

# Autonomous Animation and Control of Four-Legged Animals

Evangelos Kokkevis, Dimitri Metaxas and Norman I. Badler  
Department of Computer and Information Science  
University of Pennsylvania, Philadelphia, PA 19104-6389

## Abstract

This paper develops a framework for the realistic autonomous animation and motion control of four-legged animals. Our system uses a combination of kinematics, dynamics and control theory. The input to our system, the velocity and heading of the animal, originates either from a simulated visual sensory system or a user. Based on this input we model walking, trotting, and simple behaviors such as target pursuit. We use a combination of kinematics, dynamics and control theory. A feedback controller, using the desired velocity and animal heading, computes the aggregate force and torque vector that should be applied to the body to achieve the given motion. This force and torque is distributed to the legs in contact with the ground through a linear programming algorithm. We then use forward dynamics to compute the actual body displacement. A kinematic gait controller is in charge of the stepping pattern. It arranges the stance and transfer phases of each leg depending on the current locomotion velocity, the turning rate and the ground conditions. Although we chose to focus on four-legged animals, the approach is generalizable to other multi-legged creatures or biped locomotion. Our motion system can currently simulate variable speed walking and trotting on flat or uneven terrain. Given its flexibility, the system can be used as a basis for more complex animations involving high level behaviors and interactions with other animals.

**Keywords:** Articulated Figures, Animation, Control Theory, Dynamics.

## 1 Introduction

An important open problem in computer graphics is the autonomous realistic animation of living organisms and their behaviors. Among the major challenges of such animations are the modeling and the choice of the organism's degrees of freedom and the design of a particular behavior given the underlying dynamic model. This paper presents a new approach towards the autonomous realistic animation and simple behavior simulation of multi-legged animals by combining

kinematics, dynamics, and control theory.

A variety of techniques have been used to address some of the problems within the general area of articulated figure locomotion, applied both to humans ([BC89], [Hod94], [vFV92], [Wil86], [WS89]) and legged animals ([Gir87], [GM85], [RH91], [MZ90]). However, each of the above techniques addresses only a subset of the problems associated with autonomous animation. In this paper, we develop a unified framework aimed to address in a formal way the autonomous locomotion in variable terrain, smooth gait transitions, and simple behaviors such as target pursuit. We combine model-based control theory, dynamics and kinematic gait controllers. The input to our system comes from either a simulated visual sensory system or a user. Even though we present results for multi-legged animals, our methodology is general and can be applied to biped locomotion as well. The method used in this paper is similar to the approach used by Park[Par73] for controlling legged vehicles.

Our system works in the following way: A dynamic feedback controller computes the force and torque that need to be applied to the body so that the animal maintains a desired velocity and heading. This force and torque are subsequently distributed to the legs through the use of a linear programming algorithm. Consequently, the ground exerts on the legs frictional and vertical forces that get transferred to the body and make the animal move. In order to ensure realistic walking and trotting animation at variable locomotion speeds, we develop a kinematic gait controller that controls the motion pattern of the legs. In the following sections we present the details of our technique and demonstrate it through examples involving dog animations.

## 2 The Animal Model

Although the control methods used in this paper are general and can be applied to a variety of animals with two or more legs, we decided to concentrate on four legged creatures. The model that we use to demonstrate our technique is that of a dog shown in Figure 1. Most legged animals have many degrees of

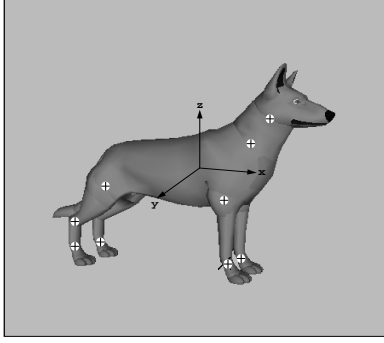


Figure 1: *The dog model. Joint positions are denoted by  $\oplus$*

freedom (DOF), primarily due to the flexibility of their body and the complexity of their joints. This makes it almost impossible to accurately control any artificial model, unless certain simplifying assumptions are made. In our case, we assume that the animal’s body is rigid. Several articulated parts are connected to the body with joints which are denoted in the figure by  $\oplus$ ’s. The parts are the four legs, the neck and head, and the tail. The front legs have two joints; the hip joint that connects them to the body and the ankle joint that connects the upper and lower parts of the leg. The rear legs have an extra joint that connects the paw to the leg. The hip joints have two rotational and two translational DOFs. For simplicity, the ankle and the paw joints have only one rotational DOF. Similarly, the neck and the tail are jointed to the body with a one DOF rotational joint. Finally, the head is connected to the neck with a two DOF rotational joint.

