Making them move: Motor control for animated humans and animals

Michiel van de Panne

Department of Computer Science
University of Toronto[1]

Abstract

Simulating the motion of humans and animals is a challenging problem that is of interest to various disciplines, including biomechanics, control, and robotics. A surprising amount of progress is also being made in the computer graphics community, whose ever-increasing demand for realism has led to the use of complex physically-based simulations. This is motivated by the goal of creating simulated autonomous characters equipped with basic motor skills for movement. The problem of encoding these basic motor skills is one which has seen a variety of proposed solutions in computer animation. This talk will give an overview of some of the control methods applied towards the animation of complex simulated models of humans and animals. These control methods vary in their choice of problem representation but typically rely on numerical optimization techniques. A variety of animation results will be shown.

1 Introduction

Graceful gymnasts, leaping lions, and lumbering dinosaurs are all the product of physics and control. They are also the kinds of motions which animators wish to reproduce and manipulate. Over the past decade, research in computer animation has sought to build tools capable of the automated synthesis of natural, dynamic motions[2]. A considerable amount of effort has been focussed on using physically-based simulations in conjunction with control methods in order

[1] *van*@dgp.utoronto.ca
`http://www.dgp.utoronto.ca/~van`

We gratefully acknowledge the financial support of ITRC (Ontario) and NSERC (Canada).

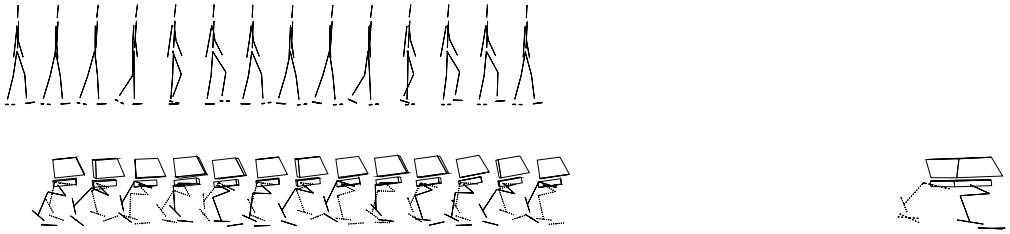


Figure 1: Simulated movements for a human model and robot.

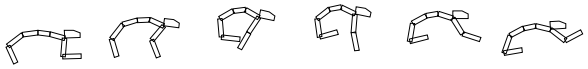


Figure 2: A bounding cheetah model.

to produce natural character movement. It is at this point that an animation problem becomes a control problem.

We have explored a variety of control techniques which make extensive use of numerical simulation and numerical optimization techniques. The availability of rich, efficient simulation environments provide new opportunities for exploring control techniques and control representations. At the same time, autonomous behaviours for simulated characters can potentially benefit from the rich family of well-studied control techniques.

The ultimate goals are challenging. How do we design a controller which implements a complete motor skill, such as a horse capable of walking over most types of terrain? How might we design and efficiently control the movement of a newly-invented creature, knowing only its physical design? These goals are ill-defined in many ways, but they form an interesting and suitably broad target.

2 An Animation Perspective

Why might animators resort to complex physical simulations combined with complex control schemes in order to obtain animated motion? To understand the answer, one first needs to understand the tools of the trade for animation. While many of today's feature films make use of impressive computer-generated imagery, character motion usually still remains the work of individual animators who direct every aspect of each character's motion.

One of the most commonly-used techniques is that of keyframing, which has the animator specify the position of a character in selected 'key' frames, which are then smoothly interpolated in order to yield the desired motion. While keyframing provides an animator with complete control over a motion, it clearly involves considerable effort and talent on the part of an animator.

A more recent innovation has been the use of motion capture techniques. A

variety of technologies exists for tracking and recording the real-time motion of multiple 'markers' placed on humans or animals. This data can then be used to animate the movement of an equivalent computer-generated model, modulo potential difficulties with noise, scaling, and editing of motion segments[29, 4, 25].

Thus, why resort to using simulations for producing animated movement? Simulated creatures capable of autonomous behaviours potentially require less effort to animate and are reusable. At present, the most significant use of autonomous simulated behaviour has been in animated scenes involving flocks or herds of tens or hundreds of animals.

Lastly, physical simulations establish the connection between the physical design of an animal and the ways it can move. They allow an animator to design a creature and then ask the question 'How would it move?' Clearly, the mass distribution, location of joints, and distribution of muscles have a strong influence on how any creature will move. Knowing the relationships between such physical parameters and the resulting motion patterns also has potential applications in prosthetic design, kinesiology, and other domains.

