Progress Report to Office of Naval Research

Dynamic Legged Locomotion in Robots and Animals

Marc Raibert, Robert Playter, Robert Ringrose
Dave Bailey, Karl Leeser

The Leg Laboratory

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

January 1995

Abstract

This report documents our study of active legged systems that balance actively and move dynamically. The purpose of this research is to build a foundation of knowledge that can lead both to the construction of useful legged vehicles and to a better understanding of how animal locomotion works. In this report we provide an update on progress during the past year. Here are the topics covered in this report:

- *Is Cockroach Locomotion Dynamic?* —To address this question we created three models of cockroaches, each abstracted at a different level. We provided each model with a control system and computer simulation. One set of results suggests that "Groucho Running," a type of dynamic walking, seems feasible at cockroach scale.
- *How Do Bipeds Shift Weight Between the Legs?* — We built a simple planar biped robot specifically to explore this question. It shifts its weight from one curved foot to the other, using a toe-off and toe-on strategy, in conjunction with dynamic tipping.
- *3D Biped Gymnastics*—The 3D biped robot has done front somersaults in the laboratory. The robot changes its leg length in flight to control rotation rate. This in turn provides a mechanism for controlling the landing attitude of the robot once airborne.
- *Passively Stabilized Layout Somersault*—We have found that the passive structure of a gymnast, the configuration of masses and compliances, can stabilize inherently unstable maneuvers. This means that body biomechanics could play a larger role in controlling behavior than is generally thought. We used a physical "doll" model and computer simulation to illustrate the point.
- *Twisting*—Some gymnastic maneuvers require twisting. We are studying how to couple the biomechanics of the system to its control to produce efficient, stable twisting maneuvers.
- *Automatic Tuning of Control Systems*—We have developed a computer program that automatically adjusts the control parameters of a simulated legged creature to accommodate changes in mass, geometry, or desired behavior. The program has been successfully tested on a simulated planar quadruped.

Contents

Chapter 1

Introduction and Summary

This report documents our recent research on the dynamics of legged locomotion. The work was done in the Leg Laboratory, part of the MIT Artificial Intelligence Laboratory. A premise for this research is that active balance and dynamics are important for the control of legged systems, robots and animals alike. Systems that balance actively can use footholds that are widely separated, they can move along narrow paths where a broad base of support is not available, and they can use their kinetic energy as a bridge to increase their effective size. Dynamics is a key ingredient in the behavior of animals and it will weigh heavily in the development of useful legged vehicles. A second premise is that the mechanical design of a legged system (or its biomechanics) can be just as important to its behavior as is the computer that controls it. Mechanics and control will work together to produce behavior in the best systems.

A dynamic treatment need not be an intractable treatment. We have found that simple algorithms can provide balance and control for a variety of dynamic legged systems. The machines we have studied include a planar one-legged hopping machine, a three-dimensional one-legged hopping machine, a planar biped running machine, a three-dimensional biped running machine, a quadruped, a planar monopod with rotary hip and ankle joints, a planar kangaroo-like monopod with tail and rotary joints, and a machine that runs in zero-gravity. The techniques used to control each of these machines derive from a single set of control algorithms, modified in various ways. These algorithms have been adapted for hopping, pronking, biped running, fast running, trotting, pacing, bounding, and simple gymnastic maneuvers. The ability of simple algorithms to operate under these diverse circumstances suggests the algorithm's fundamental nature.

In addition to work on legged machines, we have begun to apply techniques from robotics to computer simulation and animation of animal behavior. This includes automatic generation of simulation programs and graphics, task-level control of behavior, and automatic tuning of control parameters. The remainder of this report is a collection of papers that describes these projects. They are summarized in the following paragraphs.

Table 1–1: Milestones of the Leg Laboratory.

1982	Planar one-legged machine hops in place, travels at a specified rate, keeps its balance when disturbed, and jumps over small obstacles.
1983	Three-dimensional one-legged machine runs and balances on an open floor. Simulations reveal passively stabilized bounding gait for quadruped-like model.
1984	Quadruped runs with trotting gait using generalization of one-leg algorithms. Data from cat and human found to exhibit symmetries like those used to control running machines.
1985	Planar biped runs with one- and two-legged gaits and changes between gaits.
1986	Planar biped does flips, aerials, and runs at 11.5 mph. Monopod runs using leg with rotary joint and leaf-spring foot.
1987	Quadruped runs with trotting, pacing, and bounding gaits.
1988	Planar biped uses selected footholds and climbs short stairway. Quadruped does rudimentary transitions between running gaits. Reentrant trajectories found for simulated passive dynamic running. Computer simulation shows running in zero-g by bouncing between two floors. Simple physical model of zero-g running demonstrated.
1989	Planar biped jumps through hoop and runs with top speed of 13.1 mph. Simulated planar biped gallops. Hoof added to monopod improves performance of leafspring foot. Three-dimensional biped does simple running.
1990	Planar biped robot walks, gallops, and changes between walking and running. Simulation of passive dynamic running with knees.
1991	Kangaroo-like robot, with articulated leg and tail, takes first steps (hops). Computer simulations of simplified kangaroos and ostriches.
1992	3D biped robot does somersault. Kangaroo-like robot runs at 1.8 m/s. Simulated quadruped control system adjusts automatically for variations in body mass and leg length.
1993	3D Biped and Kangaroo appear with Sean Connery and Wesley Snipes in *Rising Sun.* Passive stability achieved for layout somersault in laboratory "doll".
1994	Cockroach simulations runs using kinematic motion data from animal. Stable *Groucho* running in simulated spherical cockroach (Hexahopper). Weight-shifting robot rocks with stability.

Figure 1–1: The Leg Laboratory was located at Carnegie-Mellon University from 1981–1986, and at the Massachusetts Institute of Technology from 1987–present.

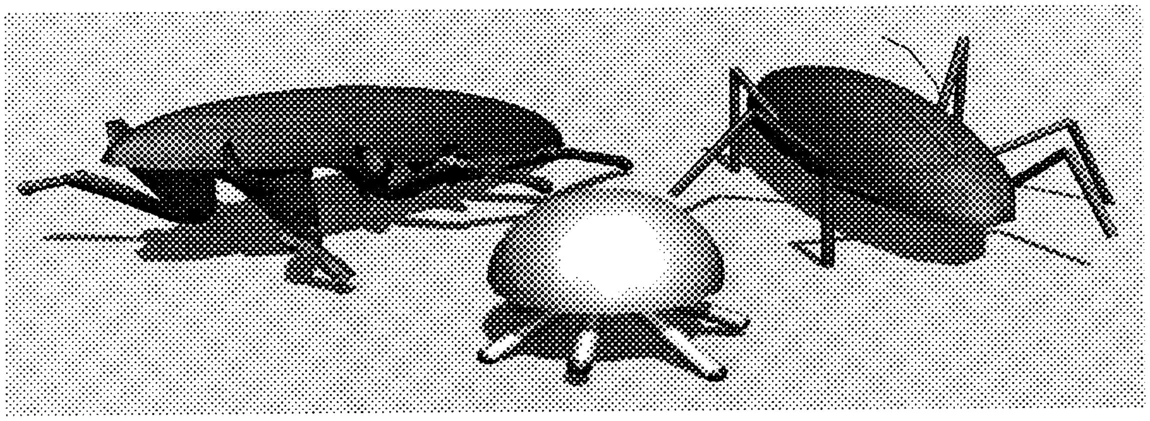


Figure 1–2: *Hexahopper*, *Hexabug*, and *Playback Cockroach,* three models used for simulation of cockroach locomotion.

Dynamic Cockroach Locomotion

Because cockroaches have small size and a broad base of support, it is commonly assumed that they use static techniques for locomotion. This assumption is based on the idea that inertial forces at cockroach scale are small and overwhelmed by frictional and surface tension forces. We are interested exploring this assumption. We were motivated by Robert Full's suggestion, based on laboratory observations, that cockroaches frequently move dynamically. To address the issue we created several physics-based computer simulations of six-legged creatures at cockroach scale and experimented with making them run. One simulation, the *Hexahopper,* is significantly abstracted from the animal cockroach design: it has telescoping legs that are attached to a hemispherical body at the center of mass. A second simulation, the *Hexabug* uses articulated legs that are attached to an elongated body at physiologically motivated positions. A third computer simulation, called the *Playback Cockroach,* is based on two kinds of data obtained (by Full) from animal cockroaches: data describing the geometry of each leg link was obtained from histology, and data for kinematic motion of the limbs during locomotion was obtained from high-speed film. Results from these simulations suggest that it is feasible for cockroaches to locomote dynamically, but they do not need to actively balance.

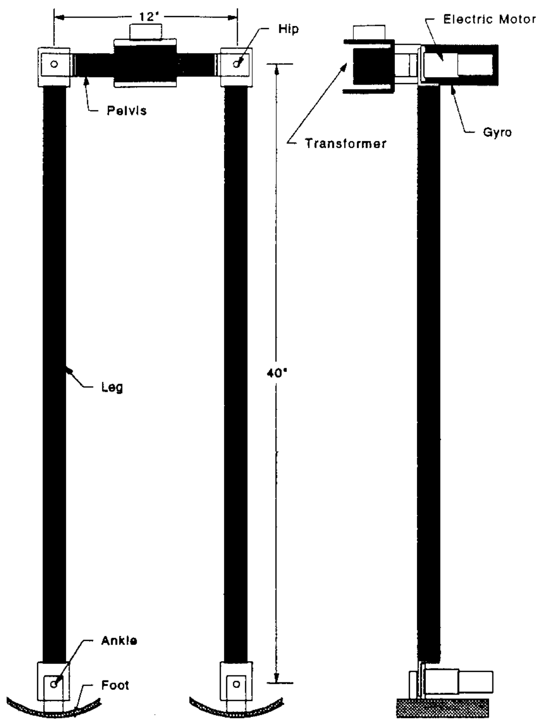


Figure 1–3: Biped Robot used for studying transfer of support from one foot to the other. It stands approximately 1.1 m tall and weights about 7 Kg. The four actuators are electric motors located at the various joints. Computing and electrical power is off board. The machine walks by rocking from side to side.

Weight-Shifting Biped

A truly graceful walking robot does not yet exist. We believe that a key ingredient for graceful walking is the smooth transfer of support, from one leg to the other. This project begins address what's required for a smooth step and graceful walk. The goal is to study the transfer of support between the feet during a dynamic walking gait. We built a biped robot apparatus for the work. It consists of five links: a pelvis, two legs, and two feet. The links are connected by revolute joints which in turn are actuated by electric motors. This robot walks by rocking from side to side, transferring support from one foot to the other. We developed a closed-loop method for controlling dynamic rocking. The controller's aim is to regulate the energies during the different states of the rocking cycle. Some initial steps have been taken towards smoothing the exchange of support, and the results look promising.

Biped Somersault in 3D

Making a legged robot perform a somersault requires us to focus on balance and control

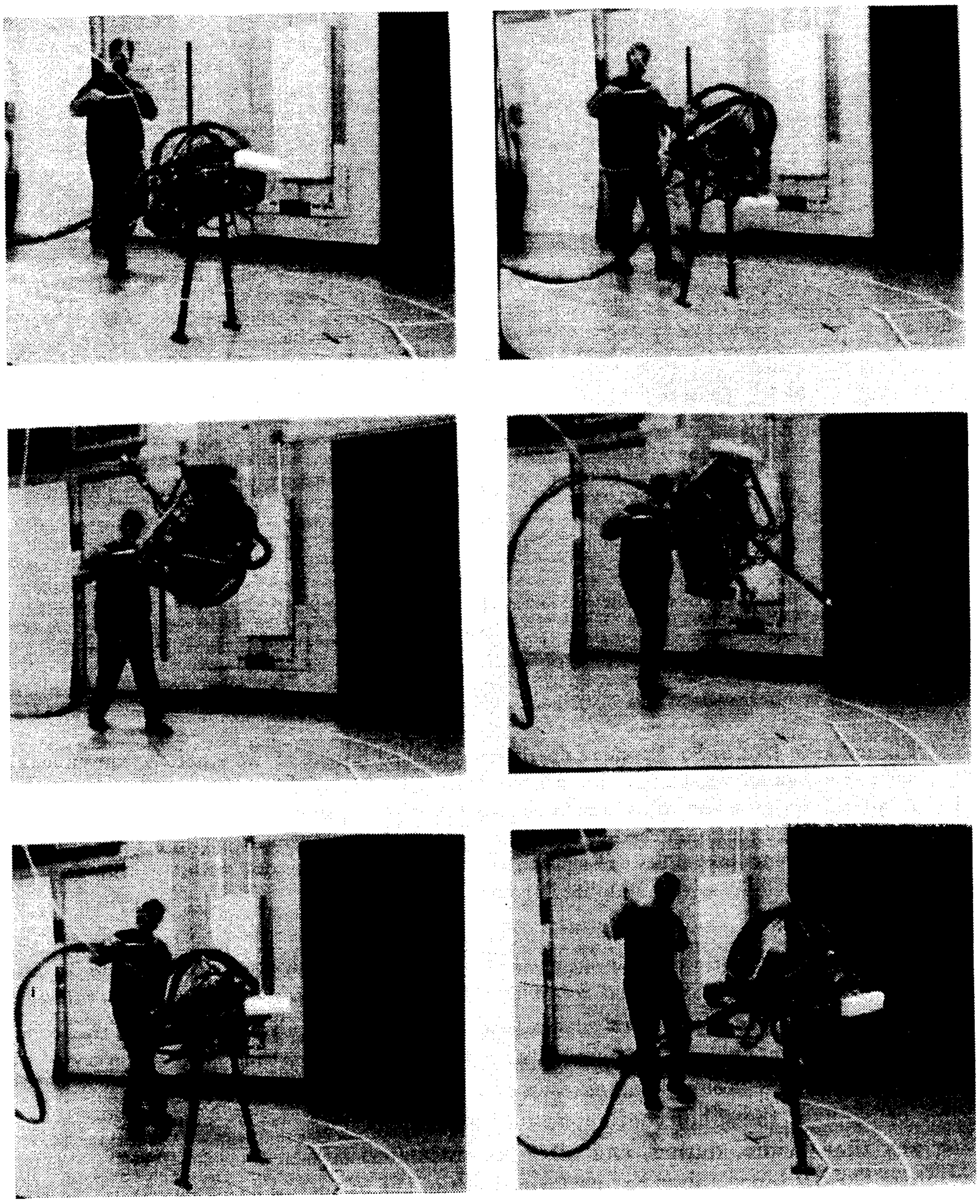


Figure 1–4: 3D Biped Doing a Somersault. The amount of leg tuck (telescopic shortening) is servoed during the flight phase of the maneuver to control rotation rate. The legs are spread slightly on landing to help control roll. Hydraulic accumulators were added to the machine to give it high-enough instantaneous power to do the trick. The photographs are arranged in sequence left to right, starting at the upper left.

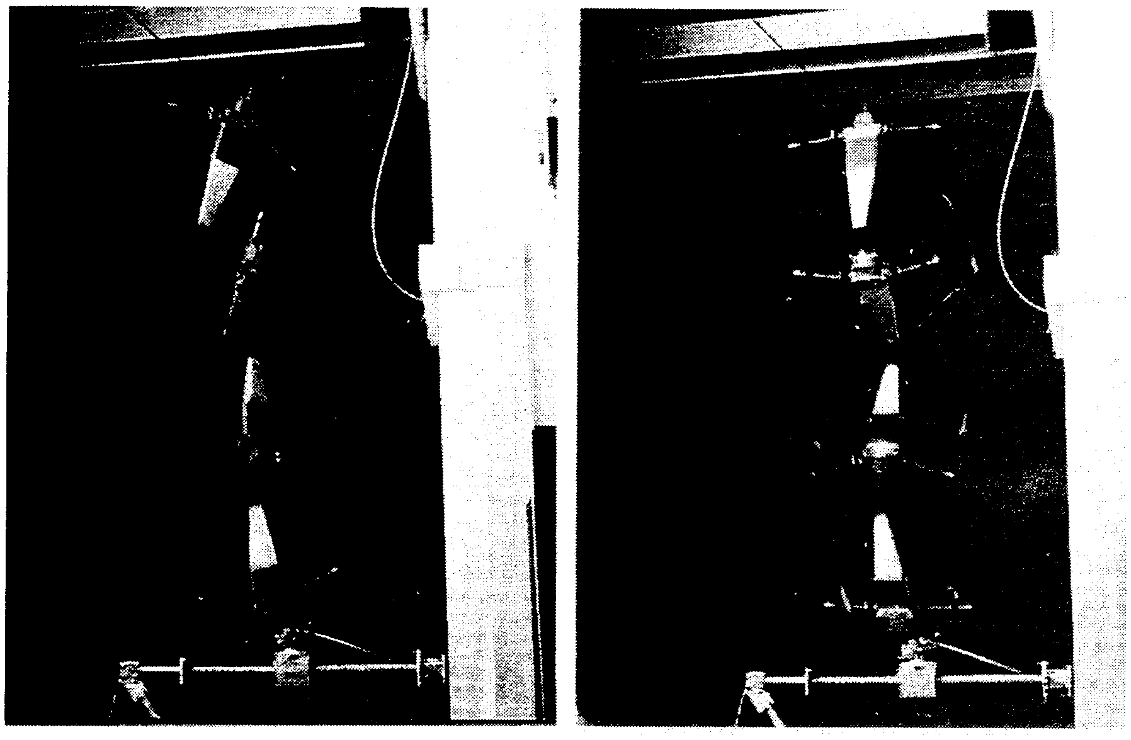


Figure 1–5: Layout Somersault Experiment. Strobed images of two separate experiments with a mechanical "doll". The doll has two "arms" that have spring-driven shoulder joints. A launching mechanism rotate's the doll about its pitching (somersaulting) axis and throws it upward. The photo on the left show the doll's behavior with arms clamped in place: it exhibits the twist instability. The photo on the right shows the doll with the arms free to flap, and thereby passively stabilized against twisting.

at a higher level of detail than for constant running. To learn something of these details we programmed a 3D Biped robot to perform forward somersaults in the laboratory. The control employs a feedback mechanism to regulate rotation rate while the robot is airborne. In flight, the robot extends or retracts its legs to change its rotation rate in order to produce a desired landing attitude. To compensate for uncertainty in the exact time of landing, the robot continuously adjust its configuration to be prepared for landing any time before, during, and after the estimated landing time. Using this control strategy the 3D biped robot has performed a number of successful somersaults in the laboratory. On the best day so far the robot executed seven successful somersaults out of ten attempts.

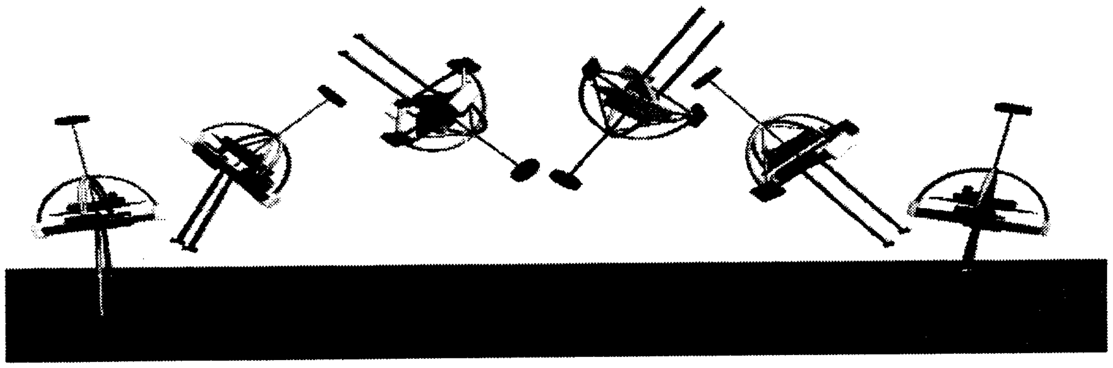


Figure 1–6: Simulated 3D Biped Somersault with One Half Twist. Twisting was initiated by scissoring the legs just before takeoff. A weight was added to the trunk (shown above the trunk) to increase the aspect ratio of the moment of inertia.

Passively Stabilized Layout Somersault

The layout or straight body somersault is considered to be inherently unstable because it requires rotation about the middle principal axis of inertia, an unstable equilibrium for a rigid body. Despite this fact, athletes regularly perform this maneuver with apparent ease. Previously, biomechanics researchers have assumed that the athlete senses the instability of the maneuver and actively compensates for it with movements of the arms and body. I show that the layout somersault can be a passive, neutrally stable maneuver. Passive stabilization of the layout somersault results from the natural dynamic interaction of the limbs and body during movement. I verify this result with linear stability analysis, non-linear dynamic simulation, and experiments on a human-like doll built and tested in the laboratory. The doll has spring-driven arms but has no other control system, sensors, or actuators. The doll routinely exhibits triple somersaults about its middle principal axis without twisting.

Twisting Maneuvers

While twisting is to be avoided in the layout somersault, it is a feature in other tricks. We present results from dynamic simulation of a human performing a front somersault with one and a half twists. The twisting maneuver is started from a front somersault by reconfiguring the body mid-maneuver. While a sequence of prescribed limb movements can be found to produce a twisting maneuver, simulation results show that the performance is sensitive to small variations in initial conditions. Reliability of the maneuver can be significantly improved by using open loop torque control with a compliant passive dynamic model of the human, rather than the prescribed movements.

We also present results from computer simulation of a front somersault with one half twist performed by the 3D Biped. We found that to make the simulated 3D Biped perform a satisfactory front somersault with one half twist we had to add weight to the robot to make its inertia more like that of a human.

Tuning Control for Model Variations

Elephants, giraffes, horses and dogs have similar topology but widely varied scales and proportions. A single well constructed control system might be able to control all of them, given the proper control parameters. We have developed a computer program which tunes the control parameters for a complex dynamic system automatically to adapt it to different physical characteristics or to a new behavior. In simulation tests, the tuning algorithm has successfully re-tuned a simulation of a planar quadruped to accommodate an increase in body mass by a factor of three, a reduction in leg length by a factor of two, and an increase in foot weight by a factor of ten.

Leg Laboratory Bibliography

More background information for the work described in this report can be found in the following publications.

Borvansky, L., 1991. *Dynamic Simulation of the Running Motion of an Ostrich.* SM Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Mass.

Brown, H. B. Jr., Raibert M. H. 1986. Legs that deform elastically, In *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy'86*, A. Morecki, G. Bianchi, K. Kedzior (eds.) (MIT Press, Cambridge).

Goldberg, K. Y., Raibert, M. H. 1987. Conditions for symmetric running in single and double support. *Proceedings of IEEE International Symposium on Robotics*, Raleigh, North Carolina.

He, J., Kram, R., McMahon, T., 1991. Mechanics of running under simulated low gravity, *J. Applied Physiology*, 71:3.

He, J., Raibert, M. H. 1990. Behaviors of a simulated 3D biped. In *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy'90*, A. Morecki, G. Bianchi, K. Kedzior (eds.).

Hodgins, J. 1988. Legged robots on rough terrain: experiments in adjusting step length. In *Proceedings of the IEEE International Conference on Robotics and Automation,* Philadelphia.

Hodgins, J. 1989. *Legged Robots on Rough Terrain: Experiments in Adjusting Step Length.* Ph.D Thesis, Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Hodgins, J. 1991, Biped gait transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation* Sacramento.

Hodgins, J., Raibert, M. H. 1987. Planar Biped Goes Head Over Heels. In *Proceedings ASME Winter Annual Meeting* Boston.

Hodgins, J., Raibert, M. H. 1987. Biped Gymnastics, In *Fourth International Symposium of Robotics Research*, B. Bolles, B. Roth (eds.), (MIT Press, Cambridge).

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

Hodgins, J., Raibert, M. H., 1991. Adjusting step length for rough terrain locomotion, *IEEE J. Robotics and Automation* 7:3:289–298.

Hodgins, J., Koechling, J., Raibert, M. H. 1986. Running experiments with a planar biped. *Third International Symposium on Robotics Research*, G. Giralt, M. Ghallab (eds.). (MIT Press, Cambridge).

Hodgins, J., Koechling, J., Raibert, M. H. 1991. Running Experiments with a Planar Biped, *CMU Computer Science 25th Anniversary*, (New York:Addison-Wesley.)

Koechling, J. 1989. *The Limits of Running Speed: Experiments with a Legged Robot.* Ph.D Thesis, Mechanical Engineering Department, Carnegie Mellon University, Pittsburgh.

Koechling, J., Raibert, M. 1988. How fast can a legged robot run? In: *Symposium in Robotics, DSC-Vol. 11*, K. Youcef-Toumi, H. Kazerooni (eds). (American Society of Mechanical Engineers, New York).

Lee, W., Raibert, M. H., 1991. Control of hoof rolling in an articulated leg. *IEEE Robotics and Automation Conference*, Sacramento.

Murphy, K. N., Raibert, M. H. 1985. Trotting and bounding in a planar two-legged model. In *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy'84*, A. Morecki, G. Bianchi, K. Kedzior (eds.). (MIT Press, Cambridge), 411–420.

Murthy, S. S., Raibert, M. H. 1983. 3D balance in legged locomotion: modeling and simulation for the one-legged case. In *Inter-Disciplinary Workshop on Motion: Representation and Perception*, ACM.

Playter, R. R. 1994. Passive Dynamics in the Control of Gymnastic Maneuvers, Ph.D Thesis, Aeronautical and Astronautical Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Playter, R. R., Raibert, M. H. 1992. Generation of a somersault by a 3D biped robot. *IFAC '92*, Japan.

Playter, R. R., Raibert, M. H. 1992. Control of a biped somersault in 3D. *Proceedings of IROS '92*, Raliegh.

Playter, R. R., Raibert, M. H. 1992. Control of a Biped Somersault in 3D. *Proceedings, IFToMM-jc Int. Symp. on Theory of Machines and Mechanisms*, Nagoya, Japan, 669–674.

Playter, R. R., Raibert, M. H. 1994. Passively Stable Layout Somersaults. *Canadian Society For Biomechanics, Proceedings of Eight Biennial Conference and Symposium the*, Calgary, Alberta, Canada.

Playter, R. R., Raibert, M. H., 1994. Passively Stable Layout Somersaults. *Eighth Yale Workshop on Adaptive and Learning Systems, June 13-15, 1994, New Haven, CT. USA* 66-71.

Raibert, M. H. 1983. Dynamic stability and resonance in a one-legged hopping machine. In *Theory and Practice of Robots and Manipulators, Proceedings of RoManSy'81*, A. Morecki, G. Bianchi, K. Kedzior (eds.). Warsaw: Polish Scientific Publishers. 352–367.

Raibert, M. H. 1984. Hopping in legged systems—Modeling and simulation for the 2D one-legged case. *IEEE Trans. Systems, Man, and Cybernetics* 14:451–463.