### 3 System Description

Our world consists of the four-legged animal and the terrain on which the animal is moving. The animal system can be further subdivided into the *body subsystem* and the *legs subsystem*. The state of the body subsystem,  $\mathbf{s}$ , can be completely described in terms of the following quantities with respect to a world coordinate system  $\Phi$ ,

$$\mathbf{s} = \{\mathbf{c}, \dot{\mathbf{c}}, \theta, \omega\}, \quad (1)$$

where  $\mathbf{c}$  and  $\dot{\mathbf{c}}$  are the position and velocity vectors of the body’s center of gravity,  $\theta$  is the orientation of the body expressed as Euler angles and  $\omega = \dot{\theta}$  is the angular velocity of the body. The leg subsystem consists of the four legs. Each leg can be described by its position, its stage in the gait cycle and its velocity or force it exerts on the ground. We chose to treat the legs dynamically only when they are in contact with the

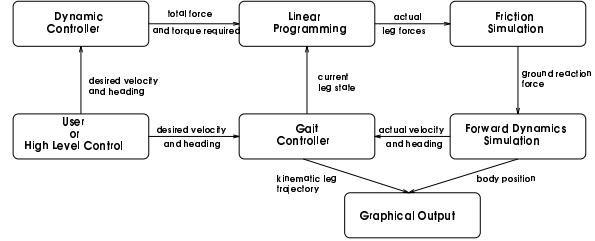


Figure 2: *System overview.*

ground, otherwise their motion is computed kinematically (we assume that the leg mass is small compared to the body mass). This reduces the complexity of the dynamic control algorithms while allowing us to carefully design a natural looking motion for the legs during their transfer phase.

A schematic overview of the system can be seen in Figure 2. The *desired* motion control input,  $\mathbf{s}^d$ , comes either from the user or from a higher level motion planning system. A motion planning system can use simulated sensory data and/or some sort of a task description to produce the desired animal behavior. In our case, the motion control input consists of the desired values for the body’s velocity,  $\dot{\mathbf{c}}^d$ , and the heading determined by the yaw angle  $\theta_z^d$ . Based on the difference between the actual and the desired states  $\mathbf{s}$  and  $\mathbf{s}^d$ , respectively, the *dynamic controller* computes the aggregate force  $\mathbf{F}^r$  and torque  $\mathbf{T}^r$  that should be applied to the body at the center of gravity to move from  $\mathbf{s}$  to  $\mathbf{s}^d$ . At any instance in time, one or more legs of the animal are touching the ground. They provide the only means by which the animal can move, hence,  $\mathbf{F}^r$  and  $\mathbf{T}^r$  should be distributed among them. The problem can be formulated as a linear constrained minimization and is solved using the Simplex method for *linear programming* (subsequent section). The computed leg forces are exerted on the ground which in turn, depending on the frictional properties (described in a following section), generates reaction forces causing the animal body to move. Based on these forces, a forward dynamic simulation is performed to compute the new state of the body.

The leg subsystem is controlled by the kinematic *gait controller*. The purpose of this controller is to arrange the motion of the legs based on their current state. The gait controller decides whether a leg will be exerting a force on the ground or will enter its transfer phase and therefore will be lifted from the ground. It also computes based on the gait pattern, the heading and the velocity of the body, how far the leg will move during its transfer phase before touching the ground. Finally, the gait controller is responsible

for the kinematic motion of the leg during the transfer phase.

What follows is a detailed description of each module of the system.

## 4 The Dynamic Controller

To be able to control a given system we first need to define it dynamically. In control theory, the model we use to simulate the dynamics of a given system is called the *plant*. Once the plant has been defined, we apply a control strategy to control the behavior of some of its variables (also called the state of the system). A number of techniques exist in the robotics literature dealing with the problem of dynamic control of articulated figures.

In this paper, we adopt the following simplifying assumptions for the animal's dynamic model. The inertial properties of the animal can be completely described by the mass  $m$  and the inertia tensor  $\mathbf{I}$  of its body. The mass of the legs is considered to be small compared to the mass of the body, so that their motion does not influence the overall animal motion. As a result of these assumptions, we need to dynamically control only the state of the body subsystem.