There are several desirable control requirements which are potentially important for use in animation. The first is the provision for different levels of control. The animator of a foreground character might demand precise control over the motion of a finger. For a background character, high-level commands such as 'walk to the door' are desirable. Clearly, the control problems involved in both cases are quite different.

Developing character models which can be simulated and controlled at multiple scales of detail is also an issue in animation. A simple example is that of a marathon race viewed from a distance. In such a case one would hope to use a greatly simplified model for the simulation and control.

Lastly, working with simplifications such as linearizations of a dynamical system is often uninteresting and inadequate for animation. The problem is that while such simplifications may make for tractable analysis, they frequently also result in visible, unnatural artifacts in the movement. This makes tackling the control problems yet more challenging.

3 The nature of the problem

It is endlessly frustrating to see the ease and grace with which human and animal motions are executed, while knowing how difficult it is to create a simulation of the same phenomena. The difficulties are many, and the following discussion examines only some of the issues intimately associated with motion control.

3.1 Defining a motor skill

In deciding to tackle a general problem such as simulating human or animal motion, it is helpful to set some lofty goals. One such goal is to design a control system capable of a particular motor skill, such as walking for a human, or galloping for a horse. Unfortunately, it is difficult to give an all-encompassing

definition of any motor skill. A walking motor skill can have an infinite number of variations, based upon how well it can cope with variations in terrain, speed, surface friction, obstacles, extra carrying weight, type of footwear, wind, gravity, etc.

This leads indirectly to the question of how to characterize the difficulty of a control problem, or, alternatively, to measure the 'skill' of a given controller. Such characteristics are perhaps best measured implicitly by having the controller perform its task over a given 'race-course' under a well-defined set of conditions, much like an athletic competition.

Two types of control subtasks seem to consistently emerge from controllers implementing a complex compound skill, such as 'walk to the door': *planning* and *execution*. Should the planning be considered part of the control problem? A variety of experience indicates that these should not be treated as independent problems. We can point to problems of the type shown in Figure 3, where the object is to walk across a river by stepping on a series of stones. The planning process needs to take the leaping abilities into account, and in particular, the ability to execute a given sequence of leaps.

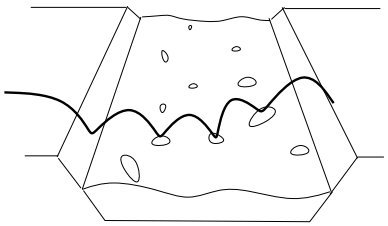


Figure 3: The crossing-the-river problem.

It is also difficult to separate the capture and interpretation of sensory information from the tasks of planning and execution. The execution of a given motor skill may change drastically based upon the available sensory information. Walking in darkness over variable terrain is an example. Although the importance of vision in walking is obvious, it is a reminder of the importance of all types of sensory information in almost any motor skill.

3.2 Experimental approaches to control synthesis

Much of our work can be regarded as taking an experimental approach to control problems. Controllers are synthesized using numerical optimization procedures, which require repeated simulations to guide the synthesis process. There are several strong arguments to be made for *numerical* approaches to control synthesis over purely *analytic* methods.

First, numerical methods can sometimes cope better with the complexity of dynamical systems. When the generation and use of complex sensory information

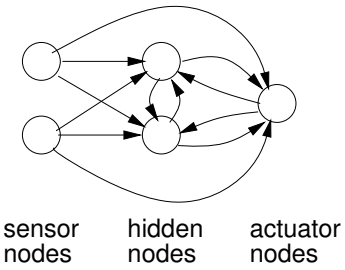


Figure 4: Topology of a sensor-actuator network.

must also be considered, the use of a simulation-based approach is perhaps the only feasible path for control synthesis.

Second, designing and testing learning or adapative techniques requires extensive use of simulations. Learning or adaptation is a key part of any realistic model of a motor skill, and learning strategies are difficult to evaluate analytically.

A last argument for a more experimental approach lies with the current availability of sophisticated physical simulation software. It is relatively simple today to build reasonably accurate physical simulations of humans or animals with given physical parameters. Simulations are also easily adapted to reflect different terrain conditions, as well as to simulate any type of sensory input. Experimental approaches allow for the evaluation of many alternative choices for representing and learning control functions. Although analysis by itself leads to important insights[1], it often does not reveal enough to build a working controller.

3.3 A sampling of results

The following three sections represent different approaches to solving control problems for autonomous simulated creatures. We focus on describing the fundamental ideas behind each approach and refer the reader to the original papers for a full description.