Raibert, M. H., (Ed.) 1984. Special Issue on Legged Locomotion, *International Journal of Robotics Research*, 3:2.

Raibert, M. H., (Ed.) 1984. Walking Machine Video, Videotape appendix to Special Issue on Legged Locomotion, *International Journal of Robotics Research*, (MIT Press: Cambridge Mass.)

Raibert, M. H. 1985. Four-legged running with one-legged algorithms. In *Second International Symposium on Robotics Research*, H. Hanafusa, H. Inoue (eds.), (MIT Press, Cambridge), 311–315.

Raibert, M. H. 1986. *Machines That Run (Videotape)*, (MIT Press: Cambridge Mass.)

Raibert, M. H. 1986. *Legged Robots That Balance* (MIT Press, Cambridge).

Raibert, M. H. 1986. Symmetry in running. *Science*, 231:1292–1294.

Raibert, M. H. 1986. Legged robots. *Communications of the ACM* 29:499–514.

Raibert, M. H. 1986. Running with symmetry. *International Journal of Robotics Research* 5:3–19.

Also reprinted in *Autonomous Robot Vehicles*, 1990, I.J. Cox, G.T. Wilfong (Eds.) (Springer-Verlag, New York).

Raibert, M. H., 1989. Legged Robots, Chapter in *Robotics Science* M. Brady (ed.), (MIT Press, Cambridge).

Raibert, M. H., 1991. Trotting, pacing, and bounding by a quadruped robot, *Journal of Biomechanics.*

Raibert, M. H., Brown, H. B., Jr. 1984. Experiments in balance with a 2D one-legged hopping machine. *ASME J. Dynamic Systems, Measurement, and Control* 106:75–81.

Raibert, M. H., Hodgins, J. K., Legged robots. *Encyclopedia of Artificial Intelligence*, (New York: John Wiley and Sons.)

Raibert, M. H., Hodgins, J., 1991. Animation of dynamic legged locomotion *SIGGRAPH '91*, Las Vegas, 25:4, 349–358.

Raibert, M. H., Hodgins, J., Playter, R. R., Ringrose, R. P. 1992. Animation of maneuvers: jumps, somersaults, and gait transitions. *Imagina*, Monte Carlo.

Raibert, M.H., Hodgins, J.K., Playter, R.R., Ringrose, R.P. 1993. Animation of legged maneuvers: jumps, somersaults, and gait transitions. In *Journal of the Robotics Society of Japan*, 11(3):333–341.

Raibert, M. H., Hodgins, J., Ringrose, R., Playter, R., Borvansky, L., Campbell, L., Evans, D., Crane, C.A., Lamb, M. 1991. "On The Run", SIGGRAPH '91 Electronic Theater Las Vegas.
Also shown at *Los Angeles International Animation Celebration*, October 1991.
Also shown at the *London Computer Animation Festival*, October 1991.
Also shown at *Imagina*, Monte Carlo, January 1992.

Raibert, M. H., Sutherland, I. E. 1983. Machines that walk. *Scientific American* 248:44–53.

Raibert, M. H., Wimberly, F. C. 1984. Tabular control of balance in a dynamic legged system. *IEEE Trans. Systems, Man, and Cybernetics* 14:334–339.

Raibert, M. H., Brown, H. B., Jr., Murthy, S. S. 1984. 3D balance using 2D algorithms? In *First International Symposium of Robotics Research*, M. Brady, R. P. Paul (eds.), (MIT Press, Cambridge), 279–301.
Also in *Robotics and Artificial Intelligence*, M. Brady, L. A. Gerhardt, H. F. Davidson, (Eds.) (Springer-Verlag, New York), 1984.

Raibert, M. H., Brown, H. B., Jr., Chepponis, M. 1984. Experiments in balance with a 3D one-legged hopping machine. *International J. Robotics Research* 3:75–92.

Raibert, M. H., Chepponis, M., Brown, H. B. Jr. 1986. Running on four legs as though they were one. *IEEE J. Robotics and Automation*, 2:70–82.

Raibert, M. H., Hodgins, J., Playter, R. R., Ringrose, R. P. 1992. Animation of maneuvers: jumps, somersaults, and gait transitions. *Imagina*, Monte Carlo.

Raibert, M. H., Brown, H. B., Jr., Chepponis, M., Hastings, E., Shreve, S. T., Wimberly, F. C. 1981. *Dynamically Stable Legged Locomotion, First Annual Report.* CMU–RI–81–9, Robotics Institute, Carnegie-Mellon University.

Raibert, M. H., Brown, H. B., Jr., Chepponis, M., Hastings, E., Murthy, S. S., Wimberly, F. C. 1983. *Dynamically Stable Legged Locomotion–Second Annual Report.* CMU–RI–TR–83–1, Robotics Institute, Carnegie-Mellon University.

Raibert, M. H., Brown, H. B. Jr., Chepponis, M., Hastings, E., Koechling, J., Murphy, K. N., Murthy, S. S., Stentz, A. 1983. *Dynamically Stable Legged Locomotion—Third Annual Report.* CMU–RI–TR–83–20, Robotics Institute, Carnegie-Mellon University.

Raibert, M. H., Brown, H. B., Jr., Chepponis, M., Hodgins, J., Koechling, J., Miller, J., Murphy, K. N., Murthy, S. S., Stentz, A. J. 1985. *Dynamically Stable Legged Locomotion—Fourth Annual Report.* CMU–LL–4–1985, Carnegie-Mellon University.

Ringrose, R. 1992. *Simulated Creatures: Adapting Control for Variations in Model or Desired Behavior.* Master Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Ringrose, R. 1994. Automatically Tuning Control Systems for Simulated Legged Robots. *Proceedings of AAAI'94*, Seattle.

Thompson, C. M., Raibert, M. H., 1989. Passive dynamic running, In *International Symposium of Experimental Robotics*, Hayward, V., Khatib, O. (eds.), (Springer-Verlag, New York).

Zeglund, G. J. 1991. '*Uniroo: A One-Legged Dynamic Hopping Robot.* Bachelor Thesis, Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Chapter 2

Dynamic Cockroach Locomotion

Karl Leeser and Robert Ringrose

2.1 Introduction

This paper investigates cockroach locomotion. Cockroaches have six legs that provide a broad base of support and, compared to humans, they are small. For these reasons, people tend to think of cockroach locomotion as being static. Full and Tu, however, suggest that cockroaches move dynamically (Full & Tu 1990). This conclusion was motivated by observed phase relationships between horizontal kinetic energy and gravitational potential energy of the center of mass. Based on these energy phase measurements, Full suggests that cockroaches locomote dynamically, using essentially the same techniques as humans and other larger animals. A goal of this research is to test the idea that cockroach locomotion is dynamic. It is part of the larger goal of understanding the general principles of legged locomotion, especially locomotion that is dynamic.

The approach used in this paper is to create models of cockroaches that capture various elements of their design, to build control systems that regulate locomotion behavior, and to simulate the controlled models using numerical methods. Using these tools we demonstrate some viable techniques for cockroach locomotion, and apply them to explore the feasibility of Full's suggestions.

We created three models of the cockroach and built control systems that made them walk and run. We call these models the *Hexahopper, Hexabug,* and *Playback Cockroach.* The control systems for the Hexahopper and Hexabug use finite state machines and algorithms for support, balance, and posture to produce a six-legged alternating tripod gait. They build on previous work done in the Leg Lab based on algorithms for regulating support, balance, and posture. This previous work resulted in a series of laboratory robots, including one-legged hoppers, biped runners, a quadruped, and two kangaroo-like robots. The robots traversed simple paths, ran with several different gaits (hop, run, trot, pace, bound), ran fast (13 mph), jumped over obstacles, climbed a simplified stairway, and did front somersaults(Raibert 1986)(Raibert 1990)(J. Hodgins 1986)(Hodgins & Raibert 1991). Although no single machine performed all these tasks, the machines all use a common set of balance and control principles.

The Hexahopper and Hexabug use a dynamic flightless running gait that McMahon calls *groucho running* when performed by humans (McMahon, Valiant, & Frederick 1987). The controller we built implements groucho running in the two cockroach models. There is no guarantee that animals use a control system like the ones we built, but study of these systems gives us an understanding of the broad issues relevant to the control, stability, and dynamics of locomotion at cockroach scale.

The Playback Cockroach model attempts to model the animals more directly. Data describing the geometry of a cockroach animal's legs were used to create leg models with good fidelity to the anatomy and structure of a real cockroach animal. Data showing the kinematic motion of a cockroach animal walking was used as the basis for locomotion control. The recorded motion data were *played back* through the control system and simulated dynamic model to see what would happen. The Playback Cockroach controller uses the recorded motion data to drive servos at each hip and leg joint, applying torques to approximate the desired joint motion. We used the Playback Cockroach to investigate the dynamic properties of cockroach locomotion.

These simulations of cockroach-sized creatures succeeded in mimicking some of the dynamic behavior of cockroaches. The Hexabug controller produced groucho-running over a wide spectrum of running speeds, up to 52 cm/sec. We combined several control techniques to simplify the coordination of multiple legs, resolving issues of force distribution and position control. The Playback Cockroach produced stable dynamic running over a range of playback speeds up to 44 cm/sec. The Playback Cockroach's gait remained stable over a wide range of playback speeds, (from 0 to 4 times the recorded speed) without adjustment of other parameters. Therefore, we believe that the Playback Cockroach's running gait, while dynamic in the energy sense, may not be dynamic in the balance sense. The stability of the Playback Cockroach simulation suggests a mechanism for dynamically stable running over a wide range of speeds without complex control.

In the sections that follow we describe the three simulations, their control systems, and present some data from their use. Then we return to the question of how best to model cockroach locomotion, as a static or dynamic problem.

2.2 Hexahopper and Hexabug Simulations

We are exploring the potential for dynamic locomotion in a cockroach. We developed two cockroach-sized simulations, both with broad bases of support similar to a cockroach. In these simulations, we did not attempt to do more than approximate the shape of an actual cockroach, although the simulations do obey the laws of Newtonian physics. The controllers are based on previous legged locomotion control systems we have previously built (Raibert *et al.* 1992) (Raibert, Chepponis, & Brown 1986) (Raibert 1985) (J. Hodgins 1986) (M. H. Raibert 1984).

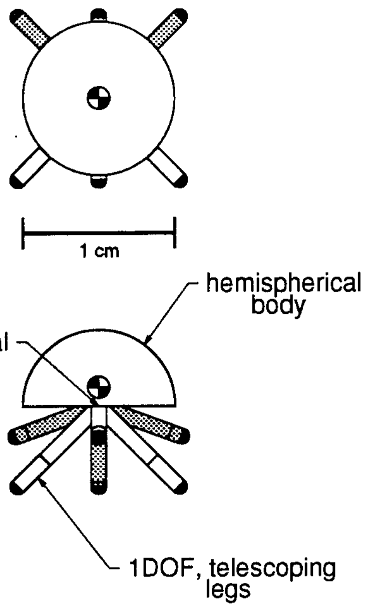


Figure 2–1: Schematic of simulated Hexahopper.

2.2.1 Hexahopper Simulation

We developed the Hexahopper as a test platform for six-legged dynamic locomotion. The Hexahopper bears little physical resemblance to a cockroach. However, its simple dynamics retain the essential elements necessary to study dynamic locomotion in a cockroach-sized, six-legged creature. The simulated creature has a hemispherical body with six legs attached to the center of the lower surface of the body. Each leg has a 3 degree-of-freedom, spherical, ball joint at the hip and a telescoping lower section. Because the hips are close to the center of mass, we assumed that axial leg forces produced negligible resultant body torques, simplifying control of the legs. Figure 2–1 depicts the creature model used in the Hexahopper simulations. Simulation parameters for the Hexahopper model are in Appendix 2.A.

The simplified dynamics of the Hexahopper facilitated rapid development of a preliminary hierarchical controller. The initial task of static walking verified the controller structure and the abstracted leg coordination scheme. Next, dynamic running based on the three-part controller used in previous Leg Lab robots was added to the suite of possible gaits. The results demonstrated that dynamic locomotion was possible in a six-legged, cockroach-sized creature. Finally, the running controller structure was modified to generate a slow (1.9 cm/sec) groucho running gait.

The Hexahopper's controller has the same global structure as the Hexabug, but uses simpler methods of force distribution and leg control.

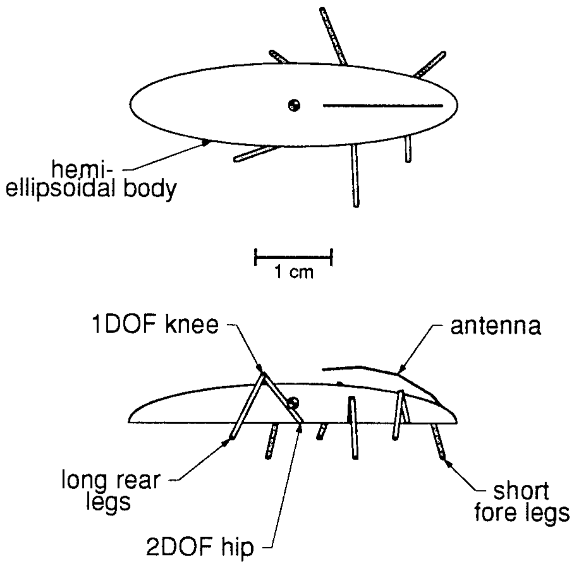


Figure 2–2: Hexabug schematic.

2.2.2 Hexabug Simulation

The Hexabug incorporates more complex dynamics than the Hexahopper, with anatomical features approaching that of a cockroach. Figure 2–2 depicts the creature model used in the Hexabug simulations. The Hexabug has 2-part, articulated legs and a hemiellipsoidal body, with body mass properties and leg attachment points taken from *B. discoidalis* anatomical data. Simulation parameters for the Hexabug model are in Appendix 2.B.

The Hexahopper and Hexabug were modeled as collections of rigid bodies with appropriate masses and moments of inertia, connected by joints. The creatures are free to move relative to the ground. Revolute joints have pure torque sources that can provide arbitrarily large specified torques or attempt to achieve a desired position using a proportional-derivative servomechanism. The telescoping joints in the Hexahopper's legs are controlled in the same way, but they have force, rather than torque, sources. Foot contacts with the ground are modeled as a spring-damper system that opposes penetration into the ground and prevents sliding on the surface. The simulations deactivate this spring-damper system if it would pull the foot into the ground. We produced the Hexahopper and Hexabug simulations using the Creature Library, a tool for creating dynamic computer simulations (Ringrose 1992). The simulations consist of equations of motion, ground contact models, a numerical integrator, a 3D computer graphics program, and user interface.

2.2.3 Groucho Running

Running is characterized by bouncing on legs as if they were springs, much like a pogo stick or bouncing ball. Cavagna described running this way some time ago (Cavagna, Thys, &

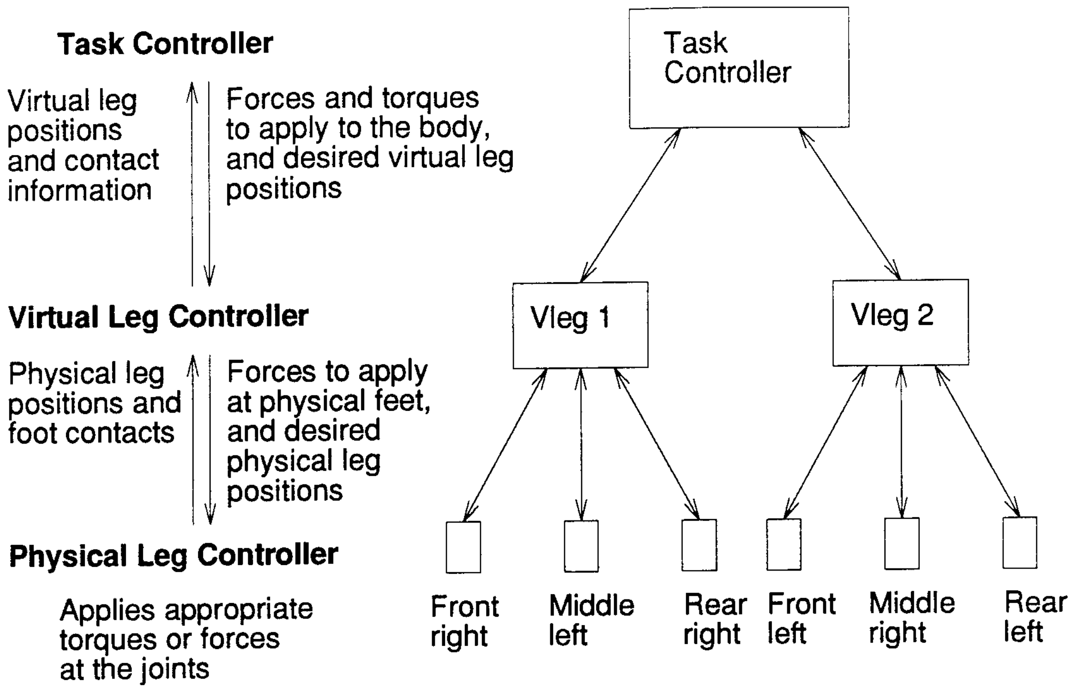

Task Controller
Virtual leg positions and contact information
Forces and torques to apply to the body, and desired virtual leg positions
Virtual Leg Controller
Physical leg positions and foot contacts
Forces to apply at physical feet, and desired physical leg positions
Physical Leg Controller
Applies appropriate torques or forces at the joints
Task Controller
Vleg 1
Vleg 2
Front right
Middle left
Rear right
Front left
Middle right
Rear left

Figure 2–3: The three level hierarchical controller common to both the Hexahopper and Hexabug simulations.

Zamboni 1976). Full and Tu observed this type of bouncing behavior in their cockroaches, despite the fact that there is no flight phase (Full & Tu 1990). In normal running a human can decrease the flight phase by flexing his or her legs to increase leg compliance. Running without a flight phase is a gait that McMahon calls *groucho running*, named after the actor who could be seen moving this way (McMahon, Valiant, & Frederick 1987). McMahon has shown that alternative methods of running exist that reduce or even eliminate the aerial phase. Examples include running in enhanced gravity, running in circles under centripetal acceleration, and running on a compliant surface (McMahon 1985).

2.2.4 Controller

We divided the Hexahopper and Hexabug controllers into a three-level hierarchy: the *physical leg level* (lowest), the *virtual leg level,* and the *task control level* (highest). The controller at the physical leg level uses the Jacobian relationship between the hip and foot to convert a desired external force vector at the foot into a set of leg joint torques. The controller at the virtual leg level coordinates the actions of three physical forming a tripod to emulate a single leg. The controller at the task level uses a variant of the three-part control used previously in the Leg Laboratory to provide the overall behavior of the system.

Physical Leg Controller

Each physical leg applies a desired force at the foot as determined by the Jacobian tensor relationship

$$\vec{\tau} = \mathbf{J}^T \vec{F}$$

relating desired force to required joint torques. Here

$\vec{\tau}$	is the vector of joint torques,
$\mathbf{J}$	is the 2-link leg manipulator Jacobian, and
$\vec{F}$	is the desired force vector.

The Hexahopper's legs are kinematically straightforward. However, the Hexabug legs are 2-link serial manipulators with a 2 degree-of-freedom universal hip joint followed by a distal revolute knee joint in a $\hat{z}$-$\hat{x}$-$\hat{x}$ rotation configuration. To simplify control, we treat the links as massless, rigid links and ignore inertial effects. Leg joint servos are implemented as ideal force or torque sources. As such, we can apply arbitrary forces at the foot of any physical leg. This has the advantage of abstracting the actual leg structure from the next layer of the controller.

Desired forces are either of two types: pure force or positioning. A positioning force drives the foot of the leg towards a desired Cartesian position. A sequence of positions defines a trajectory. We used a proportional-derivative control law to describe the desired positioning force:

$$\vec{F}_p = K_p(\vec{x} - \vec{x}_d) + K_d\dot{\vec{x}}$$

where

$\vec{F}_p$	is the positioning force vector,
$\vec{x}$	is the Cartesian virtual foot position vector,
$\vec{x}_d$	is the Cartesian desired position vector,
$\dot{\vec{x}}$	is the Cartesian velocity vector, and
K_p,K_d	are proportional and derivative gains respectively.

Hogan refers to this method of tracking a desired position in Cartesian space as opposed to joint space as one implementation of impedance control (Hogan 1985).

Virtual Leg Controller

The virtual leg control layer determines the contributions of each of three physical legs necessary to imitate a single telescoping leg with spherical hip joint. The virtual leg controller position controls elevated physical legs and force controls physical legs which are firmly planted on the ground. For simplicity, the virtual leg controller ignores the negligible inertial effects of the physical legs, letting feedback compensate for the resulting small errors.

Sutherland first introduced the *virtual leg* concept in his one-ton hexapod where passive hydraulic circuits equalized the force distribution in pairs of legs (Sutherland & Ullner 1984). Thus, two legs acted as one. The Leg Lab's quadruped made use of this same construct to control of three quadruped gaits (Raibert, Chepponis, & Brown 1986). These simple gaits, pacing, bounding, and trotting, utilized pairs of legs to generate motion. The quadruped was controlled by assigning pairs of legs to a functionally equivalent virtual leg such that the overall gait was transformed to an equivalent virtual biped gait.

Position Control Legs not in contact with the ground are position controlled. In this situation, a spatial kinematic transformation relates the actual physical foot positions to the position of the virtual foot. This transformation consists of defining a polygon with the actual feet at the vertices and the virtual foot at some location with respect to the polygon. For the Hexahopper and Hexabug, a time-invariant equilateral triangle defines this polygon spanned by a fore, middle, and hind foot, corresponding to foot positions 1–3 as shown in Figure 2–4. The virtual foot is in the geometric center of the support triangle.

The vector $\vec{u}_i$ from the virtual foot to the foot of physical leg i closes the kinematic chain for leg i and determines the desired position of the physical foot. The desired physical foot position is:

$$\vec{x}_{d_i} = \vec{x}_{d_v} + \vec{u}_i$$

where

$\vec{x}_{d_i}$	is the desired foot position of leg i,
$\vec{x}_{d_v}$	is the desired foot position of the virtual leg, and
$\vec{u}_i$	is the vector from the desired virtual foot position to corner i of the tripod base triangle.

For a given desired virtual foot position, the virtual leg controller calculates the three desired physical foot positions and passes them to the physical leg controllers. We fixed the width of the Hexabug's base triangle to 2 cm. The Hexahopper's base width was somewhat smaller. Conceptually, the shape of the base polygon can be variable as well, but we have not implemented this capability.

Force Control When planted in the ground, a leg is force controlled. In this situation, a single virtual leg receives from the task controller a desired resultant force and torque to be applied to the body. However, a virtual leg consists of three physical legs. Distributing the forces between the physical legs to create the desired resultant forces and torques is an underconstrained problem. We need to introduce constraints to the force distribution which allow us to choose and calculate one of the possible solutions.

The Hexahopper's simple geometry simplifies the coupling between applied and resultant forces and torques to the following 'triangular' form:

$$\begin{aligned} \vec{F}_d &= -\sum_{i=1}^{3}\left(\frac{\tau_i\hat{\theta}_i \times \vec{r}_i}{\|\vec{r}_i\|^2} + F_{i_l}\hat{r}_i\right) \\ \vec{\tau}_d &= -\sum_{i=1}^{3}\tau_i\hat{\theta}_i, \end{aligned}$$

where $\vec{F}_d$ and $\vec{\tau}_d$ are desired force and torque vectors applied to the body, F_{i_l} and τ_i are scalar axial leg forces and hip torques for leg i, $\hat{\theta}$ is the axis of rotation, and $\vec{r}_i$ is leg i's position vector.

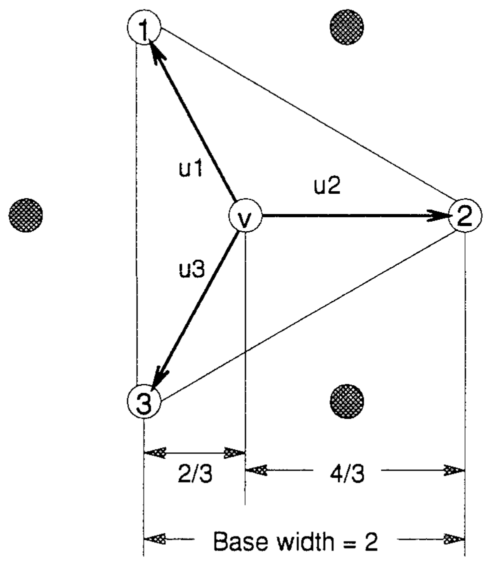


Figure 2–4: Diagram of the Hexabug tripod base triangle illustrating the spatial relationship between the virtual foot position and the position of the physical feet. The numbered circles correspond to the locations of the three feet comprising the tripod of the virtual leg centered at v. $\vec{u}_i$ are the displacement vectors from the virtual foot position, v, to the physical foot positions, 1–3. The shaded circles correspond to the physical foot positions of the other virtual leg during a typical stance.

The Hexahopper's ball joint hips allowed decoupling of $\hat{x}$ and $\hat{y}$ torques. Thus, the $\hat{x}$ and $\hat{y}$ torques could be determined separately. For example, in the x direction, we chose

$$\tau_{i_x} = -\alpha \tau_{d_x},$$

where α is a weighting term equal to

$$\alpha = x_i / \sum_{i=1}^{3} x_i,$$

and

$$x_i = \begin{cases} r_{i_x} & \text{if } r_{i_x}/\tau_{d_x} < 0 \\ 0 & \text{otherwise.} \end{cases}$$

Above, τ_{d_x} and r_{i_x} are the x components of $\vec{\tau}_d$ and $\vec{r}_i$, respectively. The hip torques in the y direction are determined similarly. In the z direction,

$$\tau_{i_z} = -\tau_{d_z}/3.$$

If $\tau_i \hat{\theta}_i \times \vec{r}_i \approx 0$, that is, if the hip torques or projections of r_i on the ground are negligible, then the axial leg forces can be approximated by

$$[F_{1_l} F_{2_l} F_{3_l}]^T = -[\vec{r}_1 \vec{r}_2 \vec{r}_3]^{-1} \vec{F}_d.$$

For the Hexahopper, the legs do not sweep very far and the projections of r_i on the ground remain small over a wide range of speeds.

For the more complex Hexabug geometry, we use the method documented by Song and Waldron for the Adaptive Suspension Vehicle to distribute the forces (Song & Waldron 1989). The results, simplified to three legs, are reproduced in Appendix 2.D. Song and Waldron's method introduces the constraint that no two feet can work against each other. This constraint is inconsistent with the force patterns observed in actual cockroaches. Full has documented the fact that frequently the front legs will push back while the rear legs are pushing forward. More appropriate, but harder to calculate, constraints exist, such as minimizing the torques applied at the joints. We plan on using a different force distribution method at a later date.