To control the body's motion, it is sufficient to control the position  $\mathbf{c}$  and the orientation  $\theta$ . In practice though, it is often more desirable to control the velocity,  $\dot{\mathbf{c}}$ , than the position  $\mathbf{c}$ . The only way the body's state can be actively changed is by exerting forces through the legs in contact with the ground (simulating the action of muscles). All the leg forces acting together result in a net force,  $\mathbf{F}$ , and a net torque,  $\mathbf{T}$ , at the center of gravity of the body. These in turn result in the linear and angular acceleration vectors  $\ddot{\mathbf{c}}$  and  $\dot{\omega}$ , from which we update the animal position in space through numerical integration. Therefore, the dynamic controller must control the body's state variables indirectly through  $\mathbf{F}$  and  $\mathbf{T}$ .

The design of the controller depends on the following two restrictions: (1) the body of the animal should never hit the ground, and (2) the animal should be able to stay on course even when moving on uneven terrain. Based on these conditions we chose to control the three orientation angles  $\theta_i$  of the body, the vertical distance of the the center of gravity (c.o.g.) from the ground  $c_z$ , and the two horizontal velocities  $c_x$  and  $c_y$  of the c.o.g. In this way we can control the speed and heading of the animal and ensure that it can deal with varying ground elevations. We will approximate the body subsystem as six decoupled dynamic sub-systems. The first three sub-systems represent the way the animal body rotates about its c.o.g. and the other three the motion of the c.o.g. through space.

Each of the six dynamic sub-systems consists of two main parts. The *ideal controller* and the *body's dynamic plant*. The ideal controller takes as input the desired and actual value of the state variable it controls and outputs a force (or torque) needed to be applied in order to eliminate any difference between the two values. In a system with no external perturbations, no feedback from the plant would be required and an open-loop controller would be sufficient. In our case, unexpected external events such as stepping on uneven or slippery terrain occur and the feedback controller ensures that the three minimum conditions mentioned above are met.

Since we need to control both the position and the velocity of the model, we employ two different types of controllers. The *double integral plant* controller shown in Figure 3(a) is used to control the position variables, namely the  $\theta_i$ 's and  $c_z$ . The *single integral plant* controller shown in Figure 3(b) regulates the two velocity variables,  $c_x$  and  $c_y$ . Both controllers have as external input the desired value of the controlled variable,  $s_i^d$ , and produce an acceleration  $\ddot{s}_i$  in order to make  $s_i$  equal to  $s_i^d$ . The acceleration can be readily converted to the *required* force  $\mathbf{F}_i^r$  or torque  $\mathbf{T}_i^r$  from the basic dynamic relationships

$$\mathbf{F}_i^r = m\ddot{c}_i, \quad \mathbf{T}_i^r = \mathbf{I}\omega_i, \quad (2)$$

where  $i$  is one of the  $x, y$  or  $z$  coordinate axes,  $\mathbf{F}^r = \{\mathbf{F}_x^r, \mathbf{F}_y^r, \mathbf{F}_z^r\}$  and  $\mathbf{T}^r = \{\mathbf{T}_x^r, \mathbf{T}_y^r, \mathbf{T}_z^r\}$ . The output  $\mathbf{F}^r$  and  $\mathbf{T}^r$  of the six controllers is then distributed to the legs using the algorithm described in the next section. The actual leg forces are used in the forward dynamic simulation that is represented as the *Body Dynamics* box in the two controller figures.

The controllers were designed so that the closed-loop response would be of the general form

$$\frac{s_i(s)}{s_i^d(s)} = \frac{1.0}{\frac{s^2}{w^2} + \frac{2u}{w}s + 1.0} \quad (3)$$

in the complex frequency domain (after taking the Laplace Transform). In this expression,  $u$  is the *damping ratio* of an ideal control loop,  $w$  is the *undamped natural frequency* and  $s$  is the *complex frequency*. The Laplace transform of the controlled variable is  $s_i(s)$ , while the Laplace transform of the time-varying desired value of that variable is  $s_i^d(s)$ . The  $u$  and  $w$  parameters specify the damping and the speed of response of the controller to a desired value and are set by the user or the high level motion planning system. The values we used in our examples were  $u = 0.707$  and  $w = 3.14 \text{ rad/sec}$ .