4 Sensor-actuator networks

Sensor-actuator networks (SANs) are a simplistic type of control structure that allows for the 'discovery' of modes of locomotion[27]. The user supplies the configuration of a mechanical system that has been augmented with simple sensors and actuators. A stochastic algorithm then explores various ways in which the sensors and actuators can be wired together in order to produce modes of locomotion.

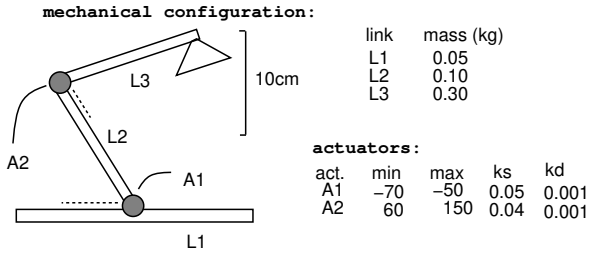


Figure 5: The Luxo creature.

There are a few noteworthy features of SANs. First, the building blocks of the control system are a rather unlikely collection of parts: binary sensors, PD actuators, and summation nodes with time delays. Surprisingly, these can be assembled together to control unstable, dynamic movements. More surprisingly, the job of building a controller from these parts can be automated. Second, stochastic techniques are used to perform the controller synthesis. This is illustrative of the potential power of numerical optimization techniques applied to control problems. Although the following discussion summarizes some of our own work, others in animation have also pursued similar lines of thought[5, 19, 23, 9].

4.1 Constructing and simulating a creature

The types of creatures used for our experiments are mostly 2D articulated figures, built out of rigid links connected with joints. Associated with each joint is an actuator that produces a torque to drive the joint towards a desired reference position using a local PD controller. An example creature is shown in Figure 5, representing a desklamp which becomes 'alive'. The equations of motion for the creatures are generated automatically by a *dynamics compiler* of our own construction, although commercial programs such as SD/FAST[12] provide even more sophisticated capabilities. Ground reaction forces are generated by a penalty method which includes a friction model.

4.2 Sensor-actuator networks

The control structure is a sensor-actuator network: a small non-linear network of weighted connections between a collection of binary sensors and the actuators (the muscles of our creatures). An example topology is shown in Figure 4. The network has internal delays, thereby giving it dynamic properties, most notably the ability to oscillate in many ways, and for sensors to influence the oscillation pattern. SANs are similar in nature to many artificial neural networks (ANNs), but are different in several respects. The nodes of SANs have associated time delays which are a key property governing their behaviour. More importantly, our synthesis method does not employ any derivative-based learning methods.

Each node in a SAN computes the weighted sum of its inputs and sets the output to 1 or 'on' after a given time delay if the input sum exceeds a fixed threshold value. The actuators are directly controlled by the output *actuator nodes* in the network. The input sum for these nodes provides a value proportional to the desired reference position for the PD controller driving the associated joint.

Each SAN is defined by a set of parameters which govern its behaviour. These include the weights connecting outputs of nodes to inputs of other nodes, the internal time-delays of nodes, and the parameters defining the mapping which provides reference positions to the actuator joints. The collection of these parameters defines the search space for the control synthesis technique.

4.3 Control Synthesis

We use a *generate-and-test* technique to synthesize appropriate SAN controllers, as described in Figure 7 and depicted in Figure 6. While this is a 'blind' technique in that it has no *a priori* knowledge about the problem, it is capable of first finding and then fine-tuning many suitable modes of locomotion, some of which are quite unexpected. Two phases are involved: a *global* search, followed by a *local* search through the parameter space. The sole purpose of the global search is to quickly determine a small set of promising regions in the search space. While most randomly generated controllers will result in little or no useful motion, 1-5% do in fact produce useful motion. The subsequent local search then refines the parameter values using a greedy local optimization procedure. The most common optimization criterion we have used is the distance travelled as measured during T seconds of simulation.

Interesting results have been achieved for a reasonable variety of creatures. A controller can be automatically generated which makes a simulated fish swim forward and even chase a target when equipped with two binary eyes. A mechanical model of a desk lamp can be taught to move in leaps and bounds. Objects as simple as two or three link chains can move in varied and curious ways.

Needless to say, there are limits to the complexity of motion that can be expected to emerge using SANs and the stochastic optimization procedure. It would be futile, for example, to expect to use our method directly to synthesize a SAN for controlling a human performing a high jump. Another major problem is that the operation of a SAN is not readily understandable. To overcome some of these problems, the next technique applies numerical optimization to a more recognizable type of control system.

5 Parameterized state machines

It is obvious that most forms of locomotion have a periodic nature. In this section we describe a control synthesis technique that exploits this, and which also relies on the surprising passive stability of many motions. In many ways, this work answers the question 'To what extent can we control gaits without the

use of sensory feedback?' The results discussed below are collated from [26] and [28].

