraints on smoothness, compliance, or energetic efficiency could eventually lead to uniformly natural looking behavior. We found that increasing the compliance of the actuators generally improved the appearance of the animated motions. We expect that constraints that lower the overall energy expenditure will also contribute to more natural looking locomotion.

As mentioned earlier, the methods described in this paper might be thought of as a tool box for handcrafting control systems for new creatures and behaviors. We expect that further development of control systems for computer animation will proceed in two steps. First, we expect control algorithms for individual creatures to become capable of a wider variety of behavior with less manual adjustment. Eventually, it should be possible to design control algorithms that will make creatures autonomous enough to "do what they are told". The animator should be able to direct the behavior at a relatively high level, and let the control system propel the system from one place to another at the desired rate along the specified path, use specified gaits, change between gaits, and maintain balance, all while adhering to physical realism. For well defined sets of creatures and behaviors, this goal is within sight.

Second, we think it is possible to automate the process of generating control algorithms for new creatures. Given control algorithms that work correctly for the locomotion of a horse, for example, it should be possible to automatically generate control algorithms for an antelope, dog, cat, or elephant. Initially such automatically generated control might be restricted to a limited repertoire of behavior. A first cut might aim at balanced running at a range of speeds with several gaits and transitions between gaits. It is difficult to predict how long it will take to achieve this level of control automatically, or to go beyond it to automate more complicated creatures or higher-level function.

One might expect the various workers involved in the study of control algorithms for legged locomotion animators, robot engineers, and biological scientists to differ in their criteria for successful algorithms. Such criteria could include precision of control, generality of control algorithms with respect to diverse behaviors and diverse creatures, the aesthetic appearance of the resulting movement, the simplicity and elegance of the solution, or the degree to which an algorithm explains the workings of animals. We might find, however, that the solutions that best explain animal behavior will be similar to those that produce the best robot behavior, and that the easiest way to make an animation look animal-like is to use a control system like the animal's. It is also possible that techniques successful in producing animations that are visually pleasing and natural will lead to a better understanding of the control at work in animals and to the construction of more effective robots.

7 Acknowledgements

We thank Tom McMahon for helping us appreciate the beauty of dynamic scaling laws, and Takashi Aoki and Don Floyd for contributing to the software used in this project. This research was supported by contracts from the DARPA Information Sciences and Technology Office and the Charles Stark Draper Laboratory.

8 References

1. Alexander, R. McN. 1988. *Elastic Mechanisms in Animal Movement* (Cambridge University Press: New York).

2. Alexander, R. McN., Vernon, A. 1975. The mechanics of hopping by kangaroos (Macropodidas). *J. Zoology (London)* 177:265–303.

3. Bruderlin, A., Calvert, T. W. 1989. Goal-Directed, Dynamic Animation of Human Walking. *Computer Graphics* 23(3):233–242.

4. Girard, M. 1987. Interactive design of 3-D computer animated legged animal motion. *IEEE Computer Graphics and Animation* June: 39–51.

5. Girard, M. and Maciejewski, A. A. 1985. Computational Modeling for the Computer Animation of Legged Figures. *Siggraph* 19(3): 263–270.

6. Hemami, H., Weimer, F. C., Koozekanani, S. H. 1973. Some aspects of the inverted pendulum problem for modeling of locomotion systems. *IEEE Trans. Automatic Control* AC-18:658–661.

7. Hodgins, J., Raibert, M. H. 1990. Biped Gymnastics. *International Journal of Robotics Research*, 9(2):115–132.

8. Hodgins, J., Raibert, M. H., 1991, Adjusting step length for rough terrain locomotion, *IEEE J. Robotics and Automation*, Sacramento.

9. Hodgins, J., Koechling, J., Raibert, M. H. 1985. Running experiments with a planar biped. *Third International Symposium on Robotics Research*, Cambridge: MIT Press.

10. McKenna, M. and Zeltzer, D. 1990. Dynamic Simulation of Autonomous Legged Locomotion. *Computer Graphics* 24(4):29–38.

11. McMahon, T. A. 1984. *Muscles, Reflexes, and Locomotion.* Princeton: Princeton University Press.

12. Raibert, M. H. 1985. *Legged Robots That Balance.* Cambridge: MIT Press.

13. Raibert, M. H., 1990. Trotting, pacing, and bounding by a quadruped robot, *J. Biomechanics*, Vol.23, Suppl.1, 79–98.

14. Rosenthal, D. E., Sherman, M. A., 1986. High performance multibody simulations via symbolic equation manipulation and Kane's method. *J. Astronautical Sciences* 34:3, 223–239.

15. van de Panne M., Fiume, E., Vranesic, Z. 1990. Reusable Motion Synthesis Using State-Space Controllers *Computer Graphics* 24(4): 225–234.

16. Wilhelms, J. 1986. Virya–A motion control editor for kinematic and dynamic animation. *Graphics Interface '86* 141–146.

17. Wilhelms, J. 1987. Using Dynamic Analysis for Realistic Animation of Articulated Bodies. *IEEE Computer Graphics and Animation* June: 12–27.

18. Witkin, A., Kass, M., 1988. Spacetime Constraints. *Computer Graphics* 22(4):159–168.

9 Appendix: Physical Parameters of Models

Link	Link length (m)	Mass center (m)	Mass (kg)	Moment of Inertia $(kg - m^2)$
Biped				
trunk			23.1	[.17 .17 .30]
upper leg	.20	.095	1.4	[.018 .017 .0014]
lower leg	.63	.22	.64	[.02 .02 .00018]
Hip location wrt trunk center of mass: $x = 0.0$, $y = \pm 0.072$, $z = 0.0$				
Quadruped				
trunk			10.0	[.54 2.35 2.39]
upper leg	.41	.2	1.5	[.043 .043 0]
lower leg	.4	.2	1.0	[.0035 .0035 0]
Hip location wrt trunk center of mass: $x = \pm 0.39$, $y = \pm 0.12$, $z = 0.0$				
Kangaroo				
trunk			3.67	.034
thigh	.13	.064	1.62	.0039
shin	.26	.105	.60	.0033
foot	.174	.082	.14	.00038
tail1	.166	.079	.24	.00058
tail2	.166	.071	.14	.00033
tail3	.166	.076	.069	.00016
head	.13	.04, .04	.33	.00046
Hip location wrt trunk mass center: [-.11 0]				
Head location wrt trunk mass center: [.21 0]				
Tail location wrt trunk mass center: [.2 0]				

Table 3: Physical parameters of models used in animations. Link lengths are from proximal joint to distal joint. Mass centers are distances from the proximal joint of a link to the mass center. Moments of inertia are about the mass center of the link. The diagonal of the moment of inertia tensor is given.