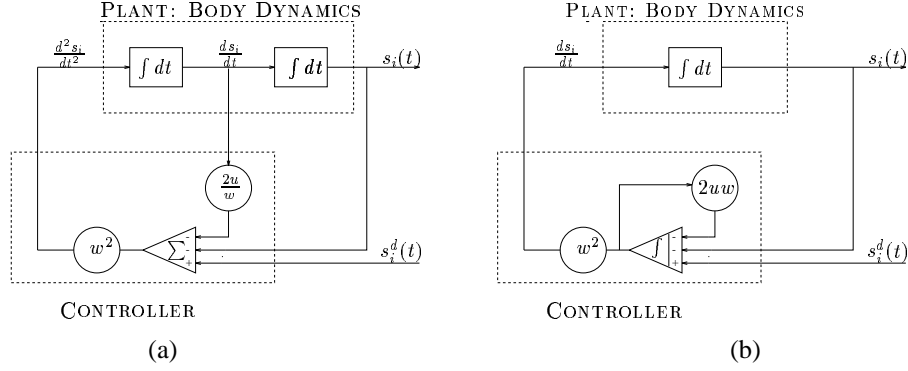


Figure 3: (a) Double integral and (b) Single integral controllers

## 5 Distributing forces to the legs: Linear Programming

As mentioned above, the dynamic controllers compute the total required force  $\mathbf{F}^r$  and torque  $\mathbf{T}^r$  that needs to be applied to the center of mass of the animal body, so that the body obtains the desired velocity and orientation. This force and torque should be the result of forces exerted by the ground through the legs. We therefore need an algorithm that determines how the total force and torque should be distributed to the legs.

The fraction of the total force that each leg will exert to the ground is determined using linear programming. Clearly, only the feet that are instantly in contact with the ground are capable of moving the body. The orientation of the body is controlled by the torques applied at the c.o.g. These torques result from the force applied to each leg by the ground and depend on the relative position of the tip of that leg with respect to the center of the body. Limits exist on the maximum force that each leg can exert. Therefore, depending on the number of legs in contact with the ground and their relative position, it might or it might not be possible to apply the required forces and torques (computed by the dynamic controllers). The purpose of the linear programming procedure is to find the optimal force assignment to the legs which minimizes the difference between the required and the actual values.

Linear programming was chosen for the force distribution for its versatility and efficiency. The Simplex method that is most often used to solve linear programming problems can be expected to run in linear time for most of the cases [WPV86]. Linear programming, sometimes called linear optimization, finds  $\mathcal{N}$  independent variables to minimize a linear cost function  $C$  subject to the primary constraint that all the variables have non-negative values and that they simultaneously

satisfy  $\mathcal{M}$  additional linear constraint equations or inequalities. In our case, the variables are the leg forces  $\mathbf{F}_l$  for all legs  $l$  that are in contact with the ground.  $\mathbf{F}_l = 0$  for all other legs. Consequently the total force exerted by the legs is

$$\mathbf{F} = \sum_{l=1}^4 \mathbf{F}_l. \quad (4)$$

If  $\mathbf{d}_l$  is a vector from the c.o.g. to the tip of leg  $l$ , we can write the total torque about the c.o.g. as

$$\mathbf{T} = \sum_{l=1}^4 \mathbf{F}_l \times \mathbf{d}_l. \quad (5)$$

We then express the cost function as

$$C = c^1 |F_x^r - F_x| + c^2 |F_y^r - F_y| + c^3 |F_z^r - F_z| + c^4 |T_x^r - T_x| + c^5 |T_y^r - T_y| + c^6 |T_z^r - T_z|, \quad (6)$$

where the total force and torque are expressed in Cartesian space. The cost function  $C$  is a measure of the total error in meeting all force and torque requirements. The constants  $c^1$  through  $c^6$  are positive weighting constants on the individual components of the total error. The function  $C$  is positive definite and becomes zero only when the required force equals the actual force and the required torque equals the actual torque. The cost function is converted to a linear function by adding a “slack” variable for each difference and removing the absolute value operators, a common technique used in linear programming [WPV86]. The weighting factors  $c^k$  may be adjusted to impose variable accuracy in each of the motion degrees of freedom. For example, increasing  $c^6$  results in reduced errors in meeting the heading angle requirement and hence results in better control over the direction the animal is heading. However, regulation of other

coordinates may suffer as a result. Finally, the  $\mathcal{M}$  additional constraint inequalities are due to the physical limits in the force that a leg can exert on the ground and have the form

$$\mathbf{F}_l \leq \mathbf{F}_{max}, \quad (7)$$

where  $\mathbf{F}_{max}$  is the maximum leg force allowed.