The best point of departure for explaining the approach is to begin with the motion of a windup toy. A mechanical windup toy goes through a periodic sequence of positions in order to achieve forward motion, driven by a toy motor or spring. There is typically no feedback – the toy is oblivious to its environment. A walking windup toy stays on its feet because of the inherent stability of its motion. Some windup toys are capable of more impressive manoevres, such as back-flips.

Now suppose we imagine providing designers of windup toys with a sophisticated suite of simulation tools, which would let them carefully optimize the design of windup toys. What types of open-loop motions are possible? Surprisingly, some very dynamic motions, such as the simulated bounding of the 'cheetah' creature shown in Figure 2, can be controlled as a *virtual windup toy*. It is also possible to automatically optimize dynamic actions such as the *leaps* of a Luxo lamp (see Figure 12) or to produce a variety of walking and running gaits for creatures like simulated cats and 'monsters' (see Figure 8).

5.1 Pose-control graphs

The basis of our simulated wind-up toys is a simple state machine, which we refer to as a *pose control graph* or PCG. An example is shown in Figure 9 for Luxo, the jumping desk lamp. Each state in the PCG has a set of reference positions for the actuated joints of a creature. The poses of Luxo in Figure 9 illustrate the reference (i.e., desired) shape of Luxo in the three states. The state transitions are based upon fixed time intervals or simple sensory information, such as a particular point making or breaking ground contact. The PCG itself defines the *desired* internal shape of the creature over time. Its real shape and motion are determined by interactions with the environment and the reaction forces that are produced.

PCGs are defined by a large set of parameters which represent a suitable search space for numerical optimization techniques. These include the joint reference positions in each state (the 'poses') and the state transition times. Any chosen set of parameter values represents a controller whose performance can be evaluated through the use of trial simulations. Speed of locomotion once again represents a straightforward optimization function, which is measured by recording the distance travelled during a fixed simulation time. Other functions may be useful, however, in ensuring that a particular type of gait is produced, or ensuring that a gait remains robust[26]. Figure 10 shows a visualization of the optimization function for a two-dimensional slice through the parameter space, in this case for a 3D simulated model of a cat attempting to trot. It is clear that any optimization technique must be able to escape from local minima.

Several varieties of numerical optimization technique seem to succeed at searching for near-optimal parameter values in the search space. We have experimented with versions of simulated annealing and other hill-climbing algorithms.

These optimization algorithms each have several parameters of their own which need fine-tuning, but in general tend to perform well. The basic optimization scheme remains effectively that shown in Figure 6.

One of the best results obtained using this type of control synthesis is for the 'cheetah' creature, shown running in Figure 2. This creature has four actuated joints, as well as two passive, springy joints in the back, connected to link L1 (see Figure 11). The numerical optimization yields running gaits which can take full advantage of the passive springs in the back to store and release energy during a fast bounding gait. The resulting motion, which makes use of no feedback from the environment, is surprisingly fluid. Fully 3D motions such as the walking monster shown in Figure 8 can also be controlled using an automatically-synthesized PCG. Both the cheetah and the monster must rely on passive stability, however.

5.2 Parameterizing motions

The optimization process described thus far is good at producing a single set of control parameters which results in a stable periodic gait. However, how could we control the speed of a gait? One can show that locomotion speed can often be controlled by suitable interpolation between the control parameters for a slow gait and a fast gait[26]. By optimizing a given controller X with respect to some criterion to yield a new controller Y, the controllers obtained by interpolating between the control parameters of X and Y often result in well-behaved, interpolated motions. This is an immediately useful characteristic for animators, and also shows a promising direction towards building complex, parameterizable motor skills.

A different type of parameterization allows us to also work with aperiodic motions. In Figure 12, the goal is to begin with a periodic hopping motion (A), and to use an optimization procedure to change the middle hop into a large leap (B). This can be accomplished by optimizing for the total distance travelled over all the hops, while restricting the optimization to only alter the parameter values associated with the control of the middle hop. Note that it is important to have each trial simulate the three hops following the leap. This ensures that a large leap followed by a fall will always be a suboptimal solution that is rejected. Figure 12(C) illustrates a detail of the final leaping motion, with horizontal displacement being scaled for clarity. Figure 12(D) shows the result of interpolating between the control parameters for a regular hop and those of the leap in (B). As one might hope, the result is an intermediate size hop.