Spring Simulation and Thrusting A virtual leg contains a Hooke's law spring that acts along the length of the virtual leg. The spring deflection is calculated as the difference between the leg length at touchdown and the current length. The spring acts as a passive virtual element whose rest length moves instantaneously during the transition from COMPRESSION to THRUST states. This action changes the energy state of the spring, passively introducing energy as necessary to compensate for system losses experienced over the previous bounce cycle.

Task Controller

The task controller uses a cyclic state machine to coordinate the two virtual legs. Similar to the three-part controller used for hopping in previous Leg Lab robots, it specifies when each of the three primary control actions are applied by the virtual legs during the locomotion cycle. These control actions are regulation of hopping height, speed, and posture. Table 2–1 outlines the state transition sequence used in task level control.

Groucho running differs from regular bipedal running in that at least one leg is in contact with the ground at all times. Normally, this would be called walking. The double support phase absent in running exists in groucho running, which requires the coordination of legs when support is transferred from one foot to another. The groucho-running controller extends the 3-part controller concept developed for running to accommodate the double support phase.

Hopping Height Humans and kangaroos store energy in their leg springs when they run and hop, as do the hopping machines we have built. In all of these cases, hopping is a resonant oscillation of the body mass on springy legs. This oscillation would normally decays, but thrust is provided once per cycle to force the oscillation. In the robots, the amount of thrust is fixed and the magnitude of the hopping oscillation converges to a steady state altitude. In the control systems we built for the Hexahopper and Hexabug, the thrust is actively controlled. The hopping height is actively controlled by adjusting the energy state of the leg spring. Moving the rest position of the leg spring by an amount ϵ, changes the energy state of the spring as determined by the following expression:

$$\epsilon = \frac{\Delta h}{|\Delta h|}\sqrt{\frac{2}{K}mg|\Delta h|} - \delta.$$

The addition of thrust to the virtual spring force in the THRUST through UNLOADING states adequately controls the hopping height. The derivation of the above equation is in Appendix 2.F.

Body Attitude Maintenance of body attitude is crucial to good robot hopping performance. For hopping, regulation of body attitude occurs once per bounce cycle within stance, while the body provides a sufficient normal force to the ground to prevent the foot from slipping in response to a torque applied about the hip. In groucho running, at least one virtual leg remains in contact with the ground at all times, so for the Hexahopper and Hexabug the virtual leg with the largest load regulates body attitude. A virtual leg applies a corrective torque, τ_i, in response to an angular error about axis i according to the control law:

$$\tau_i = -K_p(\phi_i - \phi_{d_i}) - K_d\dot{\phi}_i$$

where

τ_i	is the virtual hip torque about axis i,
ϕ_i	is the current body angle about axis i,
ϕ_{d_i}	is the desired body angle about axis i,

State	Trigger Event	Actions
DECOMPRESSION$_A$ (double support)	Leg B touches down	Control attitude with leg A Zero hip torques on leg B Simulate virtual spring in leg B Apply thrust force in leg A
UNLOADING$_A$-LOADING$_B$ (double support)	Leg A transfers support to leg B	Control attitude with leg B Zero hip torques on leg A Simulate virtual spring in leg B Apply thrust force in leg A
COMPRESSION$_B$ (support on leg B)	Leg A leaves ground	Control attitude with leg B Leg A mirrors leg B Shorten leg A Raise leg A Adjust the base size of leg A Simulate virtual spring in leg B
THRUST$_B$ (support on leg B)	Leg B reaches maximum compression	Control attitude with leg B Position leg A for touchdown Shorten leg A Leg A mirrors leg B Apply thrust force in leg B
EXTENSION$_A$ (support on leg B)	Leg B reaches negative support	Control attitude with leg B Position leg A for touchdown Extend and lower leg A Apply thrust force in leg B

Table 2-1: Finite state machine for groucho running. The state shown in the left column is entered when the event listed in the middle column occurs. The states advance in the listed order continuing on to similar states for the other virtual leg, beginning with DECOMPRESSION$_B$ and ending with EXTENSION$_B$.

$\dot{\phi}_i$	is the current body angular velocity about axis i,
K_p,K_d	are proportional and derivative gains respectively, and
$i = \{1,2,3\}$	corresponds to roll, pitch, and yaw respectively.

In addition to regulating posture during stance with the above control law designed for single-legged hopping, we used mirroring to minimize attitude disturbances. The idle leg mirrors that of the stance leg, decreasing the net change in angular momentum, and thus the net torque about the body center of mass. Because the Hexahopper and Hexabug do not experience a ballistic phase and the leg inertias are small, the value of mirroring is not as significant as in running. However, we expect that when we increase the leg inertias to values consistent with a real cockroach mirroring will produce greater benefit.

Forward Running Speed For a running biped, accurate foot placement governs the forward running speed. Foot placement occurs during flight just prior to touchdown. If you treat the leg as a spring, there is a *neutral point* such that if the leg touches down on the neutral point the leg will be at greatest compression when it is vertical. By symmetry, assuming a lossless spring, the biped will take off with the same speed as it landed. Additionally, if the foot falls short of the neutral point the biped will gain speed at the expense of hopping height, and if the foot overshoots the neutral point the biped will gain height at the expense of forward speed.

The groucho-running Hexahopper and Hexabug use the neutral point principle to control their speeds, although they calculate foot positions for two virtual legs rather than six individual legs. The task controller calculates desired foot positions with a simple feedback loop:

$$x_{d_{xy}} = \dot{x}T_c + K_{\dot{x}}(\dot{x} - \dot{x_d})$$

where

$x_{d_{xy}}$	is the planar displacement of the foot
	from the projection of the center of mass,
T_c	is the time from touchdown to maximum spring compression,
$\dot{x}$	is the current average forward speed,
$\dot{x_d}$	is the desired forward speed, and
$K_{\dot{x}}$	is a speed gain.

Comparison of the above equation with former Leg Lab foot placement controllers reveals a subtle difference in the calculation of the position of the neutral point $\dot{x}T_c$. Previously, the neutral point was determined to be $\frac{\dot{x}T_s}{2}$, where T_s is the estimated duration of next stance phase. For a symmetric stance phase, as in running, $T_c = \frac{T_s}{2}$. However, the stance phases for the Hexahopper and Hexabug are asymmetric, with T_c less than the time from maximum compression to liftoff.

The timing of touchdown is of greater importance to groucho running than running. When running, the support leg begins storing elastic potential energy at touchdown. When groucho running, in order for a leg in to begin storing energy at touchdown, the leg must not contact the ground before top of stance, the point where the body is at maximum gravitational potential. At the point that the body altitude z is at maximum, its time derivative $\dot{z}$ is zero. Since the profiles of both z and $\dot{z}$ are smooth, once scaled, $\dot{z}$ is suitable for use as a trajectory for smoothly leading the foot to the ground at top of stance. The trajectory, specified during EXTENSION as the complement to $x_{d_{xy}}$ above, that fulfills this task is

$$x_{d_z} = K\dot{z}$$

x_{d_z}	is the desired vertical foot position in the inertial frame,
$\dot{z}$	is the vertical speed, and
K	is a gain.

2.2.5 Results

We performed simulations and recorded data at various speeds. After setting a desired speed, we allowed the simulation to run until the transients had died out before we recorded data. It generally took less than ten step cycles before the variations in data values were small between successive steps.

Figure 2–5 contains data for the Hexabug simulation at steady speeds of approximately 4, 25, and 52 cm/sec. At all speeds, both virtual legs are on the ground when transferring support from one foot to the other. There is no flight phase. At higher speeds, however, some feet briefly lose ground contact just prior to double support. This can be attributed to body reactions created by motions in the other virtual leg just prior to touchdown. Fluctuations in horizontal velocity and height correspond to running behavior. Therefore the Hexabug is groucho running.

Over all speeds, footfalls occur just after maximum height is attained. The force profiles are nearly symmetric, with a small peak due to height control thrusting. At higher speeds, the Hexabug's bounce frequency increased, while the bounce amplitude and average height diminished. The amplitude of speed oscillations remained fairly constant regardless of speed. The virtual leg length profiles suggest that much longer physical legs were required at higher speeds. The leg length at toe-off is greater than at touchdown, with the difference increasing dramatically as speed increases.

As speed increases, the Hexabug has decreasing control of its body height. Leg excursion increases, resulting in more coupling between vertical and horizontal motion as the leg sweeps. The amount of thrust added to control body height is determined assuming it will be essentially applied vertically, yet the thrust vector moves during the support phase. As a result, at higher speeds the thrust which is supposed to control body height gains an increasing horizontal component.

Figure 2–6 illustrates the response of the Hexabug to a step in desired velocity. The Hexabug controller maintains a stable bounce height independent of the changing velocity. The velocity profile reaches a steady state value after approximately eight bounce cycles. At higher velocities, the settling time is lower. Even though the position-control law as implemented does not include an anticipatory, error-derivative term, the tracking error is small. The Hexabug controller produces the desired groucho-running behavior over a wide spectrum of running speeds.

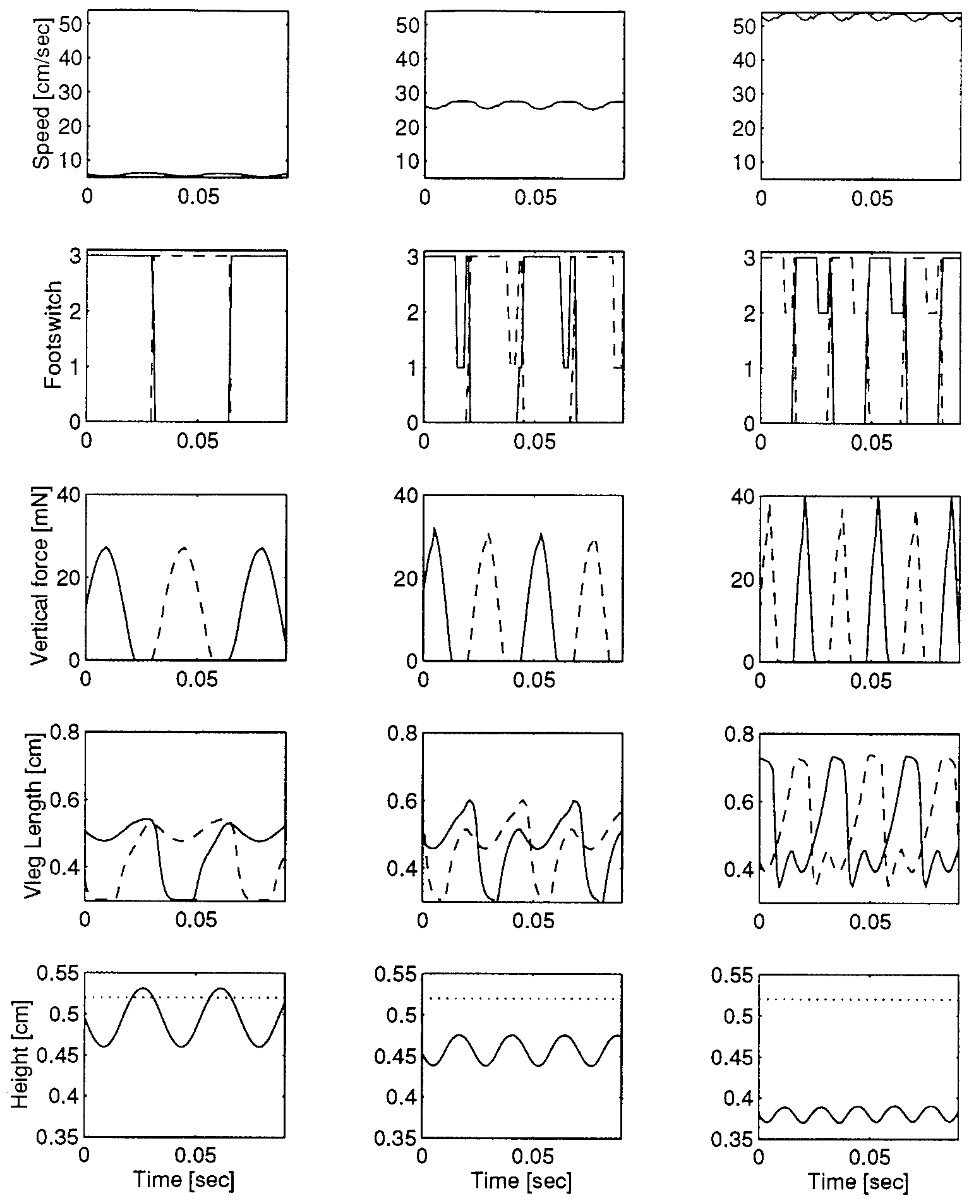


Figure 2–5: Data from the Hexabug at speeds of 4, 25, and 52 cm/sec. Solid lines indicate data for Virtual Leg 1. Dashed lines indicate data for Virtual Leg 2. From top to bottom: horizontal velocity, the number of feet in contact with the ground for each virtual leg, vertical force, virtual leg lengths, and body height (the dotted line is the desired height). The oscillations in speed are characteristic of a running creature. The force and leg length profiles are symmetric between the two virtual legs. At higher speeds, the legs extend much more than at lower speeds.

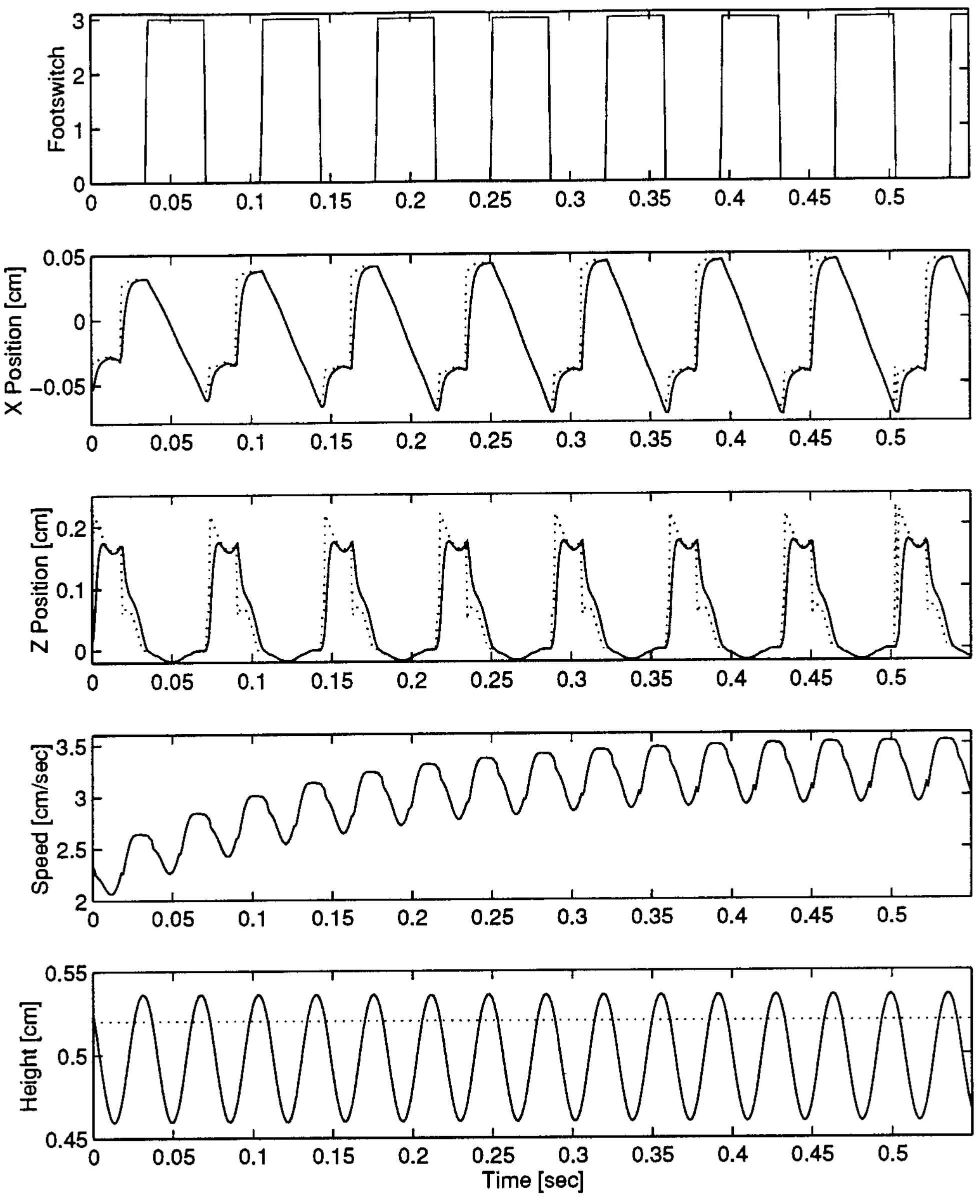


Figure 2-6: Hexabug response to a step in desired velocity. From top to bottom: times when virtual leg 1 is on the ground, two graphs with the trajectory following capability of virtual leg 1 (the dotted line represents the desired trajectory), convergence of the horizontal velocity, and body height. The lower peak in the desired z position indicates the switch to tracking the time rate of change of z position.

2.2.6 Limitations

The Hexabug simulation, although closer to a cockroach than the Hexahopper, has a few limitations on its applicability. First, it does not correspond directly with a cockroach. It is of approximately the same size, but the leg mass properties are different. We plan to change the mass properties and configurations of the legs to correspond to a cockroach in the future. Second, the Hexabug does not have any passive energy storage elements. There is a virtual "spring", but it is implemented by torques at the joints which are not acting as springs. The dependence on active elements would limit the efficiency of an actual machine using the Hexabug's controller. Third, the Hexabug's force distribution scheme prevents legs from acting against each other. In contrast, a cockroach's legs do interact. In addition, allowance of force interaction between legs can result in lower overall required joint torques or power consumption. Finally, as in any simulation, there may be effects which are unanticipated, and therefore not simulated. For example, aerodynamics and surface tension may be important for a creature the size of a Hexabug.

Experiments indicate that these limitations are unlikely to cause problems for a while. Initial experiments involving attaching cockroach-like legs to the Hexabug have had positive results. The Hexabug's lack of explicit spring storage won't become a problem until we limit the amount of energy it uses. Interacting forces on the feet could be achieved by applying additional forces at the feet, calculated to cancel each other and have no net affect on the Hexabug. Finally, unanticipated effects can only be seen when an actual creature exists. However, if we continue our work on the Hexabug and make it more like a cockroach, we can examine the differences in their behavior to find possible important unsimulated forces.

2.2.7 Conclusion

We developed inherently dynamic, groucho-running controllers that made a pair of cockroach-like, simulated creatures run, one of them at 52 cm/sec. The hierarchical controller based in part on the Leg Lab's successful, three-part controller demonstrated that Leg Lab algorithms could be extended to produce a groucho running gait. The virtual leg construct resulted in simultaneous touchdown and toe-off over all speeds as well as precise force control while reducing the control of six legs to the task of coordinating the behavior of two virtual legs. The Hexabug simulation models a creature much smaller than those previously modeled or built in the Leg Lab and may potentially be the first artificial, groucho running creature in existence. The simulations provided information that would be potentially useful in developing six-legged, dynamic robots.

2.3 Playback Cockroach Simulation

The Playback Cockroach is a physics-based simulation modeled off a cockroach (*B. discoidalis*). The simulation has sizes and mass properties measured from an actual cockroach. The controller plays back data from a digitized *B. discoidalis* running cycle as desired leg positions, and proportional-derivative servomechanisms at joints attempt to track the desired leg positions. The result is a groucho-running gait approximating that of a cockroach.

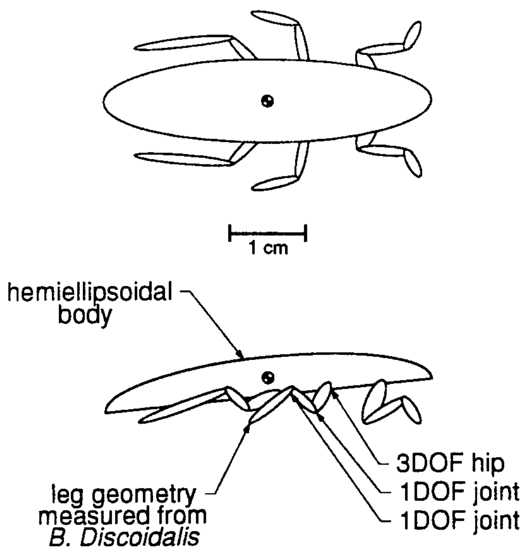


Figure 2–7: Playback Cockroach schematic.

2.3.1 Simulation

We created the Playback Cockroach creature model using physiological specifications which Full supplied (Appendix 2.C). The Playback Cockroach's legs are generated from digitized leg scans of *B. discoidalis.* We represent the body in simulation as a hemiellipsoidal body with axes corresponding to the measured dimensions of the cockroach. We use the measured cockroach body mass, but since the body inertia properties have not been measured they are taken from the hemiellipsoid, assuming uniform density. The leg attachment positions correspond to measurements from a cockroach. For each leg, a gimbal joint attaches the body to the coxa. Pin joints connect the remaining leg segments.

We eliminated the tarsus from the simulation for two reasons. Visual inspection of cockroaches on videotape seems to indicate that the tarsus plays a small role in cockroach locomotion. Also, the tarsus has an extremely small mass relative to the rest of the body. This small mass at the far end of the leg makes the simulation numerically less stable. We plan to reintroduce the tarsus at a later date to verify this decision.

We generated the Playback Cockroach simulation using the Creature Library, as described in section 2.2.2. Ground contacts are located on the end of the tibia and other locations which tend to touch the ground, such as the posterior end of the creature's abdomen and the femur-tibia joints of the front legs. The cockroach simulation has 5 degrees of freedom in each leg and 6 more to give the body free motion in three-dimensional space.

In our simulations, the links connecting the joints are absolutely rigid. However, there is some possibility that in an actual cockroach the exoskeleton connecting the joints flexes slightly during running. We have assumed that such flexion of the links in a real cockroach is small enough to be included with the joint motion of the simulation.

2.3.2 Playback Cockroach Controller

Full provided us with Cartesian joint and foot positions for a complete cockroach stride digitized from slow motion videotape. We chose a segment of the motion data which would loop well and contains one complete cycle of an alternating tripod gait. We smoothed the motion data to eliminate discontinuities at the loop point.

Using the physiological, cockroach-leg kinematics, we calculated a sequence of leg joint angles which corresponded to the motion data. Proportional-derivative servomechanisms at the joints track the replayed joint position trajectories to produce the simulated cockroach's motion. Looping the data produces a continuing running gait. Changing the rate of motion data playback makes the simulation run at different speeds.

The motion data represents data taken from a cockroach running at 23.7 cm/sec. At normal playback speed, the simulated cockroach runs at 17.8 cm/sec, a bit slower than the original cockroach. The stride is 0.096 seconds long.

Tuning

The proportional-derivative servomechanisms have spring and damping coefficients. We determined the coefficients based on a selected frequency (identical for each joint) and the inertias the joint would see if the rest of the cockroach were rigidly locked in a configuration corresponding to part way through a stride. Let I_1 and I_2 be the inertias on each side of the joint and let ω be the frequency. The spring coefficient (which is multiplied by the position error) is $k = \omega^2 I_1 I_2/(I_1 + I_2)$ and the damping coefficient is $b = 1.4\omega I_1 I_2/(I_1 + I_2)$. This formula corresponds to critically damped oscillation between two bodies, both of which are free to move. We experimented with different frequencies and dampings when the leg is in the air and on the ground, but did not find any worthwhile improvements.

Improving trajectories

Ideally, we want our simulated cockroach's joints to follow the trajectories traced out by the original cockroach. However, proportional-derivative servomechanisms do not perfectly follow a given trajectory. We can improve trajectory following by increasing the gains in the servomechanisms. Unfortunately, a high gain proportional-derivative servomechanism will accurately trace any errors that may exist in the recorded data, and will also result in a simulation which is extremely stiff.

We used a simple learning algorithm to improve the simulated cockroach's trajectories. We started with a target trajectory which was the same as the recorded trajectory. We then ran the simulation and used a fraction of the error between current position and recorded trajectory to update the target trajectory. This type of learning has the same effect as learning a feed forward torque at the joints.

This type of learning is only partially effective. It does not address the problem of errors in the recorded data. More importantly this learning method has problems dealing with feet striking the ground. When a foot hits the ground and takes the weight of the cockroach, the torques used in that leg increase dramatically. The learning algorithm learns this increase in torques. If, on a later cycle, that leg touches down a little later, the increased torques will slam the foot into the ground. This action can result in body attitude oscillations which alter the timing of other footfalls, leading to eventual instability.

2.3.3 Results

We performed simulations and recorded data over a range of playback speeds, up to three times the recorded frequency. After setting a playback speed, we allowed the simulation to run until the transients had died out before we recorded data. It generally took less than 1.75 seconds for the transients to settle, after which we recorded 0.25 seconds of data.

Figure 2–8 contains data from the playback cockroach running at three speeds. It ran successfully at all tested speeds, although at higher speeds the springs in the proportional-derivative servomechanisms were unable to track the trajectories effectively, and as a result the performance began to degrade.

As trajectory frequency increases there is a qualitative change in the oscillation associated with the cockroach's locomotion. We believe the change happens because the interaction between servo spring-like forces and the cockroach body inertias results in different low and high frequency vibrational modes. As the step frequency increases, the oscillatory behavior of the cockroach becomes increasingly dominated by the high frequency vibrational mode. We are still investigating this mode change. Figure 2–9 graphically demonstrates the nature of the mode change.

The most striking feature of the playback cockroach performance is that over all playback speeds, the cockroach runs in a stable manner. This result was not at all obvious before we attempted the simulation and varied playback speed. The upper limit to speed is mainly limited by the leg's ability to track the desired trajectories. Longer legs should allow it to run faster, since the resulting longer strides would permit a lower step frequency for any given speed. For speeds below approximately 44 cm/sec, the simulation produces acceptable dynamic locomotion.

2.3.4 Limitations

The Playback Cockroach has a few limitations, some of which it shares with the Hexabug. It, too, has some physical departures from an actual cockroach. For example, the body is not shaped right, and it is missing the tarsus on each leg. The ground contact model we use is appropriate for a foot pad, like most mammals, but not necessarily for a cockroach's foot. Modeling the possible flexion in the chitinous legs by additional joint motion may not be sufficient, or accurate enough. The controller is not very flexible: we have a set of joint paths that work, but no information on how to modify them. Finally, the Playback Cockroach suffers from the same problems with unsimulated aspects of reality as the Hexabug.