Solving the linear programming problem results in a reasonable distribution of forces to all the legs that can influence the body motion. In our experience, linear programming performs very well in cases where the errors are small. However, when the required and actual forces and torques cannot be matched sufficiently close, even small changes in the weights of the cost function  $C$  can result in significantly unbalanced force distribution. Adjusting the  $c^k$ 's can be achieved through experimentation with the system.

## 6 Modeling the Ground

Our locomotion algorithm is not tailored to any specific terrain description. We created a general motion system that can deal with a variety of terrains. We use the function  $G(x, y)$  to model the ground elevation with respect to a 2D grid.

To model the ground's frictional properties we adopt a Coulomb friction model. If the tip of a leg (the center of the dog's paw) is at the location  $(x, y, G(x, y))$ , then  $\mathbf{f}_{\parallel}^g(x, y)$  and  $\mathbf{f}_{\perp}^g(x, y)$  are the two components of the force  $\mathbf{f}^g$  exerted by the ground to the leg.  $\mathbf{f}_{\parallel}^g(x, y)$  is the component parallel to the tangent plane between the ground and the paw, while  $\mathbf{f}_{\perp}^g(x, y)$  is normal to that plane. Based on the Coulomb friction model

$$|\mathbf{f}_{\parallel}^g(x, y)| \leq \mu |\mathbf{f}_{\perp}^g(x, y)|, \quad (8)$$

where  $\mu$  is a friction coefficient whose value varies between 0 and 1. The smaller the value of  $\mu$ , the more slippery the ground.

The way the ground interacts with our dynamic model is the following. Through linear programming we compute the force  $\mathbf{f}^l$  that each leg should exert to the ground and we decompose it to two orthogonal components  $\mathbf{f}_{\parallel}^l$  and  $\mathbf{f}_{\perp}^l$ , where the directions  $\parallel$  and  $\perp$  where defined above. Then the two components  $\mathbf{f}_{\parallel}^g$  and  $\mathbf{f}_{\perp}^g$  of the force  $\mathbf{f}^g$  that the ground exerts on the leg are computed as follows

$$\begin{aligned} \mathbf{f}_{\perp}^g &= -\mathbf{f}_{\perp}^l \\ \mathbf{f}_{\parallel}^g &= \begin{cases} -\mathbf{f}_{\parallel}^l & \text{if } |\mathbf{f}_{\parallel}^l| \leq \mu |\mathbf{f}_{\perp}^l| \\ -\frac{\mathbf{f}_{\parallel}^l}{|\mathbf{f}_{\parallel}^l|} \mu |\mathbf{f}_{\perp}^l| & \text{otherwise.} \end{cases} \end{aligned} \quad (9)$$

## 7 The Gait Controller

Gaits of different animals have been studied extensively and lots of interesting observations can be found in the zoology literature [Suk68, Ale84, Bro86, Muy57]. Since our purpose is to combine kinematics with dynamics, we have used snapshots from studies of animal walking and trotting conducted in [Bro86] and [Muy57]. Some technical terms needed for the description of gaits can be found in [Ale84] and we repeat them here.

- A *stride* is a complete cycle of leg movements, for example the sequence from the setting down of a particular foot to the next setting down of the same foot.
- A *stride length* is the distance traveled in a stride.
- *Stride frequency* is the number of strides taken per unit time.
- The *duty factor* of a foot is the fraction of the stride during which the foot is on the ground.

The main task of the gait controller is to regulate the motion of each leg according to the current gait, velocity and turning rate. Regulating the leg motion consists of deciding when the leg should be pushing, when it should be lifted from the ground, and adjusting the stride length and stride frequency. We have implemented two different gaits, the *walk* and the *trot*. The dog automatically switches between them, depending on its speed. The walking gait is used by dogs when they move slowly. When walking on flat surfaces with constant velocity, three legs are always in contact with the ground and one is lifted. The order in which legs are lifted during a stride is

$$LF \rightarrow RH \rightarrow RF \rightarrow LH,$$

where  $L, R, F, H$  stand for Left, Right, Front and Hind, respectively. This order ensures that the center of mass is always within the triangle formed by the tips of the legs on the ground and guarantees the *static stability* for the animal. During a constant speed walk on flat terrain, the duty factor for each foot is 0.75. At faster speeds, dogs trot. During the trot, the LF and RH move together and so do the RF and LH. The duty factor for each foot is 0.5 and at any time two feet are on the ground while the other two are lifted. During the trot the dog has to maintain *dynamic stability*, i.e., the animal cannot keep its balance without moving. In our system, dynamic stability is enforced by the feedback dynamic controller which produces corrective torques to counteract any tendency of the body to roll over.