6 Limit cycle control

Simulating human walking has proved to be a surprisingly difficult task, and has long been a subject of fascination[7, 8, 13, 10, 16, 17, 22, 24, 6, 18]. The difficulty arises from its instability, combined with the non-linear and underactuated nature of the dynamics. As a result, a large variety of simplifications are often

introduced to make it a tractable control problem. These include planar models with planar dynamics, the use of reference trajectories and linearization, and adding unrealistic constraints which keep the feet glued in place during a stance phase. While these assumptions have been useful for understanding the nature of walking control, they are all unrealistic in some aspect.

Progress in the area of controlling hopping and running gaits has been better. The plenary work of Raibert[20] and Raibert and Hodgins[21] in controlling hopping robots (both real and simulated), has shown that hopping can be decomposed into three separable control problems, namely those of (1) controlling hop height, (2) controlling forward speed, and (3) controlling body attitude. It has also been shown that a full dynamical simulation of a human can be made to run using this type of control model[11].

The work we briefly summarize here is that of a control method for a complete 3D dynamical simulation of human walking[14, 15]. The human model has realistic proportions, masses, and moments of inertia. The articulated skeleton used for our simulations is shown in Figure 13. The speed, direction, and style of the walk can be controlled. The walks are unfortunately not yet mistakeable for a real human walk because they still have certain robotic qualities to their motion. Nevertheless, we believe that the parameters can be appropriately tuned to achieve a class of simulated, 'natural-looking' walks.

Our approach to controlling a walking gait begins with a cyclic state machine (pose control graph), much like the virtual windup toys. This provides for the basic open-loop stepping actions for a walking motion. An example is shown in Figure 14. The walking motion resulting from this open-loop control has the human figure taking several steps and then falling over. Note that the use of *open loop* is somewhat of a misnomer because PD controllers are in fact being used for individual joints. Designing the control consists of supplying suitable reference positions to the PD controllers used at each joint.

6.1 Linearizing over a limit cycle

The limit cycle method takes a basic open-loop motion and adds the necessary control to produce a stable limit cycle. There exist several possibilities for introducing control into this type of cyclical motion. The key insight into our method is the realization that while the system dynamics are non-linear and potentially discontinuous at many points in time, they are quite well behaved over a complete step. Thus, we discretize the limit cycles as shown in Figure 15.

Given the discrete dynamical system, it is possible to define a new set of input and output variables for this system. The output variables we desire to control are a projection of the full system state. In fact, it is sufficient to provide some type of control over forward-backward pitch and left-right pitch. These types of pitch can be measured using a *pitch-vector* defined in terms of the body coordinates. Several possibilites can be shown to work, as illustrated in Figure 16. We refer to the outputs as *regulation variables*, which in this case correspond to the sagittal and lateral components of the pitch-vector. The

motivation of using a smaller, projected version of the state is one of providing a minimal relevant description of the system dynamics. The reasoning behind the choice of a pitch-vector is intuitive: a walking figure can fall forwards, backwards, to the left, or to the right. In practice, controlling the pitch vector is quite sufficient to yield stable walking motions while still leaving the motion largely unconstrained.

Given the choice of a pair of regulation variables (front-back pitch and lateral pitch), we need to select an appropriate pair of control perturbations to yield a fully controllable system. One set of choices which works well is shown in Figure 17. The control perturbations are applied by altering the references poses in the state machine. The magnitudes of the two potential applied alterations for each step represent the control inputs.

The relationship between the regulation variables and control variables is approximately linear for any given step. However, the nature of the linear relationship may change from step to step — i.e., it is a function of the system state. The control algorithm constructs the appropriate linear model using a total of four simulations of the same step, each executed with different control parameters. Finite differences are then used to construct the linear model.

The resulting limit cycle which is traced by the projection of the pitch vector is shown in Figure 18. The left and right lobes of the figure-8 pattern correspond to the pitch vector swaying slightly to the left and the right during the walking motion. At the end of each step, the pitch vector passes through a 'target' point, which represents the reference input for the discrete control system.

6.2 Results

The limit cycle control technique enables the control of a full dynamic simulation of a 3D human model having realistic proportions and mass distribution. The figure can walk at a selected speed, as well following a given path. The control has also been applied to obtain walking motions for a 'scout' robot shown in Figure 1. Variations on a walk are easily obtained by changing the open-loop control. The closed-loop component of the control, effected by the limit cycle control method, automatically adapts. An example is shown in Figure 19.

7 Conclusions

The availability of sophisticated simulation packages is today making automated numerical synthesis techniques an increasingly flexible and useful approach. Numerical optimization techniques can work well in the face of complex, non-linear phenomena that might otherwise be difficult to cope with analytically.