2.3.5 Conclusion

The development of a six-legged, cockroach-like, physics-based simulation has been detailed. The Playback Cockroach utilizes playback data from one complete *B. discoidalis* running cycle as desired trajectories for the joints of a physically realistic, rigid body simulation with mass properties measured from an actual cockroach. By changing the rate of data

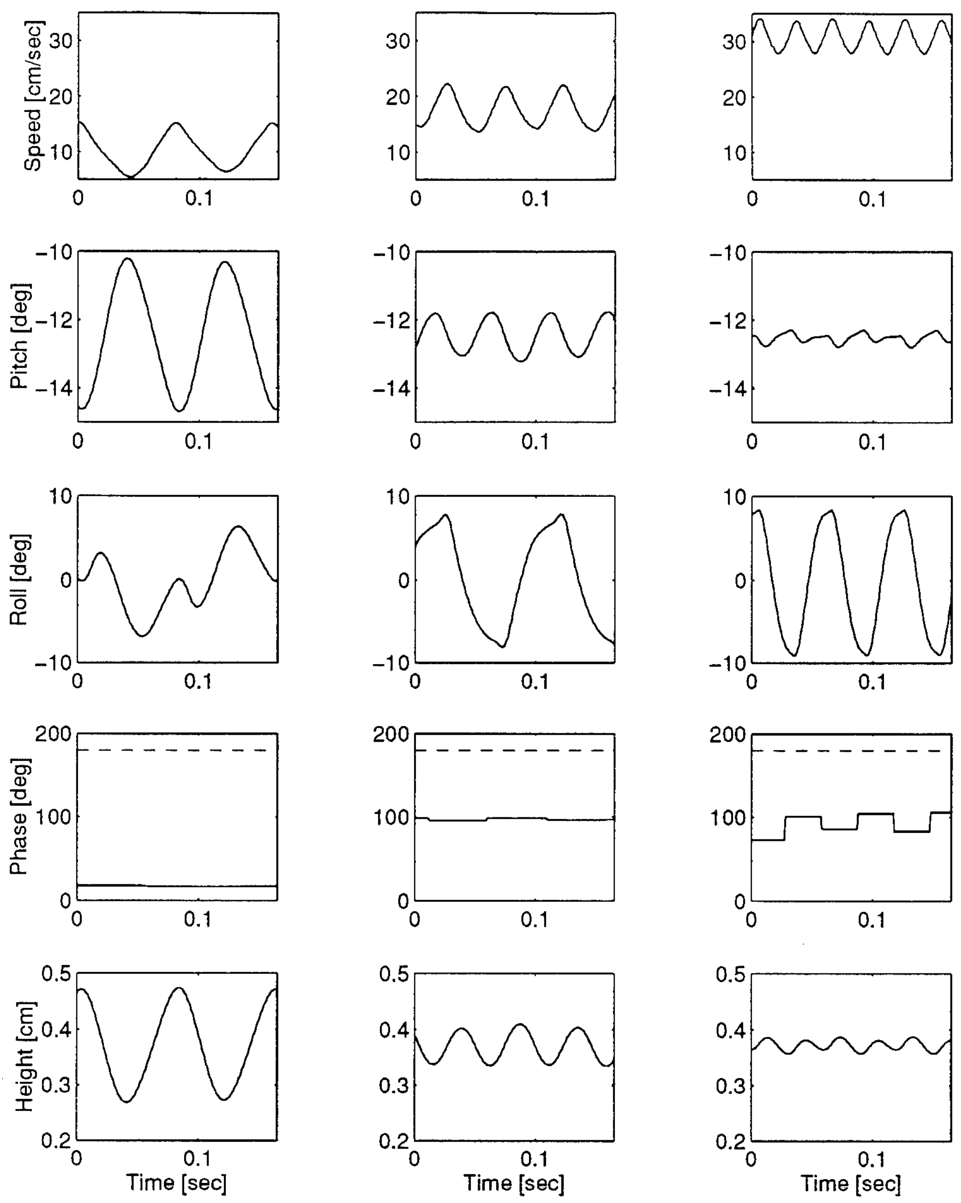


Figure 2-8: Data from the simulated cockroach at speeds of 10, 18, and 31 cm/sec (left to right). The middle set of graphs corresponds to playing back the desired trajectories at the same rate as the original recording. From top to bottom: horizontal velocity, phase shift between gravitational potential energy and horizontal kinetic energy, pitch, roll, and body height above the ground. In the phase graph, the dashed line indicates a phase shift of 180 degrees, corresponding to a walking gait.

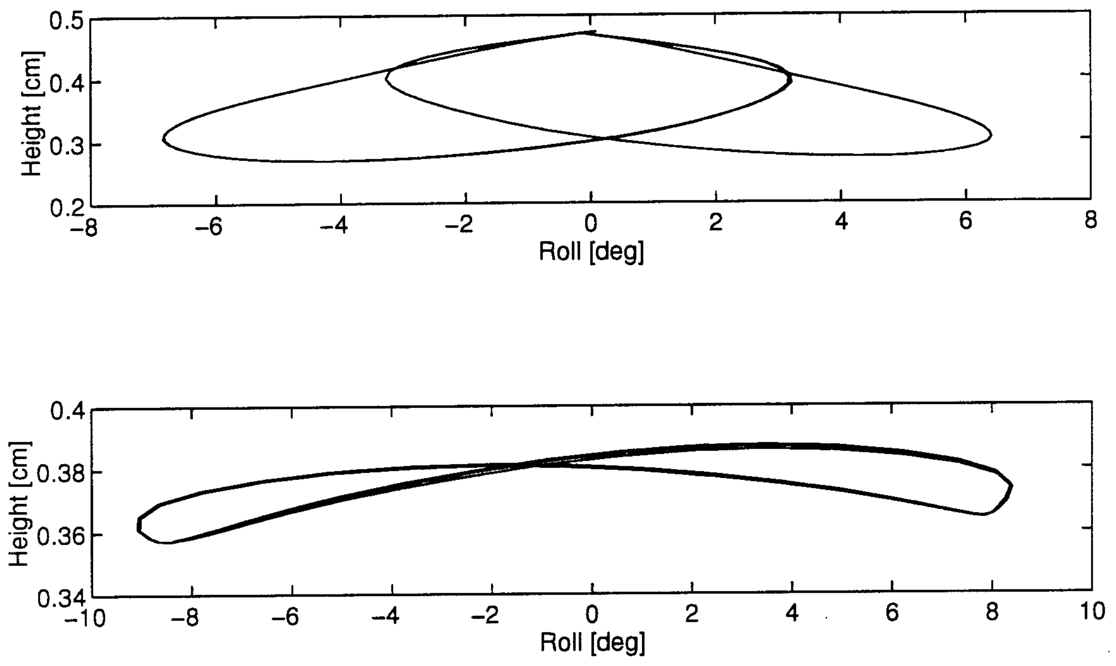


Figure 2–9: Phase plot of Playback Cockroach height and roll. The upper graph shows a typical low frequency phase relationship between body height and roll. For high frequencies, the Playback Cockroach experiences a mode change resulting in the phase relationship indicated in the lower graph. The low frequency data corresponds to the data in the left hand graphs of figure 2–8, while the high frequency data corresponds to the right hand graphs.

playback, the Playback Cockroach simulation results in stable running over a wide range of playback speeds up to 44 cm/sec. The upper speed limit is determined only by limitations in achievable leg configuration, indicating robustness with simple pattern-based control.

2.4 Is Cockroach Locomotion Dynamic?

We are investigating whether a dynamic treatment is necessary for a running cockroach. In order to ask this question, we must first define "dynamic" within the context of legged locomotion. One possible definition involves evaluating the static stability of the creature and observing that if it is not statically stable it will either fall or recover dynamically. Another definition of dynamic locomotion depends on the importance of the creature's kinetic energy. Still another definition considers energy storage in elastic elements. We use a physically realistic simulation to evaluate cockroach locomotion according to all three criteria, comparing it with measured cockroach data where possible, and find that it is only dynamic according to some of them.

Cockroaches use an alternating tripod gait at most speeds. They have at least three legs on the ground, and the center of mass is directly above the polygon of support defined by those legs.[1] For a creature moving slowly, this would indicate static stability. However, there are portions of the running cycle where the cockroach has sufficient momentum to fall despite that fact that it is statically stable.

The second definition is more intuitive: something is dynamic if the fact that it moves quickly is important. A creature which is locomoting dynamically would fall if it suddenly lost its kinetic energy. On the other hand, static locomotion does not depend on kinetic or elastic energy. Instead, static locomotion requires that all the work comes immediately from the muscles. Therefore, if the muscles are capable of quickly restoring the creature's kinetic energy to its usual level, the kinetic energy is unimportant and the motion is not dynamic.

Finally, a more restrictive definition of dynamic locomotion involves the use of elastic energy storage. When running, animals use their legs as springs, like a pogo stick or a bouncing ball. In contrast, most walking animals use an inverted pendulum gait in which they fall from one leg into a double support phase, then use the momentum from the fall to rise up onto the next leg. Most people dealing with "dynamic locomotion" consider running, not walking, to be dynamic. Although cockroaches do not have a flight phase, they run as if they store energy.

2.4.1 Cockroach Simulation

We used a physically realistic simulation of *Blaberus discoidalis*, a South American cockroach. We describe the simulation elsewhere in this report, so only an overview is presented here. Full supplied physiological specifications which we used with the Creature Library to create a rigid-body simulation with appropriate masses and moments of inertia.

[1] Cockroaches have been observed to use fewer legs at extremely high speeds. We are not considering that extreme case.

Full also provided us with motion data for a complete cockroach stride digitized from slow motion videotape. From this data, we calculated the corresponding leg angles in the simulation over the same stride. To control the cockroach's motion, we replay the calculated leg angles as desired leg positions, looping the data to allow multiple strides. Proportional-derivative servomechanisms at the joints attempt to track the replayed leg trajectories

The stride is 0.096 seconds long. The original cockroach was in an alternating tripod gait, and the data contains two tripod steps. We carefully chose a section of the recorded data which would loop well and then smoothed it to eliminate the discontinuities at the loop point. The cockroach was running at 23.7 cm/sec when Full recorded the data. The motion data represents data taken from a cockroach running at 23.7 cm/sec. At normal playback speed, the simulated cockroach runs at 17.8 cm/sec, a bit slower than the original cockroach.

In the simulation, the connections between joints are absolutely rigid. In an actual cockroach the exoskeleton connecting the joints flexes slightly during running, possibly storing energy. We assume that such flexion of the links in a real cockroach is small enough to be included in the simulation's joint motion. Because the simulation's joint are essentially springs with controllable rest lengths, including exoskeletal flexion in the joint motion still allows energy storage.

2.4.2 Static Support

A classic definition of dynamic is “not static”. We evaluated the possibility that our simulated cockroach, running at speeds and frequencies comparable to a real cockroach, was statically stable.

We evaluated static stability in two ways. Because the simulated cockroach runs in an alternating tripod gait, we checked each tripod to see if the center of mass remained directly above the triangle designated by the ground contact points, also known as the polygon of support (Figure 2–10). Ignoring velocities, if the center of mass is within the polygon of support for the legs which are on the ground, then the cockroach is statically stable. We found that the center of mass was always above the polygon of support for a tripod in contact with the ground; that is, neglecting velocities, the cockroach remained statically stable.

Next, we included velocity. At each moment in time we calculated the energy, beyond its current kinetic energy, required for the center of mass to flip up and out of the triangle of support if the body were to become completely rigid (Figure 2–11). This calculation is an approximation—the simulated cockroach's legs would bend somewhat if it stopped suddenly, absorbing energy. We found that when the cockroach data was played back at the recorded data rate, there was a small period when the cockroach simulation had sufficient energy to flip out of the polygon of support. This time occurred when a new tripod contacted the ground, and was of negligible duration. Increasing the playback speed produced higher frequency motion from the simulated cockroach, but the gait became less stable according to this measure.

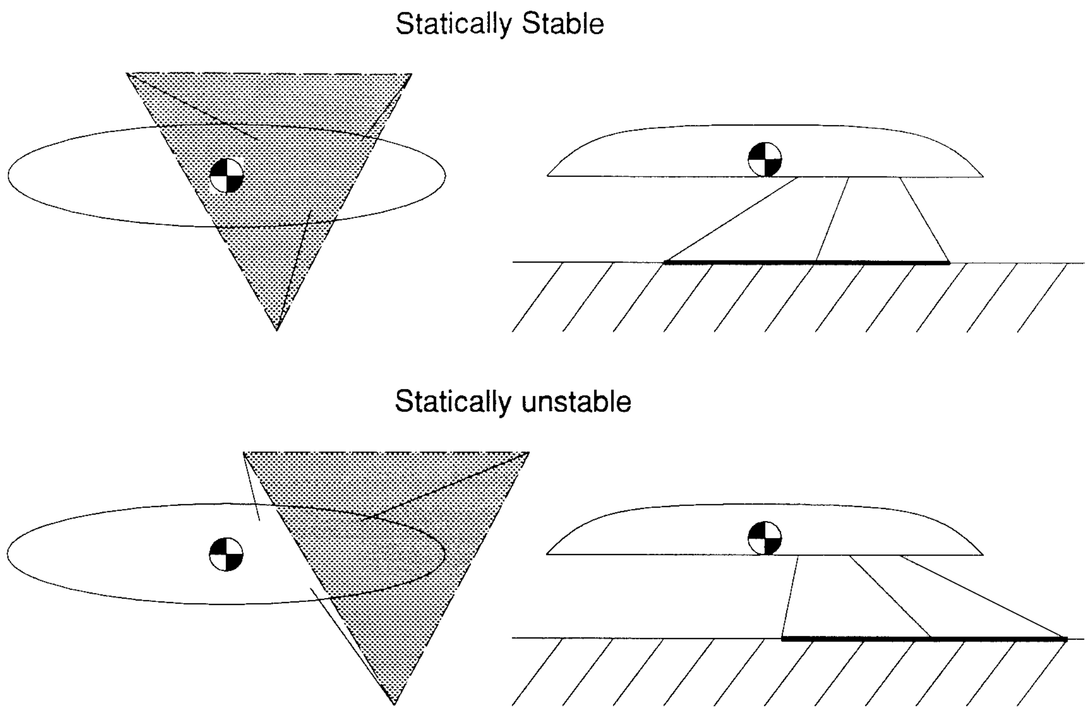


Figure 2–10: Illustration of a simplified cockroach in and out of its polygon of support. The top figure demonstrates statically stable support. The bottom figure demonstrates statically unstable support.

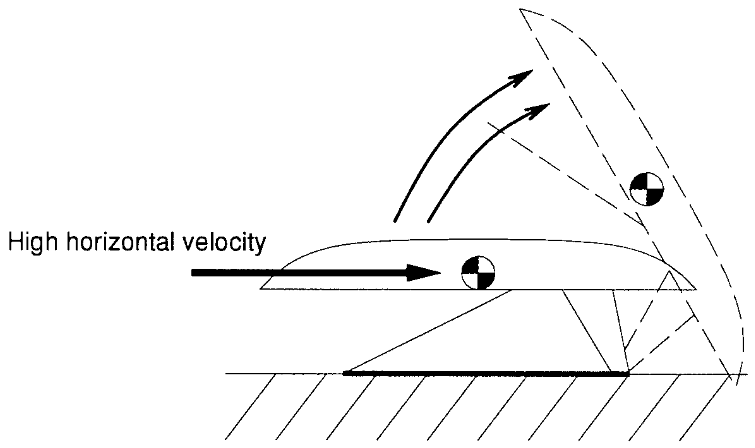


Figure 2–11: Momentum changes the stable positions of the center of mass. With enough momentum, the cockroach is able to tip over and may be unstable, even though it is statically stable.

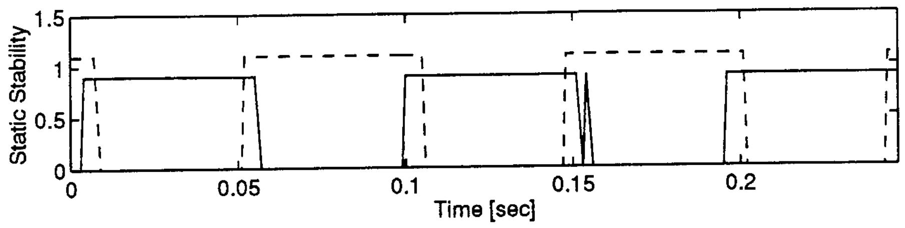


Figure 2–12: As the cockroach runs, its center of mass remains within the polygon of support for one leg or the other. The solid line is high when one tripod can provide static support for the simulated cockroach, while the dashed line is high when the other can provide static support.

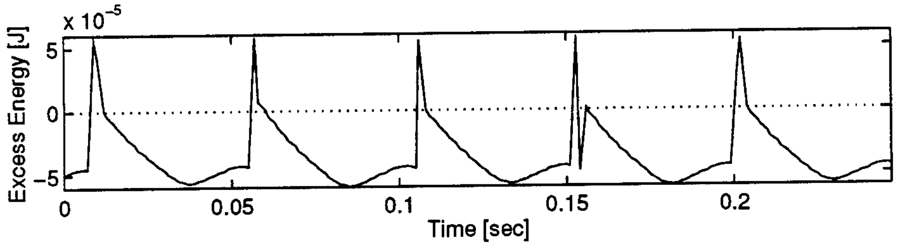


Figure 2–13: The amount of excess energy which needs to be removed in order for the cockroach to be stable despite its momentum. At the frequency at which the physical cockroach data was recorded the simulated cockroach is stable, as indicated by negative values for excess energy. The positive peaks occur when legs lift off the ground.

2.4.3 Kinetic Energy

Another definition of dynamic considers the significance of the creature's speed and associated kinetic energy. A dynamically moving object would be unable to proceed if it suddenly lost its kinetic energy. On the other hand, static locomotion requires that all the work comes immediately from the muscles. Therefore, if the muscles are capable of quickly restoring the creature's kinetic energy to its usual level, the kinetic energy is unimportant because it can be quickly replenished.

The "muscles" in our simulated cockroach are proportional-derivative servomechanisms at the joints. Currently, there is no limit to the amount of torque they produce. However, a real cockroach experiences biological and mechanical limitations to the amount of torque a joint can exert. We have yet to verify that the range of joint torques developed in simulation accurately reflect the mechanical capability of an actual cockroach. One way to do this will be to compare the forces generated by an actual cockroach leg with the forces generated by the simulation. By equating the force applied by the simulation to the force applied by a physical cockroach, we may be able to calculate a set of approximate maximum torques for each individual joint. Using these values for maximum torque at the joints will let us examine how easily the muscles can achieve the kinetic energy observed in the running simulation. Another means of evaluation would be to investigate the number and locations of muscle fibers around the cockroach joints. Future work might center on similar studies aimed at determining time constants for the joints to infuse energy into the system. Comparison of these time constants with typical stride periods will indicate if cockroach muscles can quickly restore cockroach kinetic energy to typical running levels.

2.4.4 Elastic energy

Many people studying legged locomotion use "dynamic" to mean the difference between running and walking. Running animals use their legs as springs, like a pogo stick or a bouncing ball. In contrast, most walking animals use an inverted pendulum gait in which they fall from one leg into a double support phase, relying on the momentum from the fall to rise up onto the next leg. The essential difference between running and walking is the use of elastic energy, not presence of a flight phase (Figure 2–14).

Running uses elastic energy storage, which is difficult to measure in an actual animal. You can approximate the amount of elastic energy storage if you assume locomotion is a sustained oscillation in which energy transfers largely between horizontal kinetic energy and gravitational potential energy and elastic energy storage. This approximation requires the assumption that other forms of energy storage, such as vertical kinetic energy, are negligible and that muscles actually perform very little work. Conceivably, since the contributions of muscle-induced and elastically-stored energy in external motions are indistinguishable, muscular energy could specify the system behavior. However, active, muscular-energy supply is inefficient compared to passive energy storage.

Both gravitational potential and horizontal kinetic energy are easily measured. Because the total energy is approximately constant under our assumptions, we can determine the use of elastic energy storage by noting the phase relationship between these energies. If they are in phase or of different magnitudes, then there must be substantial elastic energy storage

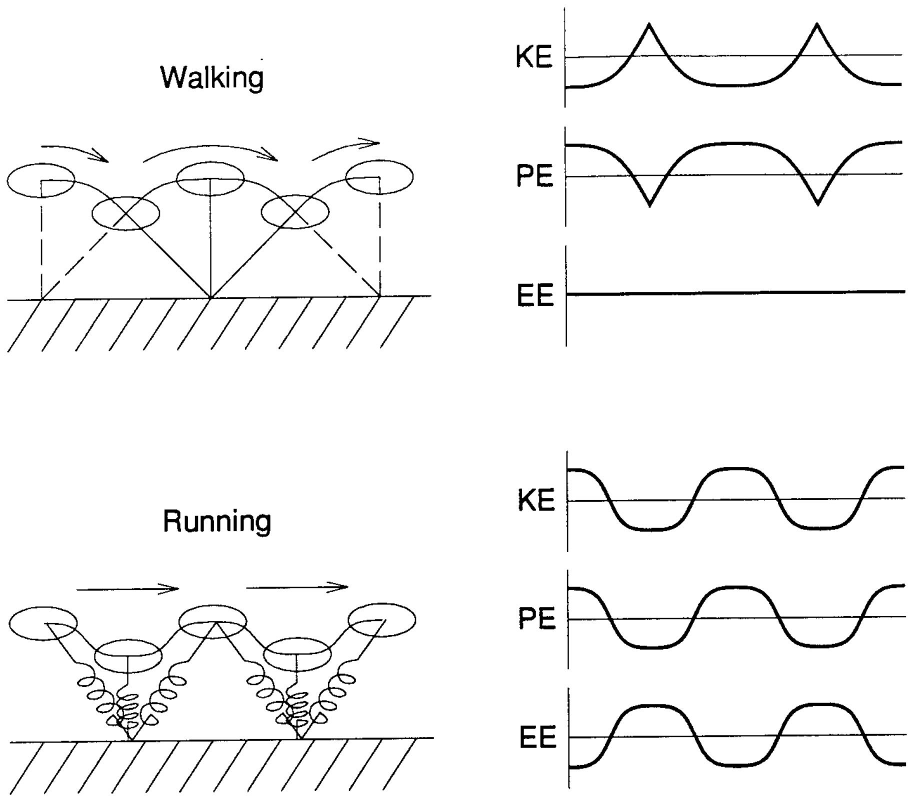


Figure 2–14: Energy fluctuations in walking and running. KE is the horizontal kinetic energy, PE is the gravitational potential energy, and EE is the stored elastic energy. While walking, the kinetic and potential energies oscillate in phase, but while running they are out of phase. For convenience, in the graphs on the right zero is set to the middle of the oscillation. The changes in value are important for gait classification; the baseline is not.

in order to conserve energy. Conversely, if the energies are out of phase their changes will cancel each other and elastic energy storage is less significant.

Full's experiments with actual cockroaches indicates that their gravitational potential energy and horizontal kinetic energy are in phase, corresponding to a running gait. As shown in Figure 2–16, our cockroach simulation had a phase shift varying from zero to 100 degrees over a wide range of speeds, all of which are indicative of elastic energy storage, and therefore a dynamic, running gait.

2.4.5 Conclusion

We examined the locomotion of a cockroach under various definitions of dynamic utilizing a physics-based simulation of a cockroach as a test platform. Although the gravitational potential energy is not precisely in phase with the horizontal kinetic over the spectrum of speeds tested, the phase difference is sufficiently small that elastic energy storage appears

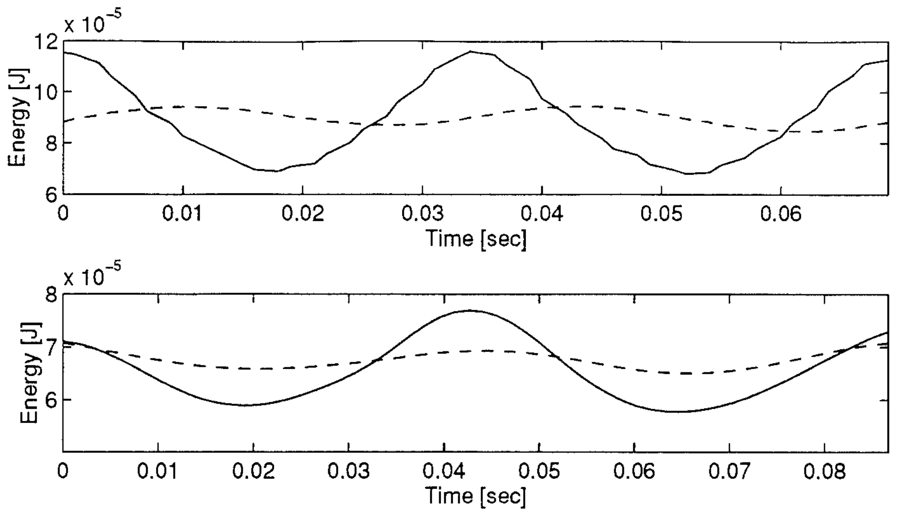


Figure 2–15: Kinetic and gravitational potential energies of the creature's center of mass. The top graph displays simulation energy data at an average speed of 26.1 cm/sec. The bottom graph displays energy data from an actual cockroach at an average speed of 24.5 cm/sec (Full & Tu 1990). The solid line represents the horizontal kinetic energy and the dashed line represents the gravitational potential energy. The potential energy plots have been shifted onto the kinetic energy plots to highlight the phase differences between the two.

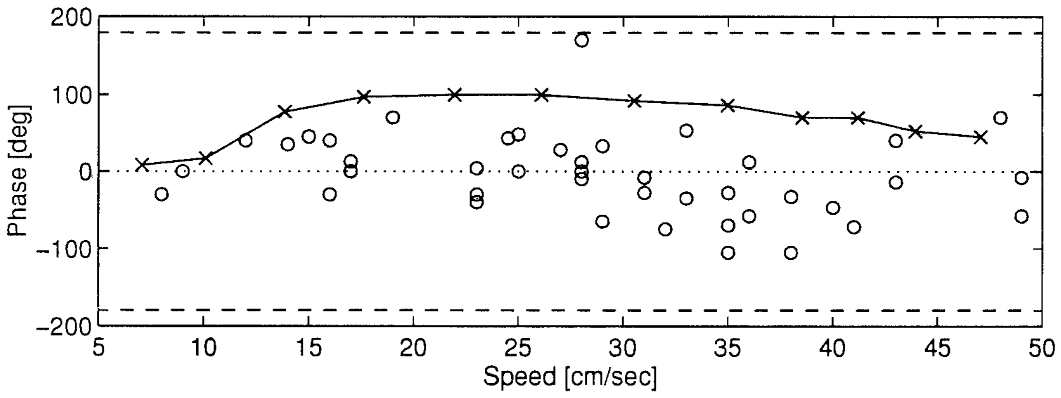


Figure 2–16: Phase shift between horizontal kinetic and gravitational potential energies of the creature's center of mass. Playback Cockroach simulation data (×) is plotted with phase data from an actual cockroach (○) (Full & Tu 1990). The dashed line at ±180 degrees indicates the point at which elastic energy storage becomes unnecessary.

important. Therefore, the locomotion of our simulated cockroach is dynamic: the simulation runs rather than walks.