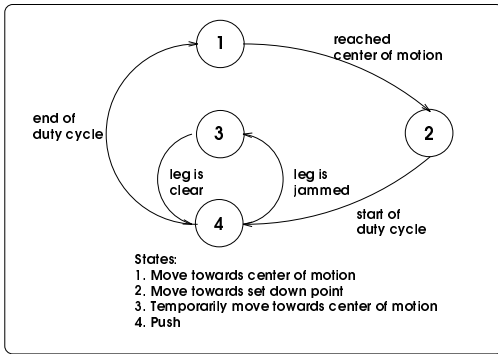


Figure 4: Automaton for leg motion transitions.

The gait controller is responsible for the kinematics of the leg motion. When a leg is touching the ground, its motion is guided by the ground reaction forces. This enhances the realism of the animation. However, when the leg is lifted from the ground, there is no reason to keep on treating it dynamically. Its motion has negligible effects on the motion of the rest of the body, due to their respective significant mass differences.

The inputs to the gait controller are the desired speed and turning rate. The higher the speed, the longer the stride length and the stride frequency. This means that the swing angle of a leg when lifted from the ground is proportional to the animal's speed. When the animal is turning, the gait controller positions the legs not only forward but also sideways so that they can apply the appropriate torque for the turn to take place.

The state of each leg is determined by the automaton shown in Figure 4. Each leg can be in one of four distinct states. Events that cause a change in the leg's state are triggered by the gait controller either as a consequence of timing or as a result of the leg's orientation. State 4 is the only state where a leg is touching the ground and pushing. When the duty interval of the leg is over, the gait controller triggers a transition from state 4 to state 1 for that leg. In state 1, the leg moves towards the center of its swing (center of motion). When it reaches that point, it starts moving towards its *set down point*. The set down point is determined by the gait controller and depends on the linear and angular velocity of the body, as explained above. Finally, when it's time for the duty interval to begin, the gait controller once more triggers a transition, this time from state 2 to state 4. The leg goes into state 3 if it gets jammed. Legs get jammed as a result of unexpected events that cause the hip joint to rotate outside the operational area of the leg. Slippery ground or very rough terrain could cause a leg to jam. Since the leg has to get back into its operational area,

it gets lifted temporarily and moves for a short period of time towards the center of motion, until it reaches a safe position where it can go back to state 4. It is desirable for a leg to stay only for short periods of time in state 3, since an unexpected lift creates potential hazards to the stability of the body.

While a leg is in states 1, 2 or 3, the gait controller is responsible for kinematically setting the angles of its ankle and paw joints so that their motion looks as natural as possible. It turns out that the correct motion of these joints adds a lot to the visual effect of the animation.

Although our specific gait controller was tailored to a four-legged animal, we could extend it to accommodate any reasonable number of legs using the same concepts.

## 8 Experiments

We have tested our system through a series of animations on both flat and uneven terrain. Our experiments include simulations of realistic walking, trotting and transitions between these two gaits on even and uneven terrain. They also include simple behaviors like target pursuit at variable speeds. Our experiments run at interactive rates on a Silicon Graphics R4000 Crimson workstation with VGX graphics. In our experiments, a simulated visual system or the user determined the values for the desired velocity and heading.

In the first experiment shown in Figure 5, snapshots (a) through (d) demonstrate a complete trotting stride on flat terrain. The body weight is supported by two legs, the front right and the hind left in snapshots (a) through (c), and the front left and hind right in snapshot (d). Therefore the dog achieves dynamic stability during this gait.

The test run shown in Figure 6 simulates a dog walking up and down a ramp. The ground elevation is given by a function  $G$ . The dog's posture is automatically adjusted to accommodate for the difference in elevation between the front and the hind legs.

To enhance the quality of the animation, we added a periodic kinematic motion of the animal's tail and a random lateral motion of the head. However, the fact that the dog model we used consisted of rigid parts made the overall motion look stiff. Real dogs have considerable flexibility in their body, and their spine bends quite a bit to facilitate walking or running. This flexing behavior could not be replicated with our current model.