Simulating human and animal motion is a challenging task from which much can be learned. How can we develop the complete sets of motor skills which a squirrel uses when jumping between branches in a tree, or that a gazelle uses in turning a sharp corner? It is exciting to think of the possible control techniques

which might be useful in solving these kinds of problems. Simulation-based approaches may prove to be a significant investigative tool of the future.

Animating human or animal movements naturally leads into an exploration of many facets of movement planning and execution. One which we have been exploring deals with a highly-unstable, underactuated 'acrobot'[3]. This creature is in effect a two-link robot, which must apply continuous active control to balance itself. With some effort, control strategies for forward locomotion and even front-and-back flips can be developed, as shown in Figure 20.

Lastly, many motions require careful planning to succeed. It is not clear where a control problem ends and a planning problem begins. Figure 21 is an example of this, where Luxo, the hopping lamp, must locomote across a variety of terrain. Initial explorations in this direction using a simulation-based synthesis approach have yielded encouraging results, although there is clearly very much exciting work left.

References

[1] R. M. Alexander. The gaits of bipedal and quadrupedal animals. *International Journal of Robotics Research*, (Summer), 1984.

[2] N. I. Badler, B. Barsky, and D. Zeltzer. *Making Them Move*. Morgan Kaufmann Publishers Inc., 1991.

[3] M. D. Berkemeier and R. S. Fearing. Control experiments on an underactuated robot with applications to legged locomotion. *Proceedings, IEEE International Conference on Robotics and Automation*, pages 149–154, 1994.

[4] A. Bruderlin and L. Williams. Motion signal processing. *Proceedings of Siggaph '95, ACM Computer Graphics*, pages 97–104, 1995.

[5] J. Auslander et al. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Transactions on Graphics*, October 1995.

[6] Vukobratovic et al. *Biped Locomotion: Dynamics, Stability, Control and Applications*. Springer Verlag, 1990.

[7] J. Furusho and M. Maubuchi. A theoretically motivated reduced order model for the control of dynamic biped locomotion. *Journal of Dynamic Systems, Measurement, and Control*, 109:155–163, 1987.

[8] J. Furusho and A. Sano. Sensor-based control of a nine-link robot. *The International Journal of Robotics Research*, 9(2):83–98, 1990.

[9] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. *Proceedings of SIGGRAPH '95, ACM Computer Graphics*, pages 63–70, 1995.

[10] H. M. Hmam and D. A. Lawrence. Robustness analysis of nonlinear biped control laws via singular perturbation theory. *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 2656–2661, 1992.

[11] J. K. Hodgins. Simulation of human running. *Proceedings, IEEE International Conference on Robotics and Automation*, pages 1320–1325, 1994.

[12] Symbolic Dynamics Inc. *SD/Fast User's Manual*. 1990.

[13] R. Katoh and M. Mori. Control method of biped locomotion giving asymptotic stability of trajectory. *Automatica*, 20:405–414, 1984.

[14] J. Laszlo. *Controlling Bipedal Locomotion for Computer Animation*. M.Sc. Thesis, Department of Computer Science, University of Toronto, 1996.

[15] J. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH '96*, pages 155–162, 1996.

[16] T. McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

[17] T. McGeer. Passive walking with knees. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1640–1645, 1990.

[18] H. Miura and I. Shimoyama. Dynamic walk of a biped. *International Journal of Robotics Research*, pages 60–74, Summer 1984.

[19] J. T. Ngo and J. Marks. Spacetime constraints revisited. *Proceedings of SIGGRAPH '93*, pages 343–350, 1993.

[20] M. H. Raibert. *Legged Robots that Balance*. MIT Press, 1986.

[21] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Proceedings of SIGGRAPH '91*, pages 349–358, 1991.

[22] F. Saito, T. Fukuda, and F. Arai. Swing and locomotion control for a two-link brachiation robot. *IEEE Control Systems*, 14(1):5–12, 1994.

[23] K. Sims. Evolving virtual creatures. *Proceedings of SIGGRAPH '94, ACM Computer Graphics*, pages 15–22, 1994.

[24] A. J. Stewart and J. F. Cremer. Beyond keyframing: An algorithmic approach to animation. *Proceedings of Graphics Interface '92*, pages 273–281, 1992.

[25] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Proceedings of SIGGRAPH '95, ACM Computer Graphics*, pages 91–96, 1995.

[26] M. van de Panne. Parameterized gait synthesis. *IEEE Computer Graphics and Applications*, pages 40–48, March 1996.

[27] M. van de Panne and E. Fiume. Sensor-actuator networks. *Proceedings of SIGGRAPH '93*, pages 335–342, 1993.

[28] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. *Proceedings of Graphics Interface '94*, pages 208–215, 1994.