However, our simulation is not dynamically stabilized. It remains statically stable, even when momentum effects are considered, except for brief moments at foot touchdown. If it were dynamically stabilized, we would expect that the cockroach simulation would be unable to run at such a variety of speeds without altering its stride patterns. To the extent that the simulation accurately reflects the locomotion behavior of a real cockroach, the simulation's stability suggests that an actual cockroach might be able to run dynamically over a wide range of speeds without a complex control system.

Our understanding of cockroach neuromotor control is limited and confined to guesses based on observed behavior. Also, we have neglected or only approximated certain phenomena that may be significant to cockroach locomotion, including cockroach leg link compliance and aerodynamic effects. Despite these shortcomings, our simulation's behavior is consistent with previously gathered cockroach data and we only suggest that pattern playback could be used to produce dynamic, stable cockroach running over a range of speeds.

The combined result of a dynamic, yet passively stable gait is interesting. It may mean that running, even with fewer than six legs, could be produced more simply than has been tried previously in existing, running, robotic systems. This result is significant because running allows higher speeds and potentially greater obstacle-crossing capability than walking. However, questions regarding the applicability of passively stable, dynamic gaits as well as implementation issues in a real robot control system remain for future study.

2.A Appendix: Hexahopper Simulation Parameters

Parameter	Metric Units	English Units
Overall length	.92 cm	.36 in
Overall height	.90 cm	.35 in
Overall width	.80 cm	.32 in
Hip height (max)	.62 cm	.24 in
Hip spacing (x)	0 cm	0 in
Hip spacing (y)	0 cm	0 in
Body mass	.262 g	9.34e-3 oz
Body inertia (x)	.017 g-cm^2	9.41e-5 oz-in^2
Body inertia (y)	.017 g-cm^2	9.41e-5 oz-in^2
Body inertia (z)	.026 g-cm^2	1.45e-4 oz-in^2
Upper leg length	.500 cm	.195 in
Lower leg length	.500 cm	.197 in
Upper leg mass	.0039 g	1.4e-4 oz
Lower leg mass	.0038 g	1.4e-4 oz
Leg total mass	.0077 g	2.8e-4 oz
Virtual spring stiffness	5000 mN/cm	904.2 ozf/in

2.B Appendix: Hexabug Simulation Parameters

Parameter	Metric Units	English Units
Overall length	4.3 cm	1.7 in
Overall height	.90 cm	.35 in
Overall width	2.0 cm	.79 in
Hip height (max)	1.2 cm	.47 in
Hip spacing ($x_{forward}$)	1.51 cm	.596 in
Hip spacing ($x_{rearward}$)	1.30 cm	.510 in
Hip spacing (y_{front})	.512 cm	.202 in
Hip spacing (y_{middle})	.670 cm	.264 in
Hip spacing (y_{rear})	.784 cm	.309 in
Body mass	1.245 g	.0445 oz
Body inertia (x)	1.2385 g-cm^2	.006856 oz-in^2
Body inertia (y)	.0938 g-cm^2	.000519 oz-in^2
Body inertia (z)	1.1817 g-cm^2	.006542 oz-in^2
Front femur length	.6 cm	.236 in
Front tibia length	.600 cm	.236 in
Front femur mass	.0017 g	6.07e-5 oz
Front tibia mass	.0017 g	6.07e-5 oz
Front total mass	.0034 g	1.21e-4 oz
Middle femur length	.750 cm	.295 in
Middle tibia length	.75 cm	.295 in
Middle femur mass	.0021 g	7.5e-5 oz
Middle tibia mass	.0021 g	7.5e-5 oz
Middle total mass	.0042 g	1.5e-4 oz
Rear femur length	.9 cm	.354 in
Rear tibia length	.900 cm	.354 in
Rear femur mass	.0025 g	8.93e-5 oz
Rear tibia mass	.0025 g	8.93e-5 oz
Rear total mass	.0050 g	1.79e-4 oz
Virtual spring stiffness	50000 mN/cm	9042.0 ozf/in

2.C Appendix: Cockroach Simulation Parameters

Parameter	Metric Units	English Units
Overall length	4.3 cm	1.7 in
Overall height	.912 cm	.359 in
Overall width	2.452 cm	.965 in
Hip height (max)	.73 cm	.29 in
Hip spacing (x_{front})	1.514 cm	.596 in
Hip spacing (x_{rear})	1.30 cm	.510 in
Hip spacing (y_{front})	.512 cm	.202 in
Hip spacing (y_{middle})	.670 cm	.264 in
Hip spacing (y_{rear})	.784 cm	.309 in
Body mass	2.2256 g	.0795 oz
Body inertia (x)	2.2142 g-cm^2	.0123 oz-in^2
Body inertia (y)	.1677 g-cm^2	9.28e-4 oz-in^2
Body inertia (z)	2.1126 g-cm^2	.0169 oz-in^2
Front coxa length	.629 cm	.248 in
Front femur length	.689 cm	.271 in
Front tibia length	.409 cm	.161 in
Front coxa mass	.0192 g	6.86e-4 oz
Front femur mass	.0094 g	3.36e-4 oz
Front tibia mass	.0029 g	1.04e-4 oz
Front total mass	.0315 g	1.13e-3 oz
Middle coxa length	.447 cm	.180 in
Middle femur length	.727 cm	.286 in
Middle tibia length	.744 cm	.293 in
Middle coxa mass	.0396 g	1.41e-3 oz
Middle femur mass	.0153 g	5.46e-4 oz
Middle tibia mass	.0076 g	2.71e-4 oz
Middle total mass	.0625 g	2.23e-3 oz
Rear coxa length	.541 cm	.213 in
Rear femur length	.898 cm	.354 in
Rear tibia length	1.313 cm	.517 in
Rear coxa mass	.0502 g	1.79e-3 oz
Rear femur mass	.0200 g	7.14e-4 oz
Rear tibia mass	.0120 g	4.29e-4 oz
Rear total mass	.0822 g	2.94e-3 oz

2.D Appendix: Force Distribution

Song and Waldron, for the Adaptive Suspension Vehicle, developed a method of distributing forces over multiple legs in contact with the ground (Song & Waldron 1989). A single virtual leg has a resultant force $\vec{R}$ to be applied to the body and a torque $\vec{T}$ to be applied to the body. However, the virtual leg consists of three physical legs, each of which needs a force $\vec{F}_i$ to be applied at its endpoint $\vec{X}_i$. We use Song and Waldron's algorithm to divide these forces. Using their notation, we present their method here simplified for the case of three legs. Appendix E2.E contains an extension of Song and Waldron's force distribution algorithm which we have not yet implemented.

Their algorithm has two parts. First, you determine the horizontal forces to apply at each foot, under the constraint that the feet cannot work against each other in the ground plane. Second, you calculate a set of vertical forces. For simplicity, we will define the origin as the center of mass of the robot.

Constraining the legs not to work against each other is roughly equivalent to the constraint on the velocities of points in a rigid body. Regardless of the object's total motion, the velocity vectors of individual points cannot bring the points any closer together. If you spin a disk with the correct center of rotation $\vec{\rho}$ and rotational velocity L, you can take the velocity vectors of points on the disk corresponding to the leg endpoints $\vec{X}_i$ and use them for foot forces. Once we calculate the disk's center of rotation and rotational velocity, the horizontal forces given by the feet are $\vec{F}_{xyi} = L\hat{z} \times (\vec{X}_{xyi} - \vec{\rho})$. L and $\vec{\rho}$ can be calculated from $\vec{T}$, $\vec{R}$, and $\vec{X}_i$. Calculate the centroid in the ground plane $\vec{C}$ and the mean squared distance between the centroid and the feet I.

$$\begin{aligned}\vec{C} &= \frac{1}{n}\sum_{i=1}^{3}\vec{X}_{xyi} \\ I &= \frac{1}{n}\sum_{i=1}^{3}(\vec{X}_{xyi} - \vec{C})^2\end{aligned}$$

$\vec{T}_z$, $\vec{C}$, $\vec{R}_{xy}$, and I uniquely determine the disk's center of rotation $\vec{\rho}$ and rotational velocity L.

$$\begin{aligned}L &= \frac{\vec{T}_z - \vec{C} \times \vec{R}_{xy} \cdot \hat{z}}{3I} \\ \vec{\rho} &= \vec{C} + \frac{\hat{z} \times \vec{R}_{xy}}{3L}\end{aligned}$$

The second step is to resolve the vertical forces $\vec{F}_{zi}$ for each foot. Song and Waldron assume that the distribution of forces F_{zi} is planar with respect to the foot positions on the ground. In our case, the fact that the distribution is only among three legs makes this constraint meaningless since any three points will designate a plane. We have included the constraint in our calculations so that the comparison to Song and Waldron's work is easier.

Let x_i, y_i, and z_i be the x, y, and z components of $\vec{X}_i$. We have already calculated $\vec{F}_{xyi}$, and need to calculate a F_{zi} such that:

$$\begin{aligned}
\sum_{i=1}^{3} F_{zi} &= R_z && \text{Resultant vertical forces} \\
\sum_{i=1}^{3}(y_i F_{zi} - z_i F_{yi}) &= T_x && \text{X axis torques} \\
\sum_{i=1}^{3}(z_i F_{xi} - x_i F_{zi}) &= T_x && \text{Y axis torques} \\
Ax_i + By_i + K &= F_{zi} && \text{Planar force distribution}
\end{aligned}$$

where the first three equations ensure we get the desired forces and torques.

Algebraic manipulation eventually leads to

$$\begin{aligned}
I_x &= \sum_{i=1}^{3}(y_i^2 - C_y^2) \\
I_y &= \sum_{i=1}^{3}(x_i^2 - C_x^2) \\
I_{xy} &= \sum_{i=1}^{3}(x_i y_i - C_x C_y) \\
U &= \frac{1}{3} R_z \\
V &= T_x + \sum_{i=1}^{3} z_i F_{yi} \\
W &= (\sum_{i=1}^{3} z_i F_{xi}) - T_y \\
A &= \frac{(V - 3UC_y)I_{xy} - (W - 3UC_x)I_x}{I_{xy}^2 - I_x I_y} \\
B &= \frac{(W - 3UC_x)I_{xy} - (V - 3UC_y)I_x}{I_{xy}^2 - I_x I_y} \\
K &= U - AC_x - B_y \\
F_{zi} &= Ax_i + By_i + K.
\end{aligned}$$

We avoid the singular case $I_x I_y = I_{xy}^2$ by making sure the legs are not collinear. It is possible to get negative values for F_{zi} if the legs are extremely poorly positioned, or $\vec{T}_{xy}$ is large. This corresponds to the cockroach attempting to pull itself down. We try to avoid this case as well, but if it happens we set the negative vertical forces to zero and rely on higher levels of control to correct the error.

2.E Appendix: Force Distribution Extension

Song and Waldron, for the Adaptive Suspension Vehicle, developed a method of distributing forces over multiple legs in contact with the ground (Song & Waldron 1989) (Appendix 2.D). Their method divides forces and torques into those parallel to the ground and those perpendicular to the ground, calculates forces parallel to the ground, and then calculates forces perpendicular to the ground. However, it only enforces the non-interaction force constraint on forces which are parallel to the ground. In a situation where a robot has incorrectly determined the vertical direction, the robot could assign forces in what it thinks is the vertical direction and have the legs interfere with each other. Additionally, if the ground is irregular, there is no well-defined "perpendicular to the ground."

We propose an extension to their method where the non-interaction force constraint is expanded to include all three dimensions. As before, let the resultant force to be applied to the center of mass be $\vec{F}^r$ and let the torque to be applied to the body be $\vec{T}^r$. The virtual leg has n physical legs, each of which needs a force $\vec{F}_i$ to be applied at its endpoint $\vec{X}_i$. Again, define the origin as the center of mass of the robot.

Divide $\vec{F}_i$ into the portion due to resultant forces $\vec{F}_i^f$ and the portion due to torques $\vec{F}_i^\tau$. Divide the resultant forces evenly among the legs $\vec{F}_i^f = \vec{F}^r/n$. There are no interaction forces among the leg forces due to resultant forces, because they are all parallel and equal magnitude. Calculate the torques resulting from the divided forces $\vec{F}_i^f$.

$$\vec{T}^f = \sum_{i=1}^{n} \vec{X}_i \times \vec{F}_i^f$$

Using a variation of Song and Waldron's non-interaction method divide the forces necessary to create the torque $\vec{T} = \vec{T}^r - \vec{T}^f$ among the feet. Constrain the forces to sum to zero, the torques to sum to $\vec{T}$, and the forces to be in a non-interacting vector field.

$$\begin{aligned} 0 &= \sum_{i=1}^{n} \vec{F}_i^\tau \\ \vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times \vec{F}_i^\tau \\ \vec{F}_i^\tau &= \vec{V} \times (\vec{x}_i - \vec{\rho}) \end{aligned}$$

Designate $\hat{v}$ as the unit vector in the $\vec{V}$ direction, so $\vec{x}_i = \vec{X}_i - \hat{v}(\vec{X}_i \cdot \hat{v})$, and solve for the central force field vector $\vec{V}$ and the central force field location $\vec{\rho}$. Substituting the third of the above equations into the first, gives:

$$\begin{aligned} \hat{v} \times \sum_{i=1}^{n} \vec{x}_i &= n\hat{v} \times \vec{\rho} \\ \vec{\rho} &= \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i \end{aligned}$$

Substituting the third equation into the second, you get a more complex result.

$$\begin{aligned}
\vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times (\vec{x}_i - \vec{\rho})) \\
\vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times \vec{X}_i) - \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times (\hat{v}(\vec{X}_i \cdot \hat{v}))) \\
&\quad -\frac{1}{n}\sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times (\sum_{j=1}^{n} \vec{X}_j)) + \frac{1}{n}\sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times (\sum_{j=1}^{n} \hat{v}(\vec{X}_j \cdot \hat{v})))
\end{aligned}$$

This equation simplifies because $\vec{V}$ is parallel to $\hat{v}$, so their cross product is zero.

$$\begin{aligned}
\vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times \vec{X}_i) - \frac{1}{n}\sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times \sum_{j=1}^{n} \vec{X}_j) \\
\vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times (\vec{X}_i - \frac{1}{n}\sum j = 1^n \vec{X}_j))
\end{aligned}$$

For convenience, define $\vec{C}_i = \vec{X}_i - \frac{1}{n}\sum_{j=1}^{n} \vec{X}_j$.

$$\begin{aligned}
\vec{T} &= \sum_{i=1}^{n} \vec{X}_i \times (\vec{V} \times \vec{C}_i) \\
\vec{T} &= \sum_{i=1}^{n} (\vec{X}_i \cdot \vec{C}_i)\vec{V} - \sum_{i=1}^{n} (\vec{X}_i \cdot \vec{V})\vec{C}_i
\end{aligned}$$

This leads to the following three by three matrix equation. Note that the vectors are considered column vectors, the prime notation indicates transposition, and **I** is the three by three identity matrix.

$$\begin{aligned}
\vec{T} &= (k\mathbf{I} - \mathbf{M})\vec{V} \\
k &= \sum_{i=1}^{n} \vec{X}_i' \cdot \vec{C}_i \\
\mathbf{M} &= \sum_{i=1}^{n} \vec{C}_i \cdot \vec{X}_i'
\end{aligned}$$

By inverting the matrix $(k\mathbf{I} - \mathbf{M})$, we can calculate $\vec{V}$. We already have $\vec{\rho} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$, so we can now use the central force field equation to determine the forces needed for torques $\vec{F}_i^T = \vec{V} \times (\vec{x}_i - \vec{\rho})$.

This force distribution guarantees that no two legs will work against each other in any direction, even on sloped, irregular, or poorly measured ground.

2.F Appendix: Height Control

Thrusting can be likened to instantaneously moving the rest position of the virtual spring at maximum compression, increasing or decreasing the amount of stored elastic energy as needed. Height control consists of determining the amount of energy to be added to the system every bounce cycle such that the Hexabug achieves a desired height on the next bounce.

Consider a single dimensional, vertical hopper. At maximum compression, the spring is compressed by a vertical displacement δ. If the rest position of the spring was moved by a distance ϵ, the amount of elastic potential energy stored in the spring would be:

$$E_e = \frac{1}{2}K(\delta + \epsilon)^2.$$

The point of least gravitational potential energy occurs at maximum spring compression. Assuming no dissipation, the energy required to bounce Δh higher than the altitude at maximum compression is:

$$E_{pe} = mg\Delta h.$$

Since the spring must supply the necessary E_{pe}, we set $E_{pe} = E_s$ and expand. For an increase in elevation, E_{pe} must be positive and we create a height error $\Delta h = z_d - z$ where z_d is the desired height. Solving for ϵ gives the following expression for the spring displacement increment:

$$\epsilon = \frac{\Delta h}{|\Delta h|}\sqrt{\frac{2}{K}mg|\Delta h|} - \delta.$$

For the Hexabug, we constrain the problem to a single dimension by projecting the virtual spring displacement onto the vertical axis. The spring displacement increment actually applied is then the leg-axis component of the vertical ϵ calculated above.

2.G Appendix: Hybrid Control of 3-Link Legs

During the development of the Hexabug simulation, control of a more realistic, 3-link, articulated, leg was considered. This serial manipulator contained a 2 degree-of-freedom universal hip joint followed by a distal revolute knee and ankle joint in a $\hat{z}$-$\hat{x}$-$\hat{x}$-$\hat{x}$ rotation configuration.

A Jacobian tensor relationship relates desired foot force, $\vec{F}_f$, to required joint torques. However, for a supplied 3-dimensional force vector, a 4-joint leg system remains configurationally unconstrained. We chose to maintain a desired angle, θ_d, between the final link and the inertial $\hat{z}$ vector as an additional kinematic constraint. As shown in Figure 2–17, θ is the angle whose magnitude is to be regulated. Since the four, joint degrees-of-freedom of the leg manipulator are used to control three degrees of force applied at the foot and one degree of spatial configuration, a hybrid control scheme is employed (Raibert & Craig 1981).

A positioning vector force, $\vec{F}_p$, applied normal to the leg link l_3 drives θ to θ_d as determined by a 2-link manipulator Jacobian, calculated for l_1 and l_2. The positioning force takes the form,

$$\vec{F}_p = [K_p(\theta - \theta_d) + K_d\dot{\theta}]\hat{F}_p$$

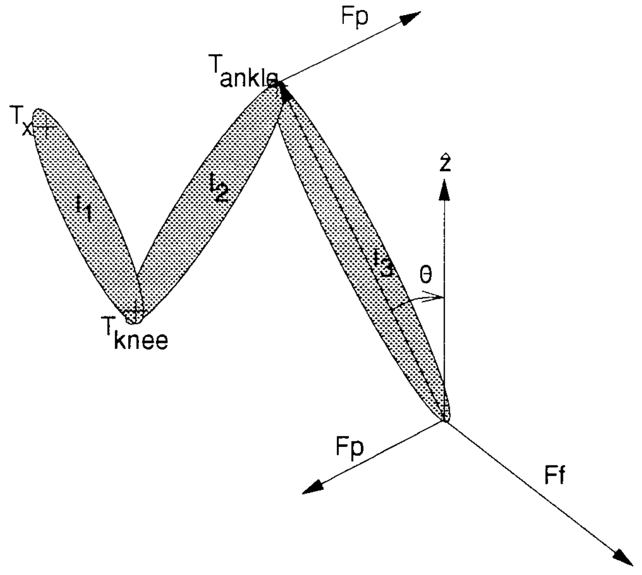


Figure 2–17: Schematic of the hybrid control scheme used to control the four degree-of-freedom, 3-link leg. A 3-dimensional desired force vector, F_f, is applied at the end of the manipulator. A desired angle, θ_d, is supplied as an additional, kinematic constraint.

where K_p and K_d are gains and the direction of $\vec{F}_p$ is determined by

$$\hat{F}_p = \frac{\vec{l}_3 \times an\hat{k}le}{|\vec{l}_3 \times an\hat{k}le|}.$$

This action specifies a set of positioning torques, $\tau_{\hat{z}}$, $\tau_{\hat{x}}$, and $\tau_{kn\hat{e}e}$. An equal and opposite force, -$\vec{F}_p$ is applied at the foot by an ankle torque of

$$\tau_{an\hat{k}le} = an\hat{k}le \bullet (\vec{l}_3 \times \vec{F}_p),$$

resulting in no net acceleration of the foot. Thus, by this application of internal forces, the beginning (hip) and end (foot) of the kinematic chain remains unmoved and l_3 effectively rotates about the foot, creating the regulated virtual joint angle θ. Since application of $\vec{F}_f$ determines a set of joint torques, superposition of these and $\tau_{\hat{z}}$, $\tau_{\hat{x}}$, $\tau_{kn\hat{e}e}$ and $\tau_{an\hat{k}le}$, specified by $\vec{F}_p$, results in the required hybrid control scheme that effectively applies $\vec{F}_f$ at the foot while regulating θ.

Figure 2–18 shows the results of application of the hybrid control scheme. One foreleg of the Hexabug has been replaced with a 3-link leg. The rest remain as the original, two-link legs. A desired angle of $\theta_d = .6$ rad was specified as shown by the dotted line. The solid line indicates the achieved angle. As shown in the top graph, two instances of deviation periodically occur. A major deviation occurs, followed sequentially by a minor deviation, two cycles later, at which time the body height falters. These deviations are causes as the

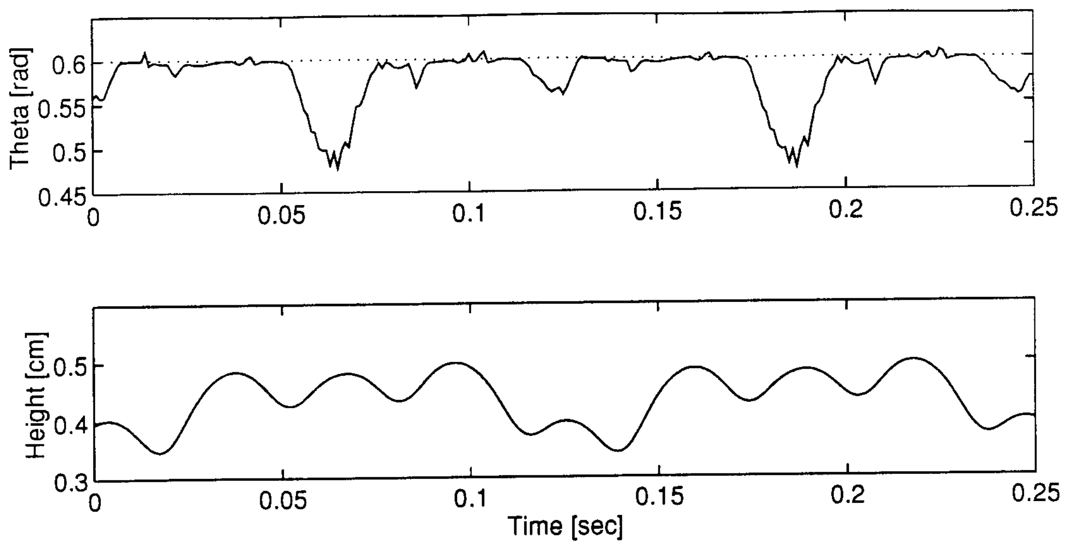


Figure 2–18: Hexabug leg data at a horizontal speed of approximately 27 cm/sec where leg 1 has been replaced with a 3-link leg. The top figure shows typical values for the virtual angle, θ, compared to the desired value indicated by the dotted line. The second figure illustrates the resulting period-4 variation in body height.

leg approaches a workspace boundary singularity in which the leg is physically unable satisfy both θ_d and maintain the current foot position. Since the Hexabug runs with an alternating tripod gait and only one leg has been replaced with the 3-link leg, even periodicity results as one tripod corrects for the other. A different choice of θ_d or link lengths could remedy this problem.

References

Cavagna, G. A.; Thys, H.; and Zamboni, A. 1976. The sources of external work in level walking and running. *Journal of Physiology* 262:639–657.

Full, R. J., and Tu, M. S. 1990. Mechanics of six-legged runners. *Journal of Experimental Biology* 148:129–146.

Hodgins, J. K., and Raibert, M. H. 1991. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation* 7(3).

Hogan, N. 1985. Impedance control: An approach to manipulation: Part i–theory. *Journal of Dynamic Systems, Measurement, and Control* 107(1).

J. Hodgins, J. Koechling, M. H. R. 1986. Running experiments with a planar biped. In *Third International Symposium on Robotics Research.*

M. H. Raibert, H. B. Brown, M. C. 1984. Experiments in balance with a 3d one-legged hopping machine. *International J. Robotics Research* 3:75–92.

McMahon, T. A.; Valiant, G.; and Frederick, E. C. 1987. Groucho running. *Journal of Applied Physiology* 62:2326–2337.

McMahon, T. A. 1985. The role of compliance in mammalian running gaits. *Journal of Experimental Biology* 115:263–282.

Raibert, M. H., and Craig, J. J. 1981. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement, and Control* 102.

Raibert, M. H.; Hodgins, J. K.; Playter, R. R.; and Ringrose, R. P. 1992. Animation of maneuvers: Jumps, somersaults, and gait transitions. In *Imagina.*

Raibert, M. H.; Chepponis, M.; and Brown, Jr., B. 1986. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation* RA-2(2).

Raibert, M. H. 1985. Four-legged running with one-legged algorithms. In *Second International Symposium on Robotics Research.*

Raibert, M. H. 1986. *Legged Robots that Balance.* Cambridge, MA: MIT Press.

Raibert, M. H. 1990. Trotting, pacing and bounding by a quadruped robot. *Journal of Biomechanics* 23.

Ringrose, R. 1992. The creature library. Unpublished reference guide to a C library used to create physically realistic simulations.

Song, S.-M., and Waldron, K. J. 1989. *Machines That Walk.* Cambridge, MA: MIT Press.

Sutherland, I. E., and Ullner, M. K. 1984. Footprints in the asphalt. *International Journal of Robotics Research* 3(2):29–36.

Chapter 3

Design and Control of a Planar Biped Rocker

David Bailey

3.1 Introduction

A truely graceful walking robot does not yet exist. We believe that a key ingredient for graceful walking is the smooth transfer of support, from one leg to the other. This paper describes the beginning of a project aimed at achieving a smooth step and a graceful walk.

The goal of this project is to study the transfer of support between the feet during a dynamic walking gait en route to smooth dynamic walking. We are not only interested in the impulsive forces that occur when a foot strikes the ground, but also how the center of pressure moves along the foot during a step cycle. More than just measuring this information, we are interested in discovering what, if anything, can be done to apply this information towards development of a control law. We are also interested in the preparations needed just before a foot touches the ground, in order to get a smooth exchange of support.