To demonstrate how our system can be used as a basis for more complex animations (shown in the videotape), we created a higher level motion driving system that generated tracking behavior. A little red

ball was the target our dog was trying to catch. With simulated visual sensors, we compute the position of the ball. The motion driving system commands the dog to head towards the ball and to move towards it with a velocity proportional to the distance from the ball. We kinematically added a motion of the head so that the dog is always looking in the direction of the ball. In this simulation the dog was able to start from rest, accelerate to moderate walking, then move faster and switch to trotting while approaching the target. Unfortunately for the dog, when the dog would get too close to the ball, the ball would roll to a different position, causing the dog to move towards the ball. The advantage of having an autonomous locomotion simulation system is apparent since we did not have to modify our walking code at all for this experiment.

## 9 Conclusions

In this paper we have developed a new approach towards the animation of autonomous legged animals combining dynamic control with kinematic motion. A dynamic feedback controller regulates the state of the animal's body by computing the necessary forces and torques to achieve a desired state. The forces are distributed to the animal's legs using linear programming. Furthermore, we developed a kinematic gait controller to arrange the motion of the legs during a stride. We were able to generate realistic walking and trotting animations on flat, and uneven terrain. Our locomotion system is autonomous in the sense that it is not tied to any particular environment. It works without modification for any appropriately defined terrain.

In our experience, the correct motion of the animal's legs while they are off the ground is a very important factor in determining the quality of the animation. A carefully tuned kinematic procedure can accommodate this motion. Dynamics and control should be used to account for unexpected events and to ensure the feasibility and validity of the desired motion of the whole body given the current model state and the ground friction properties. We believe that our combined approach successfully deals with many of the issues in autonomous realistic animations of legged animals.

We can currently model simple behaviors such as target following at variable speeds and motion on uneven terrain with variable frictional properties. These simple behaviors can be used by a higher level cognitive system to simulate more complex behaviors.

## 10 Acknowledgments

We would like to thank Viewpoint Datalabs for providing us with the accurate German Shepherd polyg-

onal model. This work was supported by NSF IRI-9309917 and NSF MIP-94-203907.

## References

- [Ale84] R.Mc.N Alexander. The gaits of bipedal and quadrupedal animals. *International Journal of Robotics Research*, 3(2), 1984.
- [BC89] Armin Bruderlin and Thomas Calvert. Goal-directed, dynamic animation of human walking. In *Computer Graphics (SIGGRAPH proceedings)*, volume 23, pages 233–242. ACM, July 1989.
- [Bro86] M.C. Brown. *Dog Locomotion and Gait Analysis*. 1986.
- [Gir87] Michael Girard. Interactive design of 3d computer-animated legged animal motion. *IEEE Computer Graphics and Applications*, 7(6):39–51, June 1987.
- [GM85] Michael Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. In *SIGGRAPH*, volume 19, 1985.
- [Hod94] Jessica Hodgins. Simulation of human running. In *IEEE Robotics and Automation*, 1994.
- [Muy57] E. Muybridge. *Animals in Motion*. Dover Publications, 1957.
- [MZ90] Michael McKenna and David Zeltzer. Dynamic simulation of autonomous legged locomotion. In *Computer Graphics (SIGGRAPH proceedings)*, volume 24. ACM, August 1990.
- [Par73] William Park. *Control of Multilegged Vehicles*. PhD thesis, University of Pennsylvania, 1973.
- [RH91] Marc H. Raibert and Jessica K. Hodgins. Animation of dynamic legged locomotion. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 349–358, July 1991.
- [Suk68] V.B. Sukhanov. *General System of Symmetrical Locomotion of Terrestrial Vertebrates and some Features of Movement of Lower Tetrapods*. Nauka Publishers, 1968.
- [vFV92] Michiel van de Panne, Eugene Fiume, and Zvonko Vranesic. A controller for the dynamic walk of a biped across variable terrain. In *Proceedings of the 31st Conference on Decision and Control*, December 1992.
- [Wil86] Jane Wilhelms. Virya—a motion control editor for the kinematic and dynamic animation. In *Graphics Interface*, 1986.
- [WPV86] S.Teukolsky W. Press, B. Flannery and W. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1986.
- [WS89] J. Wilhelms and R. Skinner. An interactive approach to behavioral control. In *Graphics Interface*, 1989.