[29] A. Witkin and Z. Popovi'c. Motion warping. *Proceedings of SIGGRAPH '95*, pages 105–107, 1995.

Figure 6: Generate-and-test control synthesis.

```
generate and evaluate 200 random controllers
for each of the 10 best
  for 1000 trials
    randomly choose a parameter to vary
    perturb the parameter value by +delta or -delta
    evaluate the new controller by simulation
    if movement has improved then
      keep change
    else
      reject change
  end for
end for
output: 10 controllers, each providing a potentially
        unique mode of locomotion
```

Figure 7: Pseudocode for generate-and-test optimization

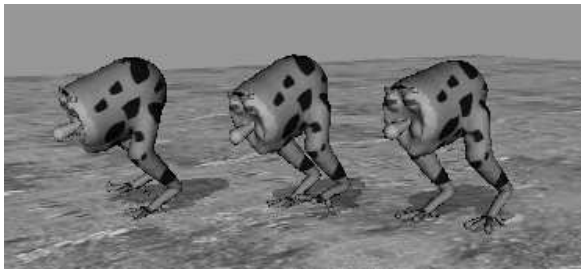


Figure 8: A simulated windup toy: the monster

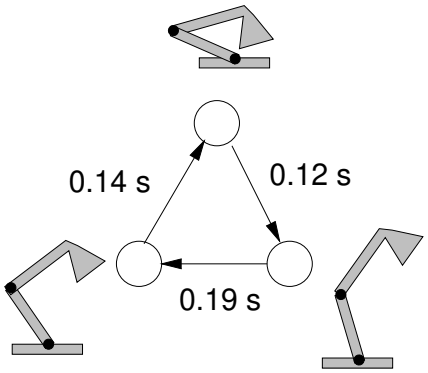


Figure 9: A pose control graph.

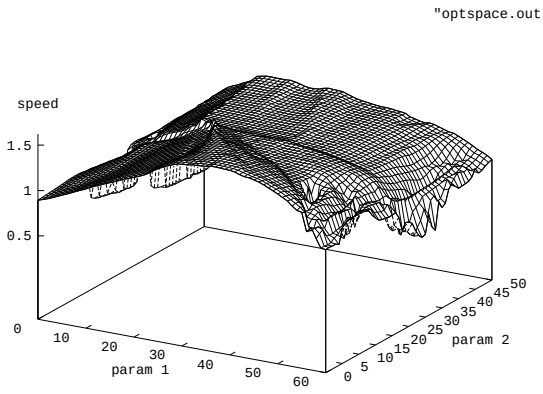
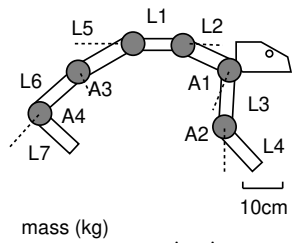


Figure 10: Optimization space for a cat trot.

mechanical configuration:



link	mass (kg)
L1	0.30
L2	0.35
L3	0.15
L4	0.10
L5	0.35
L6	0.15
L7	0.10

actuators:

act.	min	max	ks	kd
A1	−45	45	0.4	0.01
A2	−90	0	0.4	0.01
A3	40	70	0.4	0.01
A4	−120	−30	0.4	0.01

Figure 11: The cheetah creature.

Figure 12: Optimizing a Luxo leap.

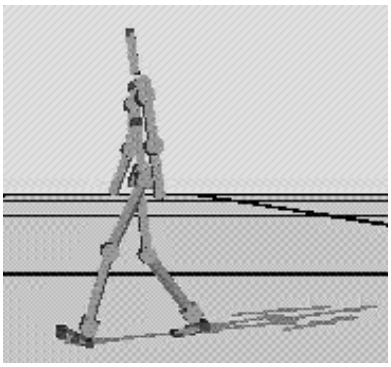


Figure 13: The human model.

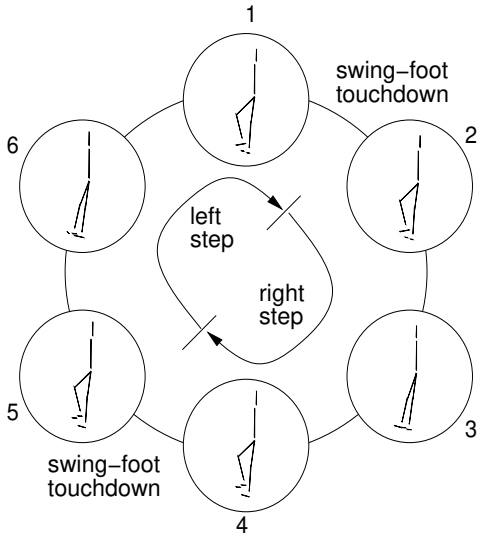


Figure 14: Finite state machine employed for walking.