The GeekBot is a simple prototype robot. In designing the GeekBot, the goal was to create a very straightforward robot that could be built quickly, but was still capable of a rich set of behavior.

3.2 Background

Previous experiments with walking have concentrated on static stable locomotion. McGhee's group at the Ohio State University built an insectlike hexapod that would walk with a number of gaits, turn, walk sideways, and negotiate simple obstacles. Gurfinkel and his co-workers in the USSR built a machine with characteristics and performance quite similar to McGhee's (Okhotsimski *et al.* 1977)(Gurfinkel *et al.* 1981)(Devjanin *et al.* 1983). Hirose's work led to a leg with a special mechanical structure that simplified the control of locomotion and improved its efficiency (Hirose & Umetani 1980)(Hirose 1984).

In order for machines to exhibit the speed and grace of natural walkers, they must move with a dynamic gait and be actively balanced. Kato and his co-workers built a biped that walked with a *quasi-dynamic* gait (Ogo, Ganse, & Kato 1980)(Kato *et al.* 1983). This machine walked using a statically stable gait except for brief dynamic periods when it destabilized itself to tip forward so that support would be transferred quickly from one foot to the other. (Miura & Shimoyama 1984) built a walking machine that may have been the first to balanced itself actively. Their *stilt biped* was patterned after a human walking on stilts. (Furusho & Masubuchi 1987a) and (Furusho & Masubuchi 1987b) used an inverted pendulum model in conjunction with an hierarchical control scheme. (McGeer 1990) designed dynamic walking machines that travel downhill using no sensors, actuators, or computers. They rely on appropriate choices of machine geometry—link lengths, link masses, joint damping, walking surface slope, foot shape—to do the computing. Takanishi, Kato and their colleagues (Takanishi *et al.* 1990) built several robots that walk dynamically on simple rough terrain. Kajita and Tani (Kajita & Tani 1991) used a *linear inverted pendulum* model, where the body would move at a constant height above the ground. They also used ankle torques to help control walking cycle. Yoneda (Yoneda 1991) has created an interesting 3Dbiped that can walk both statically and dynamically, and can climb stairs.

To move with greater speeds, machines must begin to truly utilize their dynamics and to manage their energy better. One way of doing this is by having a ballistic motion, and transferring some portion of the machine's kinetic energy into potential energy, either gravitational during the flight phase or elastically stored in the legs during stance. Both two and three dimensional bipeds with such characteristics have been created at Marc Raibert's Leg Lab. Koechling (Hodgins, Koechling, & Raibert 1986) researched the parameters that limit the maximum speed of a legged system. Hodgins (Hodgins & Raibert 1991) found several methods of controling step length for rough terrain locomotiom. Raibert, Hodgins, and Playter (Hodgins & Raibert 1989)(Playter & Raibert 1992) have made their two and three dimensional bipeds perform somersaults.

For the most part, all of the previous work has been concerned soley with stabilizing the walking or running cycle. With the GeekBot, we are looking towards making improvements in how the feet are loaded, and how support is transferred between the legs. If this transfer can happen smoothly, the walking motion will become more efficient, as more energy is carried through each step.

3.3 Robot Design

Four major decisions were made early to guide the GeekBot's design:

1. The machine will be planar.
   The dominant behavior of walking can acceptably be reduced to the planar case. Also, a planar machine simplifies both the actuation and control required since out of plane effects can be ignored.

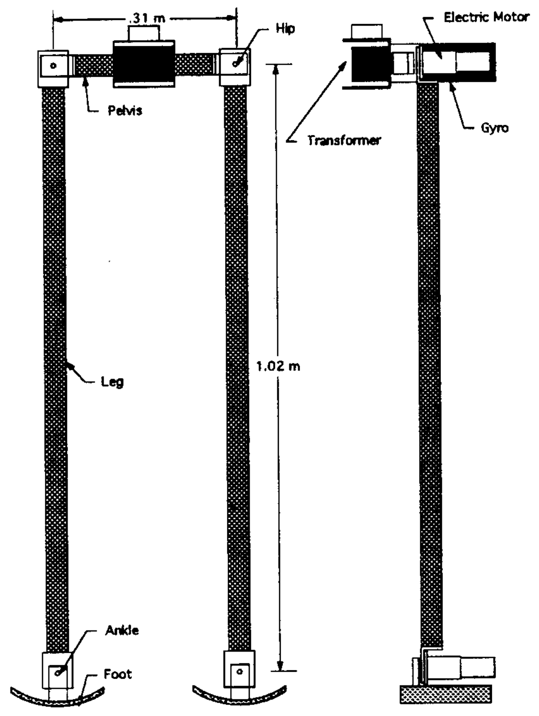


Figure 3–1: Schematic of GeekBot. The GeekBot stands approximately 1.09 meters tall and has a weight of around 7 kg. The four actuators are electric motors located at the various joints. There are potentiometers at each joint to measure relative angles and the mechanical gyro measures the inertial orientation of the pelvis. The feet have force sensing resistors to measure foot ground contact. Computing and electrical power is off board.

2. The machine will have two legs and actuated ankles.
   A minimum of two legs is necessary for walking, and additional legs would only add to the robots complexity, without a substantial increase in its capabilities. We need the actuated feet to allow us to control the load transfer dynamics.

3. The machine will be powered by electric motors.
   Electric motors were chosen basically for their controllability, and the ease of using them in a revolute joint.

4. The machine will not carry its own power supplies nor its own computing resources.
   Removing the weight of the power supplies and computing power greatly reduces the power requirements of the actuators. Again, the goal is to study smooth walking, not to develop an autonomous walker.

As shown in Figure 3–1, the robot consists of five links: a pelvis, two legs, and two feet. The legs are connected to the pelvis through revolute hips, and the feet are connected to the legs through revolute ankles. The axes of all of the joints are parallel, and are powered by collocated electric motors. The legs and pelvis are made of carbon fiber tubing, with aluminum lugs and brackets to connect to the motors. The feet are circular aluminum arcs with rubber pads on the soles. The pads serve both to cushion landing, and to provide some traction with the ground. The center of the arc is beyond the ankle. During initial trials, the GeekBot would inevitably end up rocking on the edges of the feet. By having curved feet, we can control the point on the foot that the robot is resting on. Since the point of contact is in actuality the center-of-pressure, the curved feet allow us to control around a link position, instead of around force information. The motors are powered by Copley current amplifiers, which in turn are controlled by our standard Leg Lab interface hardware.

The instrumentation of the robot is simple and includes:

- Potentiometers to measure relative joint angles
- A mechanical gyroscope to measure the inertial orientation of the pelvis
- Force sensing resistors to measure foot ground contact

By combining the relative joint angles with the inertial orientation obtained from the gyro, the inertial orientation of every link can be obtained.

3.4 Current Status

Limited static walking has been achieved, but this was mostly to verify the kinematic analysis. After experimenting with several open-loop control strategies, a closed-loop method for dynamic rocking has been developed. A finite state machine regulates the rocking motion. The state transition diagram is shown in Figure 3–2. The robot falls from single support on one foot, through a double support phase, and lifts off onto single support on the other

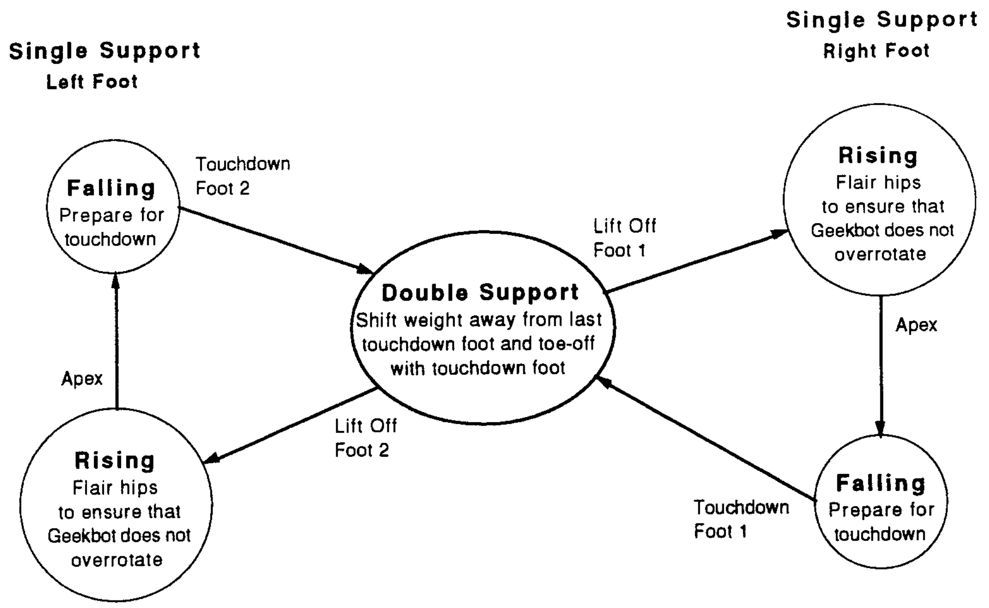


Figure 3–2: State Transition Diagram. The rocking control cycle is governed by the finite state machine described above. The action during each state is explained in the circles, and the transition condition is shown by the arrows.

foot. The trajectory of the center of mass of the robot is shown in Figure 3–3. By alternately increasing and then decreasing the stance width while the robot is in single support, a type of walking can be produced. Actually, it is more of a shuffle, as one foot is placed farther away, and then the other foot is brought back to the nominal stance width during the second part of the step.

Control of the rocking cycle has two basic concerns:

- Adding sufficient energy during double support so that the robot will rock into a single support phase
- Controlling the kinetic energy during single support so that the robot does not overrotate, and returns to double support

The first concern is handled by shifting the robot's hips towards the striking foot, and by kicking, or toeing-off, with the previous stance foot. Both of these actions add energy by getting the machine kinematically moving in the right direction for the next single support phase. The hip is just driven to a 90° angle. The toe-off is done by driving the ankle to an open-loop position, which can be controlled by the operator.

The second concern is addressed by comparing the the robot's rotational kinetic energy and the remaining gravitational potential energy available to store the kinetic energy. If there is too much kinetic energy, the robot's control system moves to dissipate the excess kinetic energy. When the robot flairs its hips, as shown in Figure 3–4, the center-of-mass moves away from the foot, and the moment of inertia increases. To develop a control law

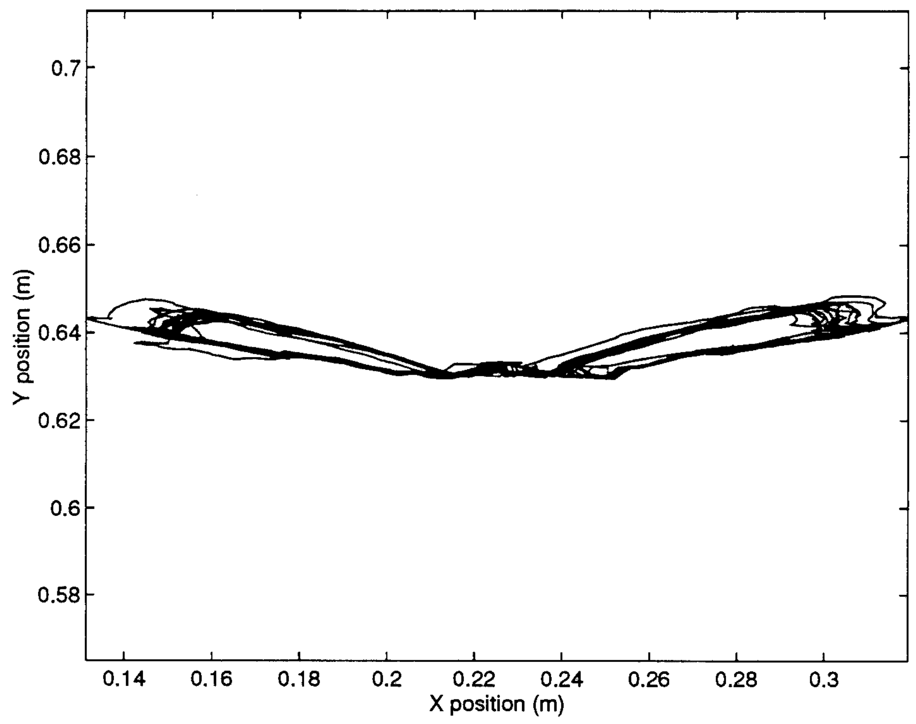


Figure 3–3: Center of Mass Trajectory. The center of mass rocks up and to the left or right during the first part single support. At the apex, the robot reconfigures itself, and the center of mass moves outward. The robot then falls back to double support, which is the flat portion in the middle of the plot.

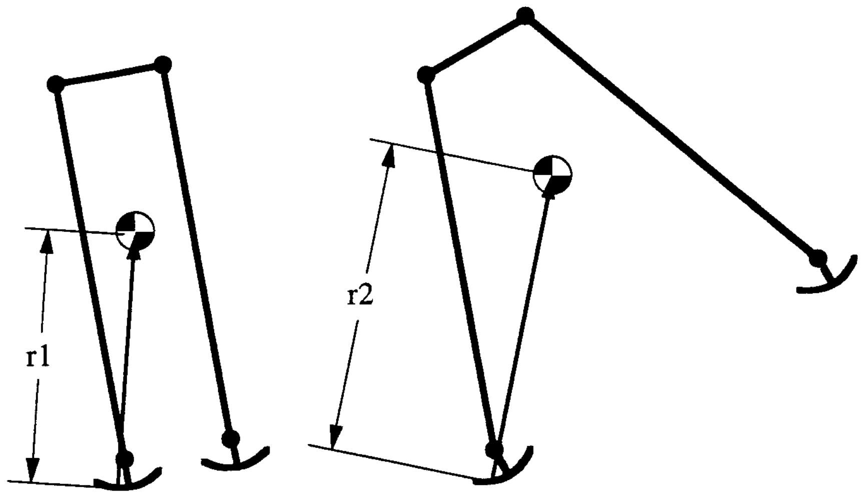


Figure 3–4: GeekBot with Flaired Hips. The figure on the left indicates the GeekBot configuration prior to flairing. The right-hand side depicts the GeekBot in "spread-eagle" configuration, with the hips flaired

that utilizes this movement, we modelled the robot's movement during single support as a simple inverted pendulum, shown in Figure 3–5.

The first step of the control is to compare the rotational kinetic energy to the work that gravity will do before the pendulum is totally inverted. This work is the difference between the potential energy when the pendulum is upright, and the potential energy at its current state. If there is more available potential energy than there is kinetic, the pendulum will not over-rotate, and no action needs to be taken. However, if there is more kinetic energy, than some action needs to be taken. If you assume that the robot can change its configuration instantaneously, then increasing the length and inertia of the pendulum will happen with conservation of momentum. Initially, if the pendulum has moment of inertia I_1, mass m at a radius of r_1, is at angle θ and angular velocity ω_1, subject to gravity g, then

$$T_1 = \frac{(I_1 + mr_1^2)\omega_1^2}{2}$$

and

$$V_1 = mgr_1(1 - \sin\theta)$$

where T_1 is the kinetic energy and V_1 is the remaining potential energy. Through the configuration change, momentum is conserved, so we can determine T_2 from a momentum (H) balance.

$$H_1 = H_2$$

$$(I_1 + mr_1^2)\omega_1 = (I_2 + mr_2^2)\omega_2$$

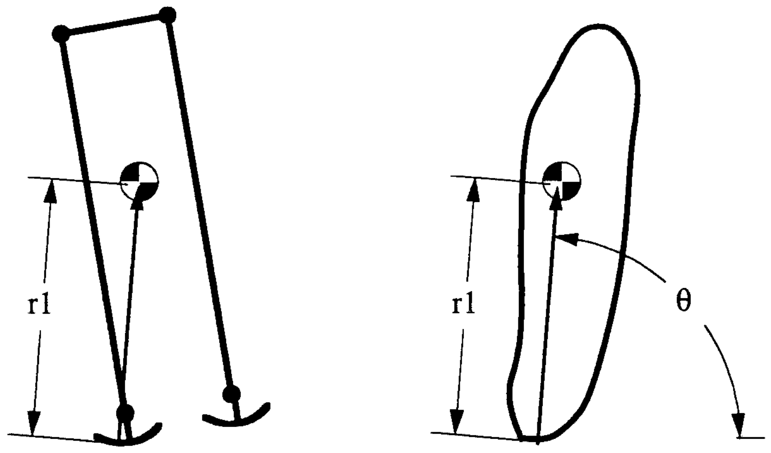


Figure 3–5: Inverted Pendulum Model of GeekBot in Single Support. The left diagram indicates the true robot configuration, while on the right the robot has been reduced to a simple rigid body model.

$$T_2 = \frac{(I_1 + mr_1^2)^2\omega_1^2}{2(I_2 + mr_2^2)}$$

also,

$$V_2 = mgr_2(1 - \sin\theta) > V_1$$

since $r_2 > r_1$ and $I_2 > I_1$, two things happen. Since $T_2 < T_1$, the kinetic energy decreases. With $V_2 > V_1$, the available potential energy increases. What this means is that flairing the hips both decreases the kinetic energy and increases the available potential energy and therefore improves things on both fronts.

This control strategy was implemented successfully on the robot, but not with out several modifications. First, when the robot flairs its hips, the center of mass must move in a straight along the original angle θ. The robot then rotates inertially to compensate for this. This movement creates a lot of low frequency noise into the angular velocity measurement. This signal is therefore put through a low pass filter. However, to adequately filter out the noise, the break frequency of the filter is quite low, and this introduces a substantial time delay. As added protection to stop an over-rotation, when calculating the robot's kinetic energy is given an additional feedforward value. This makes the robot flair its hips a least a little. Lastly, the configuration change is not instantaneous, as assumed early. In spite of theses limitations, the control strategy works adequately.

3.5 Looking Ahead

Looking towards the project's future, there are two major tasks. The first is to finalize the control methods that will produce a smooth rocking motion. The current idea about solving this problem is based on the concept presented in (Dunn & Howe 1994). The idea is to lengthen the stance leg while shortening the striking leg. If these actions are done correctly, a smooth exchange of support can be obtained, because the velocity of the center of mass before and after exchange-of-support will be the same. As an initial step towards this, the robot currently performs an open-loop "toe-down," where the striking foot is collapsing the same amount that the stance foot is toeing-off. The toe-down had a noticeable benefit in the conservation of energy through double support. We are currently looking at the mechanics of the touchdown, so that we can put a closed loop control around it. The toe-down could be shortening the strike leg, which would seem to agree with Dunn's method. It may also be doing a form of ground speed matching.

The second task is to develop a way of quantifying smooth walking. By developing this metric, we will be able to compare the affects of different strategies. Current thoughts on this include obtaining force plate measurements of the rocking motion, or using the maximum stance width that the robot will rock with.

References

Devjanin, E. A.; Gurfinkel, V. S.; Gurfinkel, E. V.; Kartashev, V. A.; Lensky, A. V.; Shneider, A. Y.; and Shtilman, L. G. 1983. The six-legged walking robot capable of terrain adaptation. *Mechanisms and Machine Theory* 18:257–260.

Dunn, E. R., and Howe, R. D. 1994. Towards smooth bipedal walking. *IEEE Journal of Robotics and Automation.*

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

Furusho, J., and Masubuchi, M. 1987b. Control of a dynamical biped locomotion system for steady walking. In Miura, H., and Shimoyama, I., eds., *Study on Mechanisms and Control of Bipeds*, 116–127. Tokyo: University of Tokyo.

Gurfinkel, V. S.; Gurfinkel, E. V.; Shneider, A. Y.; Devjanin, E.; Lensky, A. V.; and Shitilman, L. G. 1981. Walking robot with supervisory control. *Mechanisms and Machine Theory* 16:31–36.

Hirose, S., and Umetani, Y. 1980. The basic motion regulation system for a quadruped walking vehicle. *ASME Conference on Mechanisms.*

Hirose, S. 1984. A study of design and control of a quadruped walking vehicle. *International J. Robotics Research* 3:113–133.

Hodgins, J. K., and Raibert, M. H. 1989. Biped gymnastics. *International Journal of Robotics Research.*

Hodgins, J. K., and Raibert, M. H. 1991. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation* 7(3).

Hodgins, J.; Koechling, J.; and Raibert, M. H. 1986. Running experiments with a planar biped. In *Third International Symposium on Robotics Research.*

Kajita, S., and Tani, K. 1991. Study of dynamic biped locomotion on rugged terrain. In *IEEE International Conference on Robotics and Automation*, 1405–1411.

Kato, T.; Takanishi, A.; Jishikawa, H.; and Kato, I. 1983. The realization of the quasi-dynamic walking by the biped walking machine. In Morecki, A.; Bianchi, G.; and Kedzior, K., eds., *Fourth Symposium on Theory and Practice of Robots and Manipulators*, 341–351. Warsaw: Polish Scientific Publishers.

McGeer, T. 1990. Passive dynamic walking. *International J. Robotics Research* 9(2):62–82.

Miura, H., and Shimoyama, I. 1984. Dynamic walk of a biped. *International J. Robotics Research* 3:60–74.

Ogo, K.; Ganse, A.; and Kato, I. 1980. Dynamic walking of biped walking machine aiming at completion of steady walking. In Morecki, A.; Bianchi, G.; and Kedzior, K., eds., *Third Symposium on Theory and Practice of Robots and Manipulators.* Amsterdam: Elsevier Scientific Publishing Co.

Okhotsimski, D. E.; Gurfinkel, V. S.; Devyanin, E. A.; and Platonov, A. K. 1977. Integrated walking robot development. *Conference on Cybernetic Models of the Human Neuromuscular System.*

Playter, R. R., and Raibert, M. H. 1992. Control of a biped somersault in 3d. In *IFToMM-jc International Symposium on Theory of Machines and Mechanisms.*

Takanishi, A.; Lim, H.; Tsuda, M.; and Kato, I. 1990. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. *IEEE International Workshop on Intelligent Robots and Systems* 323–330.

Yoneda, K. 1991. Biped. Videotape of 3D Biped.

Chapter 4

Robot Tucked Somersaults

Robert Playter, Marc Raibert

4.1 Introduction

The somersault is a maneuver in which a performer jumps into the air and rotates once about a side-to-side axis before landing on the ground. The main requirement of a successful somersault is a balanced landing which in turn requires that the performer finish the somersault with a pre-specified body attitude. In this chapter, I explore attitude control techniques for producing stable landing configurations for the tucked somersault of a 3D biped running robot.

The tucked somersault is distinguished by maintaining the tuck position during most of the maneuver. Humans 'tuck' by holding their knees close to the chest with the knee joints flexed to fold the lower legs under the body. An important dynamic feature of the tuck somersault is that in most humans it involves rotation about the major principal axis of inertia. Since a rigid body rotating about its major principal axis is directionally stable, we might expect the tuck somersault to be stable in the sense that the axis of rotation will tend to maintain its inertial orientation. This stability in turn simplifies the control of body attitude at landing in the tuck somersault.

Directional stability of the spin axis during a tucked somersault means that a somersault control strategy need deal primarily with avoiding over-rotation or under-rotation about the somersault axis. Rotation rate about the somersault axis can be controlled by changing rotational inertia. Since angular momentum must be conserved during flight, increasing inertia will slow down the somersault rate while decreasing inertia will increase somersault rate. If the performer knows the time of flight, then control of somersault rate provides a means of controlling the somersault angle at landing.

In this chapter, I discuss somersault experiments with a 3D biped robot, (Figure 4–1). Figure 4–2 shows a sequence of photographs of the 3D Biped taken while the robot

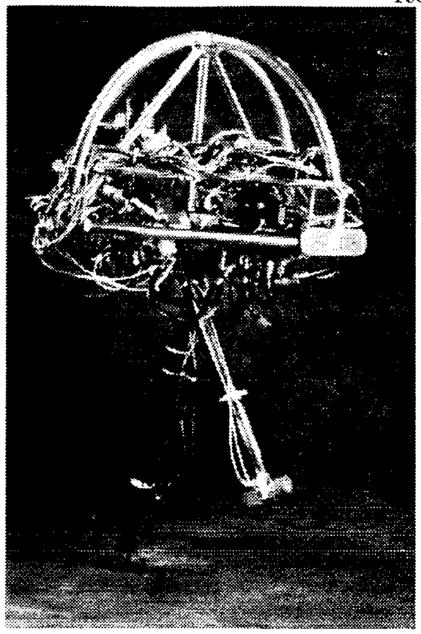


Figure 4–1: Photograph of the 3D Biped used for experiments. The body is an aluminum frame on which are mounted hip actuators and computer electronics. Each ball and socket hip joint has three low friction hydraulic actuators that can position the leg forward and aft, side-to-side, and can rotate the leg along the axis of the leg. A hydraulic actuator within each leg changes its length, while an air spring makes the leg springy in the axial direction. Sensors measure the lengths of the legs, the positions of the hip actuators, pressure in the hip actuators and contact between the foot and the floor. Gyroscopes measure the inertial attitude of the body. An umbilical cable connects the machine to hydraulic, pneumatic, and electrical power supplies. Control computations are done my microprocessors, some located on board and some nearby in the laboratory. Communications cables connect all processors.

performed a successful somersault. The somersault axis of the robot is coincident with the maximum principal axis of inertia making the robot somersault dynamically similar to the tuck somersault in humans. To initiate the somersault the biped runs forward, jumps to attain a double stance phase, then thrusts with both legs while pitching its body forward. Once airborne, the robot shortens its legs (tucks) to accelerate the forward rotation and swings its legs to a predetermined position with respect to the body. During flight the robot uses a feedback algorithm that changes the leg length to produce a rotation rate that will yield the desired somersault angle at landing. To accommodate errors in the estimated time of landing, the robot moves its feet to track the desired landing configuration as the system approaches the ground. The robot does not use any active control of out-of-plane rotation during flight. Rather it uses a broad stance during takeoff to minimize tilt rotation at the beginning of flight and it uses a broad stance during landing to minimize the effect of landing tilt errors. The robot has successfully performed the somersault in the laboratory. On its best day, the robot regained balance on landing to continue stable running on seven out of ten somersault attempts.