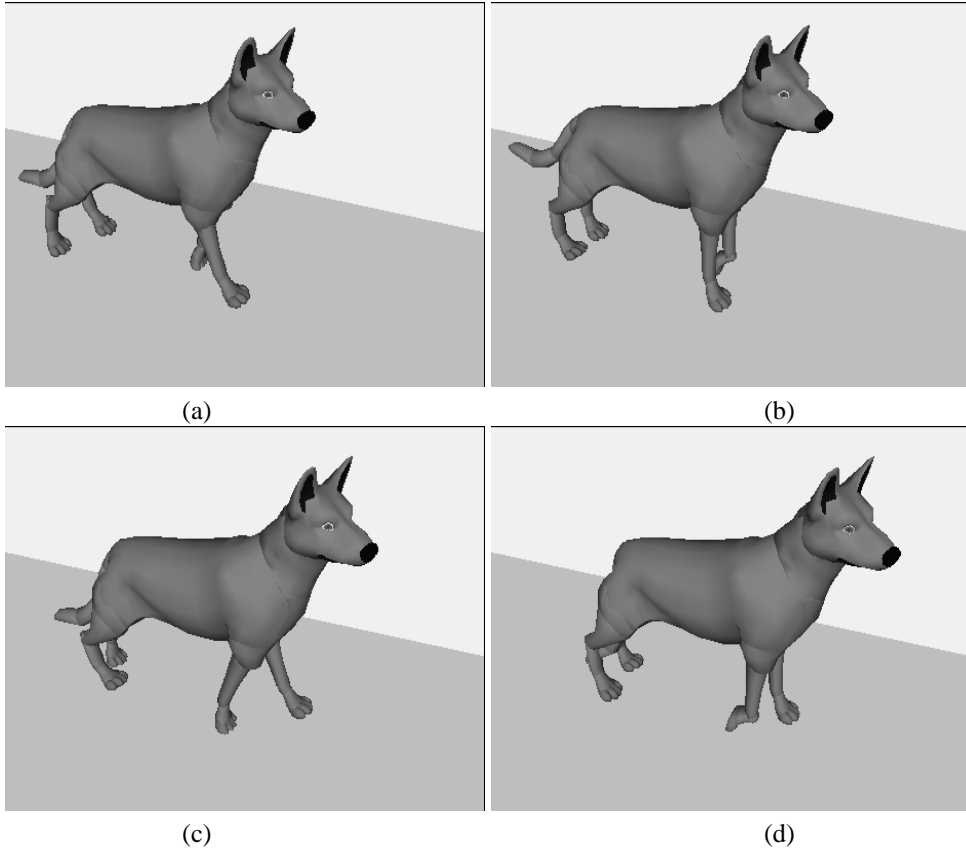


Figure 5: *A complete stride with the walking gait.*

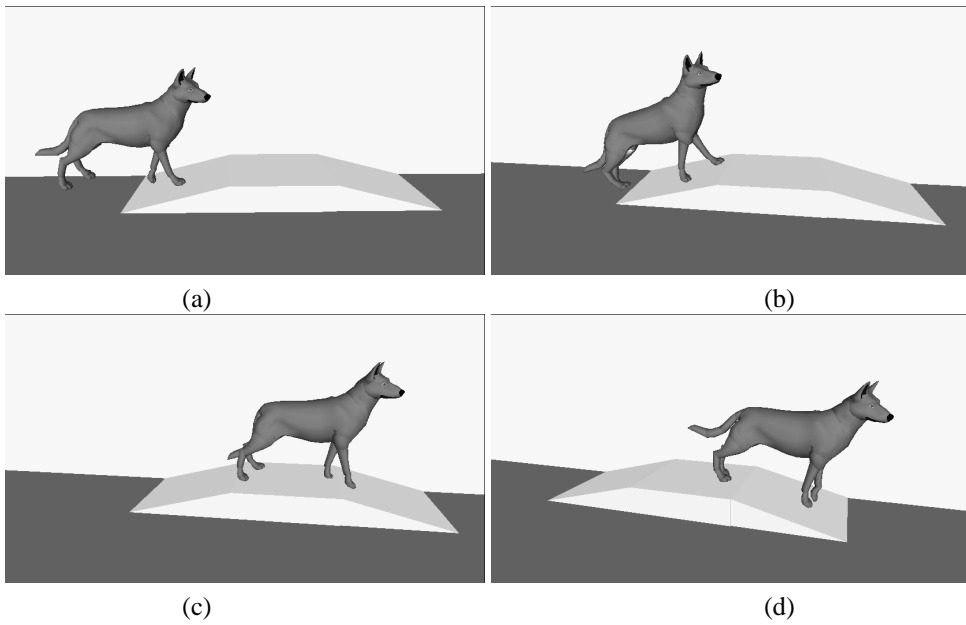


Figure 6: *Walking over a bump*