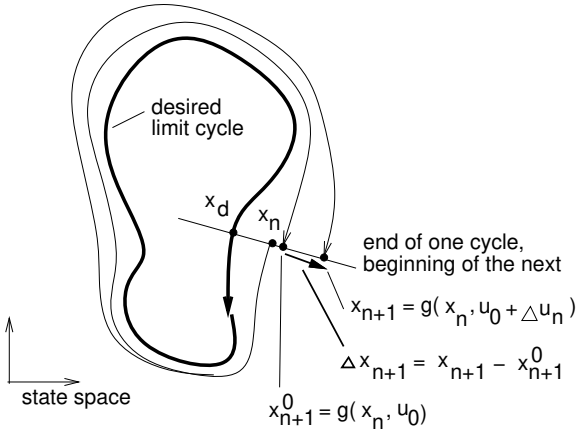


Figure 15: A limit cycle as a discrete dynamical system.

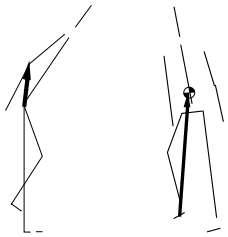


Figure 16: Regulation variables for use in walking.

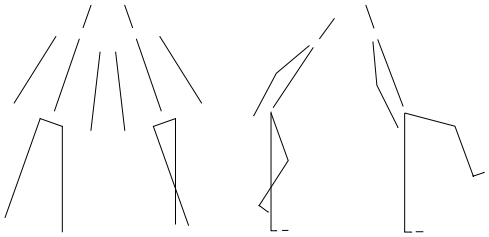


Figure 17: Control perturbations used for walking.

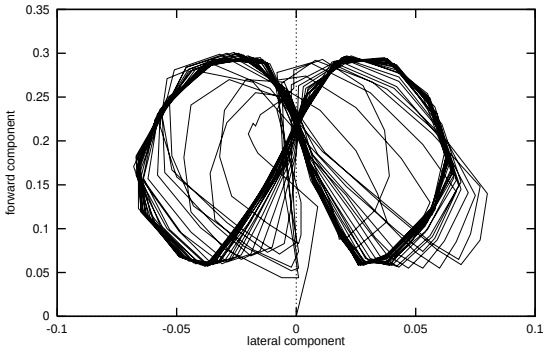


Figure 18: Regulation variable limit cycle.

Figure 19: Stylistic variation on a walk.

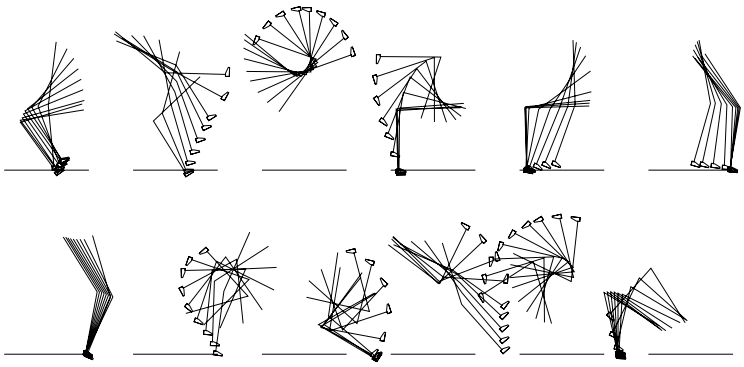


Figure 20: A Flipping Acrobot.

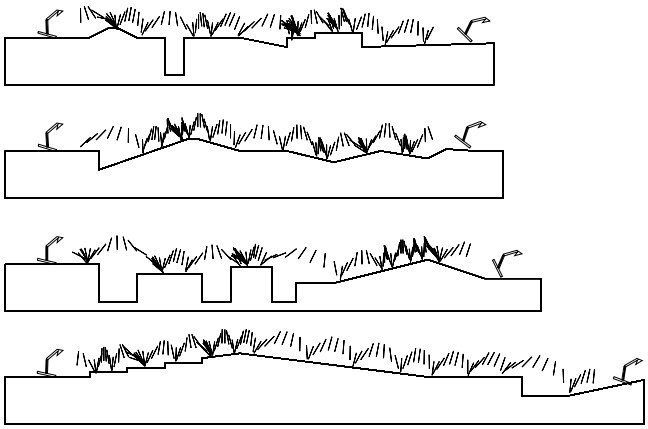


Figure 21: Synthesized control for Luxo on cross-countryruns.