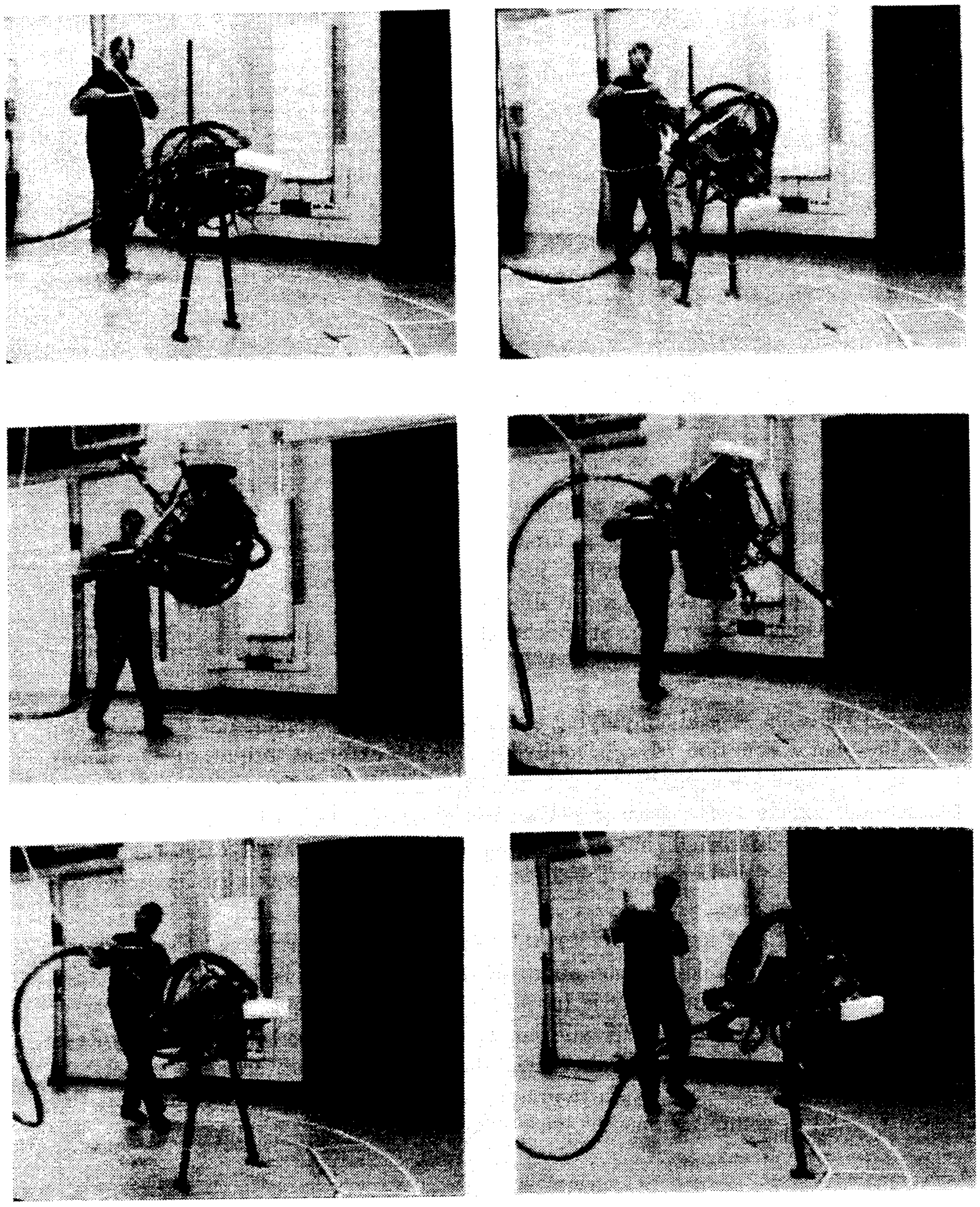


Figure 4–2: A sequence of photographs (arranged in left-right order starting upper left) taken during the execution of a somersault. The robot is running from left to right. Approximate relative time of each image: upper left–0.0 s, upper right–0.15 s, middle left–0.33 s, middle right–0.66 s, lower left–0.80 s, lower right–1.02 s.

4.2 The Mechanics of the Somersault

The most basic requirement of a somersault is that the performer neither over-rotate nor under-rotate the landing. Accurate control of the landing attitude allows careful placement of the foot relative to the center of mass of the robot which is a requirement for stable dynamic running (Raibert 84). Considering only the somersault degree of freedom, the attitude requirement is expressed by equating the time of flight and the time to rotate through the desired change in somersault attitude,

$$\frac{\Delta\Phi}{\dot{\Phi}_o} = \frac{\dot{z}_o + \sqrt{\dot{z}_o^2 + 2g(z_o - z_{td})}}{g} \tag{4.1}$$

where

$\Delta\Phi$	is the change in somersault angle required during flight
$\dot{\Phi}_o$	is the somersault rate of the body at lift-off
z_o	is the height of the center of mass (c.o.m.) at lift-off
$\dot{z}_o$	is the vertical velocity of the c.o.m. at lift-off
z_{td}	is estimated height of the c.o.m. at touchdown
g	is the acceleration of gravity

Equation 4.1 relies on several simplifying assumptions: 1) the somersault dynamics are governed by the planar equation $I\ddot{\Phi} = 0$, implying that the external torques due to supply cables or wind resistance are negligible, and 2) only the rotation in somersault is significant and the somersault axis is a maximum principal axis of inertia so that tilt and twist angles will stay small if they start small thus allowing us to ignore them, and 3) the legs do not swing with respect to the body during the flight phase, so $\dot{\Phi}_o$ represents the angular rates of both the body and the legs.

When the 3D Biped robot somersaults it rotates about its major principal axis. Figure 4-3 shows that the region of stable rotation about the somersault axis is large. We may then expect that as long as the somersault is initialized with the angular momentum vector close to the major principal axis then it will remain close to that axis. This in turn means that the tilt and twist angles of the robot at landing will be small and the somersault dynamics simplify as indicated above. In the remainder of this chapter we assume that this simplification is valid in computations involving rotational dynamics of the robot.

4.3 Somersault Control Strategies

The goal of the somersault control strategy is to produce a landing attitude that allows the robot to maintain balance. To regain balance on the landing, it is important that the robot achieve a desired horizontal displacement of the foot relative to the center of mass. In the plane of the somersault this horizontal displacement is given by

$$d = l \sin\beta \tag{4.2}$$

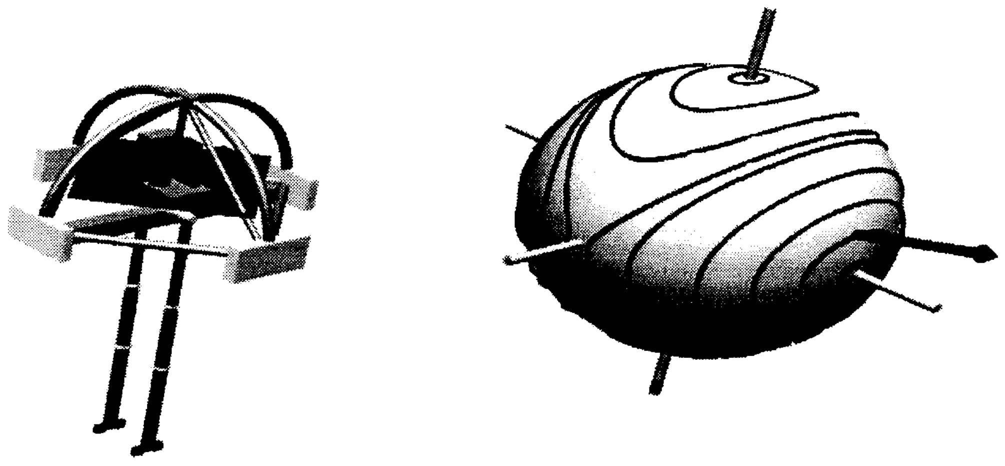


Figure 4–3: The somersault axis of the 3D Biped is coincident with the major principal axis of inertia. With the legs in the fully extended position, as shown here, the minor principal axis is the 'head-to-toe' axis.

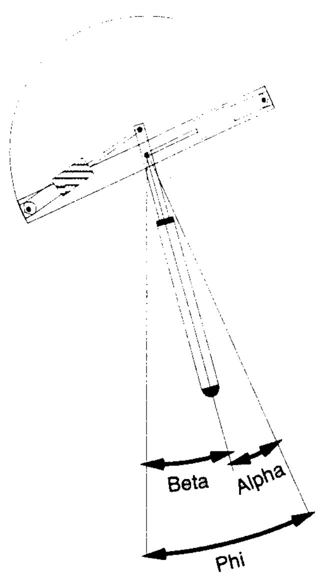


Figure 4–4: In order to maintain balance at landing the leg inclination angle, β, must be near the desired value. β is the angle the leg makes with the local vertical. It is a function of the the hip angle, α and the body attitude, Φ.

where l is the leg length and β is the leg inclination angle, the angle the leg makes with vertical in the plane of rotation (Figure 4–4). If the legs are held fixed relative to the body, then the desired landing attitude, or equivalently $\Delta\Phi$, can be found from d. If the initial conditions of flight, $\dot{\Phi}_o$, $\dot{z}_o$, and z_o, are established accurately so that for a desired z_{td} and $\Delta\Phi$ Equation 4.1 is satisfied, then the desired landing attitude will be achieved. More generally, one approach for generating somersaults is to establish accurately the initial conditions of flight to a state that is empirically determined, then to execute an open loop pattern of actuator signals to produce components of the desired behavior. Hodgins and Raibert used this approach in programming a planar biped robot to perform front somersaults with a 90% success rate. The success of such an approach depends upon how precisely one can reproduce the state of the robot and how sensitive the desired movement is to variations in the state.

For a running robot with less regular and repeatable motion, such as the 3D Biped, precise initialization is more difficult. Therefore, reliable production of a desired landing attitude might be improved with an in-flight feedback strategy that modifies the performance of the somersault based on the state of the robot in flight.

To the extent a system with non-zero angular momentum can change its inertia, it can also change its rotation rate. For a somersault, if the time until landing is known then control over rotation rate amounts to control over the landing attitude. The basis of the somersault control strategy is to change the robot somersault inertia to produce a rotation rate that will yield the desired body attitude at the time of landing.

The robot can change its inertia by extending or retracting (untucking or tucking) its prismatic legs. In flight, if the angular rate and moment of inertia of the robot in one configuration are $\dot{\Phi}_1$ and I_1 then with constant angular momentum the angular rate in a configuration with inertia I_2 is $\dot{\Phi}_2 = (I_1/I_2)\dot{\Phi}_1$. Joint limits restrict the range of inertia and thus limit control over the rotation rate. For the 3D Biped robot, the inertia about the somersault axis ranges from $1.22\,kg\,m^2$ with legs retracted to $1.50 kg\,m^2$ with legs fully extended. In moving from a fully retracted to a fully extended position the robot can reduce its somersault rate to 81% of its initial value. We will refer to the regulation of somersault rate and landing attitude via leg length as the *tuck servo.*

4.3.1 Control of Somersault Angle

To implement the tuck servo with state feedback, we first pose the requirement of Equation 4.1 as a function of the state during flight rather than at lift-off.

$$\frac{\Phi_{td} - \Phi}{\dot{\Phi}} = \frac{\dot{z}_o + \sqrt{\dot{z}_o^2 + 2g(z_o - \cos\beta_{td}\, l_{td})}}{g} - (t - t_o) \tag{4.3}$$

where

Φ_{td}	is the desired somersault attitude at touch down,
Φ	is the current somersault attitude,
$\dot{\Phi}$	is the current somersault rate,
β_{td}	is the desired leg inclination angle at touch down,
l_{td}	is the leg length at touch down,
t	is the current time,
t_o	is the lift off time.

The robot inertia and thus $\dot{\Phi}$ are functions of l_{td}, so we solve for the l_{td} that will make Equation 4.3 an equality.

We nominally require the hip angle to be zero on landing. If this is the case then $\beta_{td} = \Phi_{td}$ and the landing attitude and leg length determine the height of the hip. The center of mass is coincident with the hip, so z_{td} is given by

$$z_{td} = \cos\Phi_{td}\, l_{td} \tag{4.4}$$

We have also assumed that the tilt angle at touchdown is zero.

Define a function, f, as the difference between the left and right hand sides of (4.3).

$$f = \frac{\dot{z}_o + \sqrt{\dot{z}_o^2 + 2g(z_o - \cos\Phi_{td}\, l_{td})}}{g} - (t - t_o) - \frac{\Phi_{td} - \Phi}{\dot{\Phi}} \tag{4.5}$$

If the desired landing attitude is to be achieved then $f = 0$. Otherwise, the pitch rate needs to be increased or decreased depending on the sign of f. The control algorithm we use changes the leg to a length that will make $f = 0$. The function, f, is a non-linear function of the leg length, so we use a Newton search (Strang 86) to recursively solve for the desired leg length. The recursion uses the first order, Taylor series expansion of $f = 0$,

$$0 = f(l) + \frac{df}{dl}\Delta l \tag{4.6}$$

This yields the following equation for Δl

$$\Delta l = \frac{-f}{df/dl} \tag{4.7}$$

Ideally, once the desired leg length is achieved, f remains zero for the rest of the somersault and the desired leg length becomes the leg length at touch down, $l \rightarrow l_{td}$.

During each control cycle the tuck servo executes the following process

1. measure the state of the robot
2. compute f and df/dl_{td}
3. compute Δl_{td}
4. estimate the next value of f using $l_{td} = l_{td} + \Delta l_{td}$.
5. if $f \approx 0$ go to (6), else go to (2) using new values of f and l_{td}
6. servo the leg length to l_{td}.

To perform step 2 above we need to solve for df/dl_{td}. Taking the derivative of (4.5) with respect to l_{td} results in

$$df/dl_{td} = \frac{-\cos\Phi_{td}}{\sqrt{\dot{z}_o^2 + 2g(z_o - \cos\Phi_{td}\, l_{td})}} + \frac{\Phi_{td} - \Phi}{\dot{\Phi}^2}\, d\dot{\Phi}/dl_{td} \tag{4.8}$$

To obtain $d\dot{\Phi}/dl_{td}$ we use the fact that the angular momentum is a constant so that

$$\dot{\Phi} = \frac{h}{(I_o + 2m_l r^2)} \tag{4.9}$$

where

h	angular momentum
r	distance from the lower leg c.o.m. to the robot c.o.m.
m_l	lower leg mass
I_o	robot pitch inertia about c.o.m. when $r = 0$.

The total inertia of the robot as a function of leg length is $(I_o + 2m_l r^2)$. The $2m_l r^2$ term represents that inertia due to the distance of the lower legs from the robot c.o.m. This is the component of inertia that we control as the legs change length. From (4.9) we get

$$d\dot{\Phi}/dl_{td} = -\frac{2m_l r h}{(I_o + m_l r^2)^2}\, dr/dl_{td} \tag{4.10}$$

The robot center of mass moves very little when the legs are extended or retracted so we assume $dr/dl_{td} = 1$. Substituting for h from (4.9) results in

$$d\dot{\Phi}/dl_{td} = -\frac{2m_l r \dot{\Phi}}{(I_o + m_l r^2)} \tag{4.11}$$

Now by substituting (4.11), (4.8), and (4.5) into (4.7), we can compute the required Δl_{td}.

The computation of the desired leg length depends upon knowledge of robot parameters such as inertia and leg mass. However, since the process is repeated each control cycle based upon the sensed state of the robot, sensitivity to precise knowledge of these parameters is reduced. In exchange for this robustness to uncertainty we give up the ability to pre-specify both the landing attitude *and* the leg length at touchdown.

4.3.2 Accommodating Landing Time Errors

A limitation of the tuck servo strategy is its dependence upon accurate knowledge of the time until landing. With somersault rates on the order of 500 *deg/s*, small errors in the predicted landing time can result in intolerable landing attitude errors. Because we have no measurement of the vertical position or speed while airborne these quantities must be derived from estimates of the lift-off conditions. To accommodate errors in these estimates, we have modified the landing strategy to allow the robot to land with the desired configuration anywhere within a 'window' of the predicted landing time.

The leg inclination angle in (4.2) is the difference between the somersault angle and the hip angle, $\beta = \Phi - \alpha$. By changing the hip angle the control algorithm makes the foot track the desired displacement for a short time just prior to or after the expected landing time. In this way, the robot maintains the desired landing configuration during a 'landing window' that is centered about the nominal landing attitude. In order to maximize the landing window the legs are moved 'back' relative to the body ($\alpha > 0$) early in the flip. Prior to landing, the foot is swept forward in order to track the desired foot position until touchdown.

4.3.3 Control of Tilt and Twist Angles

Achieving a takeoff attitude with no tilt and twist is challenging with the 3D Biped. As the robot runs its body somersault angle stays close to zero but its body tilts and twists in phase with the stepping cycle as it runs. During a normal running cycle, the robot uses hip actuators during stance to apply torques to the body in order to control body attitude. However, we found that it was difficult to achieve small tilt and twist angles and rates at takeoff using only the hip servos. We found that the best method for keeping tilt and twist angles and angular rates small during takeoff was to use a wide double stance during the flip initiation in combination with the normal attitude control used during stance ((Raibert 84)). This wide double stance helps stabilize the robot's tipping motion during takeoff. An important component of this approach was to achieve simultaneous touch down of both feet at the beginning of the double stance phase. Similarly, a wide double stance during somersault landing and simultaneous touchdown of both feet were observed to minimize the effect of tilt and twist errors on the somersault recovery.

Table 4–1
3D Biped Parameters

Symbol	Description	Quantity
m	total mass	31.44 (kg)
m_l	lower leg mass	0.652 (kg)
I_o	somersault inertia	1.02 $(kg\, m^2)$
r_{min}	min. lower leg radius	0.404 (m)
l_{min}	min. leg length	0.647 (m)
l_{max}	max. leg length	0.862 (m)

4.4 Experiments with 3D Biped Somersaults

The 3D Biped is a two-legged robot that is free to translate and rotate in 3D space. Each leg is mounted adjacent to the center of gravity of the body with a ball and socket hip joint. The hip joint allows leg rotation about the x, y, and z axes ($\pm 20°, \pm 60°, \pm 15°$ see Fig. 1). Hydraulic actuators control each of these degrees of freedom. The robot's telescoping legs contain a fourth hydraulic actuator that acts in series with a pneumatic compression spring. Mass properties of the 3D Biped are included in Table 3.1 Some of the kinetic energy of the machine is stored in compression of the air spring during each bounce, and returned to power the subsequent flight phase. Energy is added to the hopping oscillation by actively compressing the air spring with the hydraulic piston during stance. The 3D Biped maintains balance while running by performing three control tasks (Raibert 84):

1. during stance, the robot maintains body posture by applying hip torques between the legs and the body,

2. during stance, the robot adds energy to the air spring to maintain the hopping oscillation, and

3. during flight, the robot positions the foot in anticipation of the next stance phase in order to control forward velocity.

To execute a somersault, the 3D Biped modifies three steps in an otherwise normal running sequence. The robot performs a hurdle step during which it hops higher than normal as it prepares to land on both feet for the flip step. The flip step is initiated by thrusting with both legs while pitching the body forward. During the landing step the robot lands on both feet then resumes a normal running gait. The control actions used to execute the flip are summarized in Table 3.2.

In laboratory experiments, the 3D Biped has successfully performed the forward somersault and regained a stable running cycle afterwards (Figure 4–2).

Table 4–2
Control Summary for the Somersault

Step	Action
Approach	Run forward ($\approx 1m/s$) with alternating gait
Hurdle	Pitch up slightly Hop with increased leg thrust Prepare to land simultaneously on both feet
Flip	Jump with maximum thrust Pitch body forward with maximum torque Shorten legs once airborne Servo hips to feet back position Engage tuck servo Prepare to land simultaneously on both feet Track desired foot position
Landing	Dissipate energy on landing Return somersault rate to zero, restore posture Adjust nominal leg length based on l_{td}
Following	Resume running with alternating gait

Three sets of data from a successful somersault and a nearly successful somersault are shown in Figures 4–5, 4–6, and 4–7. Figure 4–5 shows data for the approach, flip, landing, and continuation of running for a successful somersault. The nominal desired landing attitude was set to 350°. The robot ran steadily until the hurdle step at which time it hopped higher than normal as it prepared to land on both feet. During the flip step the body is thrust upwards and accelerated in somersault. The desired leg inclination angle was set to $-5.7°$ based on the forward speed in flight. The robot lands $0.080s$ earlier than anticipated with a somersault attitude of 325° and with a leg inclination angle of $-11°$. The actual leg inclination angle is much closer to its desired value than the corresponding values in somersault because of the feet-back position in flight. The foot positioning servo was not used in this somersault because the desired foot position was always slightly in back of the actual foot position, and the legs were already in a swept back configuration. Balance is regained on landing, but since the robot lands with a slight backward lean, forward speed is lost. Forward speed and posture are quickly restored during the following steps.

Note the oscillation in tilt angle during flight. This oscillation occurs because the robot took off with a non-zero tilt rate. Since rotation about the somersault axis is passively stable, the oscillation does not grow. We found that in order to regain balance after the somersault, the tilt angle on landing must be kept moderately small, $|\Theta| < 15°$. To do this we increased the stance width during the flip step, thereby providing a passively stable stance configuration in tilt during initiation.

The data in Fig. 4 shows the somersault action on a larger scale to illustrate the function of the tuck servo. Between the beginning of the flip step and lift off, the magnitude of the somersault rate, hip angle rate, and leg length all increase. At lift off, the somersault rate of the body declines rapidly as the legs are accelerated to the rotation rate of the body. The conservation of momentum constraint produces the symmetry between the absolute angular rate of the body and the relative angular rate of the hips. During this time, the legs are tucked to the shortest possible length.

The tuck servo is engaged as the hips reach the desired feet-back position and come to rest relative to the body. The robot has a somersault rate of 606 deg/s at the time the tuck servo is engaged. At this somersault rate, it is estimated that the robot will over-rotate by 84°. This error is illustrated by the third graph of Fig. 4 which compares the estimated time until touchdown and the estimated time until the desired somersault attitude is achieved. The tuck servo extends the legs to the maximum possible length to slow down the somersault rotation to 463 deg/s at which point it is estimated that the robot will land at nearly the desired attitude. The robot maintains this configuration until landing.

Data from another nearly successful somersault is shown in Fig. 5. In this somersault, the robot uses the foot positioning servo to keep from over-rotating. Once again the desired somersault landing attitude was 350°. The desired leg inclination angle was set to $-5.9°$. The robot lands $0.065s$ later than anticipated with a somersault attitude of 391° and with a leg inclination angle of $-3.2°$. As the robot detected that it was over-rotating it swept the feet forward quickly to track the desired foot position. This increased the somersault rate because of the conservation of angular momentum and contributed to the somersault attitude error on landing. A velocity measurement error during stance after landing led to the loss of forward speed of the robot so that it was momentarily supported by safety ropes

Table 4–3
3D Biped Flip Attempts

File	Outcome	β_{td} deg	Φ_{td} deg	$\dot{\Phi}$ deg/s	$\dot{z}_0$ (est) m/s	$\dot{z}_0$ (meas) m/s	$\dot{z}_0\%err$	speed$_{lo}$ m/s
92.346.4	success	0.12	-10.8	547	3.08	3.04	-1.2	1.04
92.346.5	success	-0.14	-6.29	529	3.48	3.33	-4.3	0.77
92.346.6	success	0.33	-8.47	524	3.41	3.32	-2.8	0.89
92.346.7	success	-0.31	-13.7	549	3.53	3.48	-1.4	1.10
92.346.8	6 steps	-1.95	-18.9	553	3.69	3.70	1.6	1.02
92.346.9	success	-0.39	-12.7	551	3.45	3.47	0.7	1.04
92.346.10	success	-0.04	-11.6	545	3.54	3.48	- 1.7	0.96
92.346.11	success	-0.28	-15.3	541	3.15	3.13	- 0.5	0.90
92.346.12	fall	-3.42	-1.04	529	3.28	3.04	- 7.4	0.89
92.346.13	fall	-1.87	-17.7	503	3.33	3.20	-3.9	0.88

before resuming running and therefore not considered a complete success.

Table 3.3 shows data compiled for ten somersault attempts performed by the 3D Biped in the laboratory. The robot successfully regained balanced running on seven of these attempts.

4.5 Summary

In this chapter, I discuss a strategy for robot somersaults that combines elements of feed forward control, feedback control, and passive dynamic stability. I also presented results from somersault experiments done in the laboratory on a 3D biped running robot. The somersault is initialized using pre-programmed patterns of action. In flight, a feedback strategy changes robot inertia to control the landing attitude of the somersault. The robot actively positions its feet to maintain a desired landing configuration during an interval surrounding the predicted landing time. The passive tilt stability inherent in a wide double stance is used to reduce tilt angle and rate on takeoff. The passive stability of a rigid body rotating about its maximum principle axis of inertia accounts for moderate tilt angles on touch down given moderate tilt angles and rates on lift-off.

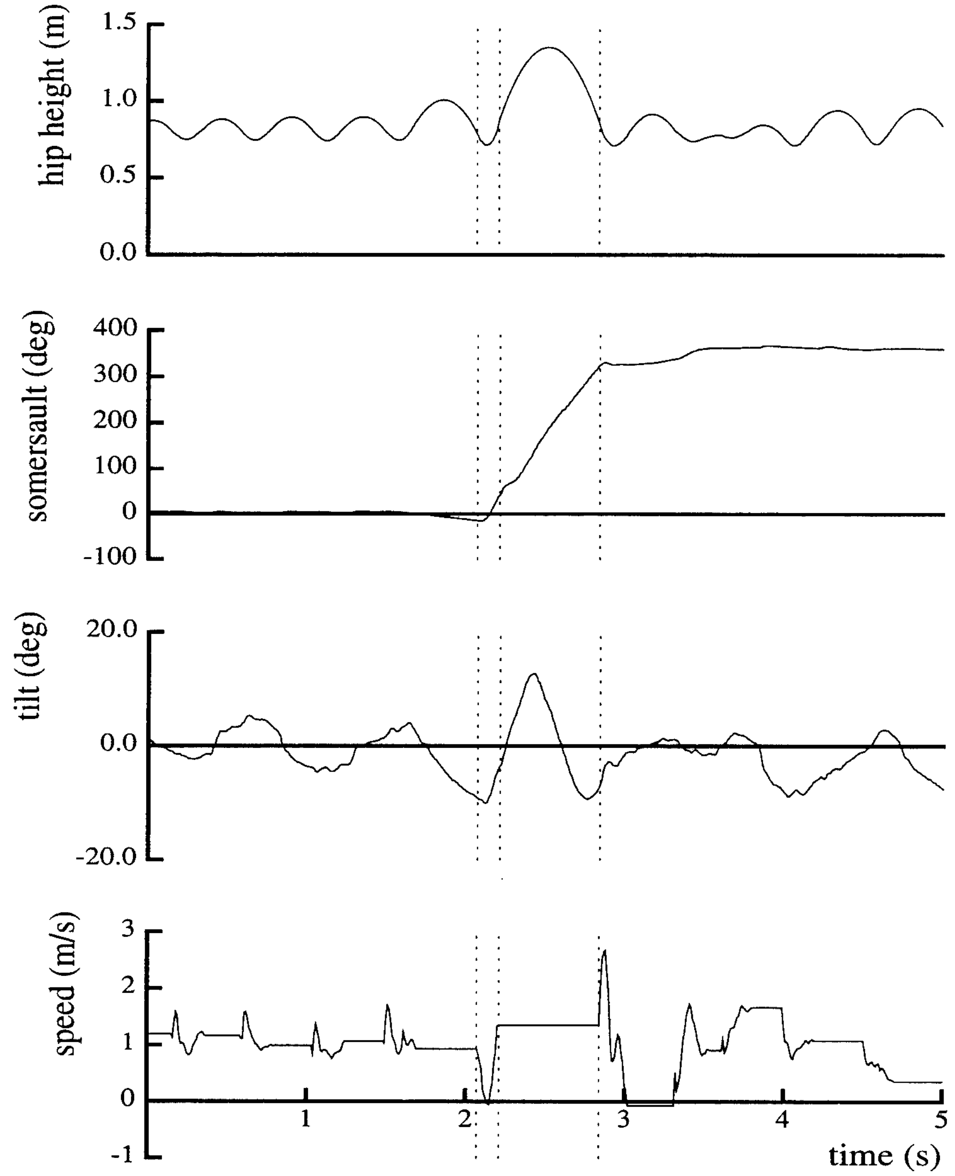


Figure 4–5: Hip height, somersault, and tilt angles, and forward speed of the 3D-Biped during a somersault. Vertical lines indicate initiation of flip, lift-off and landing. Data file B92.181.3

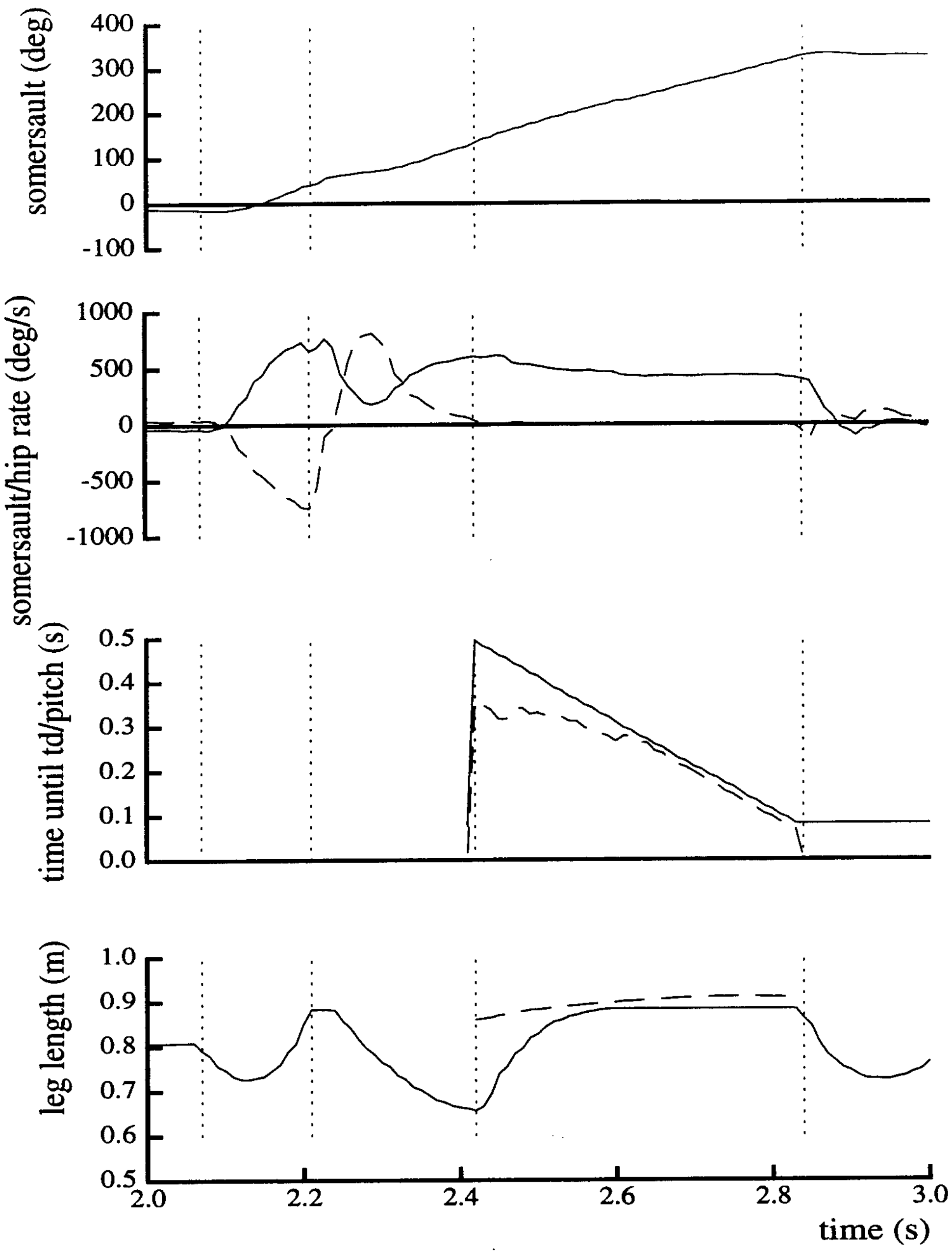


Figure 4–6: Somersault angle, absolute somersault rate (solid) and relative hip rate (dashed). The third graph shows the estimated time until touch down (solid) and estimated time until desired somersault attitude is achieved (dashed). The last graph is the measured leg length (solid) and desired leg length(dashed) for one leg. Vertical lines indicate initiation of flip, lift-off, tuck servo initiation, and landing. Data file B92.181.3.

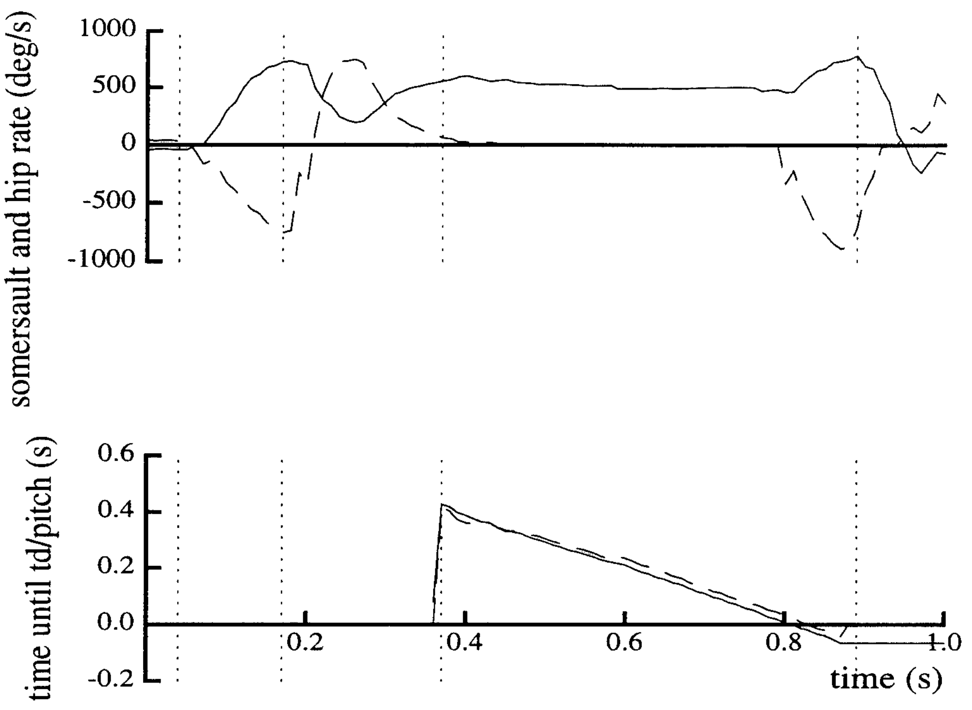


Figure 4–7: This somersault was over-rotating so the feet were swept forward at the end to track a desired foot position. Data shows absolute somersault rate (solid) and relative hip rate (dashed) and estimated time until touch down (solid) and estimated time until desired somersault attitude is achieved (dashed). Vertical lines indicate initiation of flip, lift-off, tuck servo initiation, and landing. Data file B92.181.4.

References

Raibert, M. H., H. B. Brown, Jr., and M. Chepponis, 1984. Experiments in Balance with a 3D One-Legged Hopping Machine. *International J. Robotics Research*, 3:(2) 75–92.

Strang, G. 1986.*Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA.

Chapter 5

Passively Stable Layout Somersaults

Robert Playter

5.1 Introduction

The layout or straight body somersault is considered to be inherently unstable because it requires rotation about the middle principal axis of inertia, an unstable equilibrium for a rigid body. Despite this instability, athletes perform this maneuver with apparent ease. Previously, biomechanics researchers have assumed that the athlete actively stabilizes the maneuver by making compensatory movements of the arms and body (0; 0; 0). Could it be that passive arm movement, movement that arises from body rotation and spring torques at the shoulder joints, stabilize the layout somersault with no active sensing or control? A passive stabilization technique is appealing because it potentially relieves the human performer of sensing small variations in movement and responding quickly enough to prevent instability. Instead, the athlete needs only to initialize the maneuver with a suitable shoulder spring constant tuned to the rotation rate and let the maneuver "unfold".

In this paper we verify the stability of the passive layout somersault in simple models with linear stability analysis, non-linear dynamic computer simulation, and laboratory experiments using a passive, human-like doll. The doll has spring-driven arms but has no other control system, sensors, or actuators. The doll routinely exhibits triple somersaults about its middle principal axis without twisting.

Passive dynamic stability is routinely designed into aircraft and spacecraft. However, the relevance of passive dynamics to the control of movement in biology and robotics is only beginning to be explored. The idea that relatively complex human movement could be governed largely by passive dynamics is demonstrated in ballistic walking. First proposed by Mochon and McMahon (0), a ballistic walker uses only gravity and the dynamic interaction of the swing and stance legs to produce a repetitive walking pattern. McGeer (0) showed the viability of ballistic walking by building passive, planar, anthropomorphic linkages that demonstrated stable walking down an incline.

In the following sections we review relevant background material from the field of biomechanics. Then we introduce the simple model used to study this problem. Next we examine the stability properties of the layout somersault. Finally, we present results from somersault experiments done in the laboratory on a human-like doll.

5.2 Background

Several biomechanics researchers have questioned how people control the layout somersault. Nigg recognized that the layout somersault may be unstable since it involved rotation about the middle principal axis (0). Hinrichs (0) measured the orientation of a performer's principal axes during a layout somersault to confirm that the athlete was rotating about the middle principal axes. He hypothesized that the athlete made small corrective movements of the arms and torso in flight to stabilize the somersault. When Yeadon (0) used body configuration data from a filmed double layout somersault as input to a dynamic simulation, he found that the maneuver appeared to be inherently unstable. He designed a linear feedback controller that used the sensed twist rate to manipulate (asymmetric) arm velocity in order to stabilize the layout somersault. Using a linear model of the dynamics he showed that an athlete would have to react to a growing twist instability within one quarter of a somersault in order to maintain stability. In the next section we introduce the dynamic model used to study the layout somersault.

5.3 Dynamic Model

Figure 5–1 shows a simple human model used to study the dynamic stability of somersaulting motion. The model has two shoulder pin joints that allow the arms to be raised and lowered. The head, torso, and legs are modeled as a single rigid body. Torques at the shoulder joints are provided by torsional springs. Fifteen parameters are required to describe this general model, but, by analyzing a linearized model of the dynamics, we reduce the stability analysis to the study of five dimensionless parameters that govern the behavior of the system.
We linearize the equations of motion for the case of steady somersaulting with the arms held symmetrically at a nominal angle of Φ_0 from the side. The resulting equations have the following form.

$$\Omega^2(M\ddot{z} + G\dot{z} + (K + K'/\Omega^2)\,z) = 0$$

where z is the state vector comprised of three Euler angles describing body attitude (somersault, tilt and twist angles) and two arm angles. The somersault angle describes the body rotation about the '2' axis. The tilt angle describes the inclination of the body relative to the somersault plane and the twist angle describes rotation about the long body axis. Ω is the nominal somersault rate. The state derivatives, $\dot{z}$ and $\ddot{z}$, have been made non-dimensional through scaling by Ω and Ω^2 respectively. With the equations of motion in this form, one can see that the nominal somersault rate only affects the magnitude of K' which contains the shoulder spring terms.

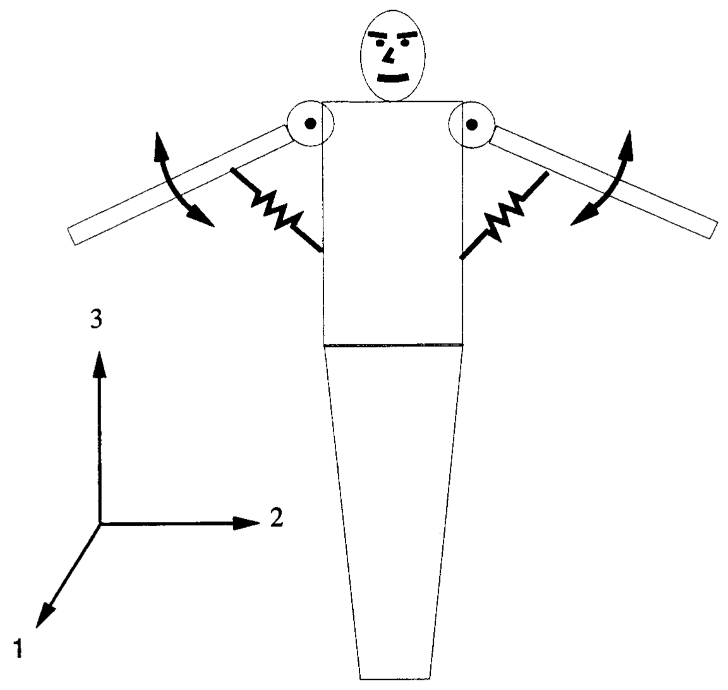


Figure 5–1: Three body model used to study layout somersaults. The model has pin joint shoulders with shoulder torques provided by springs.

The linearized five degree of freedom model decouples into two distinct subsystems. One subsystem is comprised of the somersault rotation and the symmetric motion of both arms. The other subsystem is comprised of the tilt and twist Euler angles and the asymmetric movement of the two arms. The twist stability of the five d.o.f. model can be studied by considering only the three d.o.f. subsystem of tilt, twist, and asymmetric arm movement. Further simplification of the equations of motion reveal that five non-dimensional parameters are important in determining the behavior of this system. They are the rigid body inertia ratios, k_1 and k_3, the relative size of the arm inertia, I_{rel}, the non-dimensional shoulder spring constant, K_{sh}, and an arm inertia ratio analogous to k_1 that we call k_{1arm}. To determine specific values of these parameters we use the estimated inertial data from the 1982 World Trampoline Champion, Carl Furrer (0).
The inertia ratios k_1 and k_3 represent the inertia of the equivalent rigid body, the rigid body that would result with the shoulder joints clamped in place.

$$\begin{aligned} k_1 &= \frac{I_2 - I_3}{I_1}, \quad -1 < k_1 < 1 \\ k_3 &= \frac{I_2 - I_1}{I_3}, \quad -1 < k_3 < 1 \end{aligned}$$

where I_1, I_2, and I_3 are the principal moments of inertia about the 1,2 and 3 axes of the whole body. For Furrer's data, with arms held straight out to the side, $k_1 = 0.65$ and $k_3 = -0.87$.
The relative inertia of an arm is given by $I_{rel} = I_{sh}/I_1$ where I_{sh} is the inertia of the arm about the shoulder. For Furrer $I_{rel} \approx 0.06$. The larger this number, the easier it is to stabilize the layout somersault. The non-dimensional shoulder spring constant is given by $K_{sh} = k/(I_{sh}\,\Omega^2)$ where k is the (dimensional) torsional shoulder spring constant in Nm/rad. Picking $K_{sh} = 1.0$ tunes the shoulder oscillatory mode to the rotation rate. The arm inertia ratio, k_{1arm} is very nearly equal to one for arms that are long and thin like those on a human. In the following section we use human inertial properties in the given model to study the stability of the layout somersault.

5.4 Stability Analysis

In this section, we present a linear stability analysis of the layout somersault. The goal is to determine the shoulder spring constants and arm angles that produce stable somersaults when the remaining body parameters are fixed. As a reference, we present the stability of the equivalent rigid body, that is the rigid body that would result if the arms were rigidly fixed to the body. Then we proceed to an analysis of the three-body system. A root locus plot shows that under some conditions, all poles of the linear system are simultaneously on the imaginary axis, implying stability. The results of the root locus are also presented in the form of stability diagrams that show which values of shoulder spring and nominal arm angle stabilize the layout somersault. These data were gathered using a search over the $K_{sh} - \Phi_0$ parameter space.

A rigid body rotating about its intermediate principal axis is unstable. A linearized analysis shows that the growth of the unstable mode is approximately governed by the

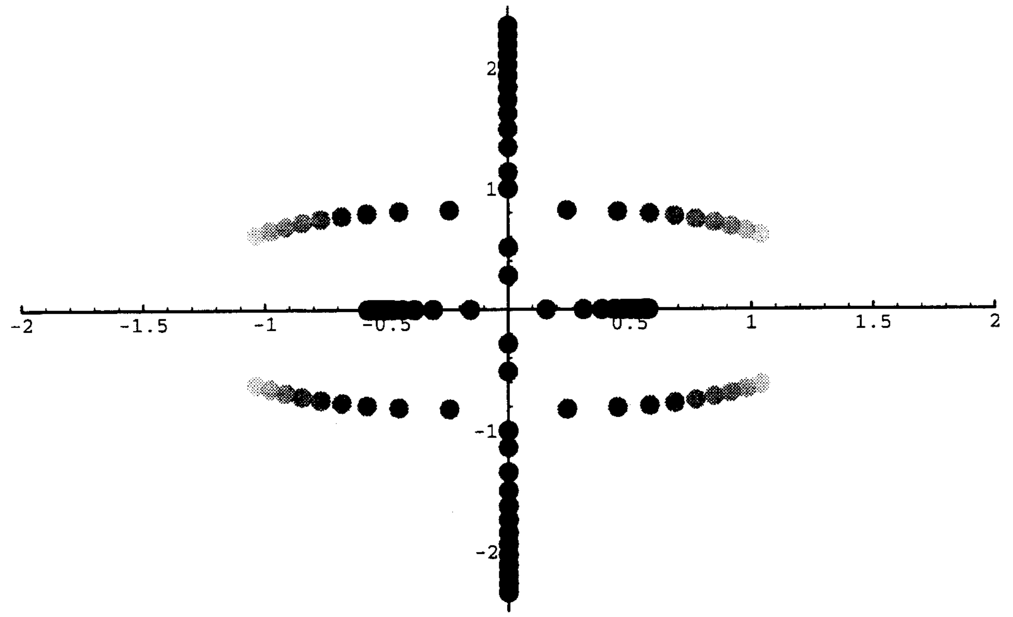


Figure 5–2: Root locus plot for the 3-by-3 subsystem of tilt, twist, and asymmetric arm movement. $\Phi_0 = 1.3$rad. Light dots indicate root locations for $K_{sh} \approx -2.0$. Dots get progressively darker as K_{sh} increases to $K_{sh} \approx 4.0$.

following equation

$$z = z_0 e^s \text{ where } s = (-k_1 k_3)^{1/2} \Omega\, t$$

Since the product of time and the nominal somersault rate, $\Omega\, t$, is simply the nominal somersault angle, the somersault angle required for the unstable mode to grow by a factor of N is given by

$$\Theta = \Omega\, t = \frac{ln(N)}{(-k_1 k_3)^{1/2}}$$

Using the data for Furrer, if $N = 10$ then $\Theta = 3.1$ rad or the unstable mode will grow by a factor of 10 in less than one half somersault.

Figure 5–2 is a root locus plot that shows how the location of the linear system poles of the three d.o.f. subsystem of tilt, twist, and asymmetric arm move as the shoulder spring constant is systematically varied for a fixed nominal symmetric arm angle. For some values of shoulder spring all of the roots of this system are located on the imaginary axis. These marginally stable conditions suggest that for these values of the parameters the physical system may be stable. No asymptotically stable points were found in this search as may be expected since energy is conserved in this model. Furthermore, adding shoulder damping tends to destabilize the system. This observation corroborates Hughes statement that damping tends to destabilize gyrically stabilized systems (0). Figure 5–3 shows the

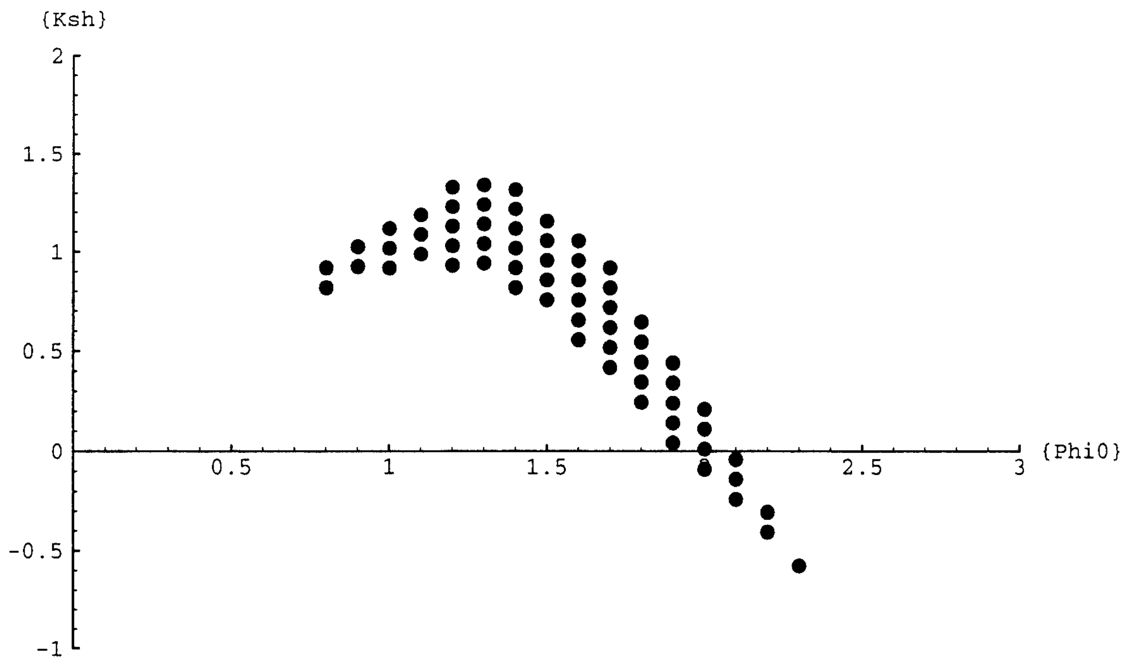


Figure 5–3: Theoretical stabilizing values of the non-dimensional shoulder spring, K_{sh}, as a function of the nominal shoulder angle for human data.

values of K_{sh} that stabilize the three d.o.f. system as a function of the nominal symmetric arm angle, Φ_0.

Neutral stability (i.e. all poles on the imaginary axis) implies that we can not conclude stability for the non-linear system. However, using representative values of K_{sh} from the linear analysis, non-linear computer simulations of the layout somersault starting with tilt deviations of 5° remain stable for as many as 105 full somersaults. To prove to ourselves that this behavior could exist in a physical system, we built a human-like doll for laboratory experiments. These experiments are described in the next section.

5.5 Experiments

The goal of somersault experiments with the human-like doll were to determine if passive layout stability was physically possible, and to empirically determine the shoulder spring constants that best stabilized the motion. The doll was built to be dynamically similar to the five d.o.f. model of the human performer studied in the last section. We built a mechanical launcher so that we could initialize the somersaults with consistency and with minimum direct human influence. The launcher functioned by accelerating the doll in rotation around a horizontally fixed bar then releasing the doll at a predetermined angle.

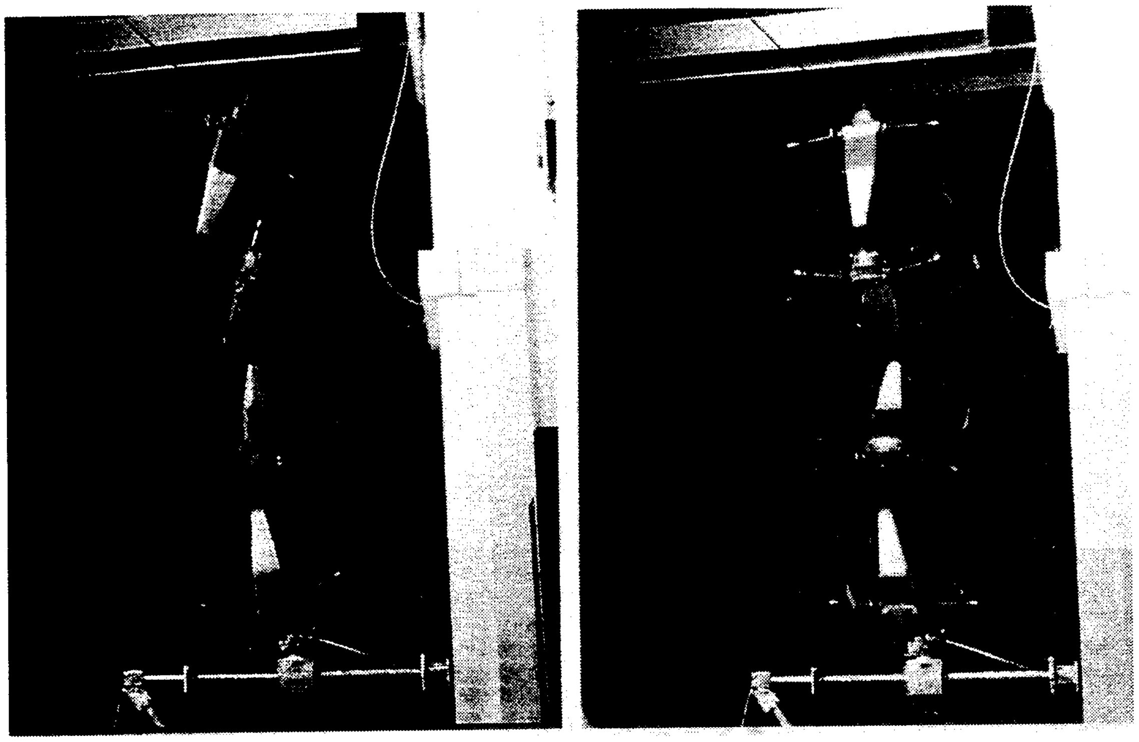


Figure 5–4: Strobed images of two separate experiments with the mechanical doll. The photo on the left show the rigid doll with arms clamped in place. This doll exhibits the twist instability. The photo on the right shows the doll with flexible arms performing a stable layout somersault.

The experiments consisted of a sequence of launches to determine which spring best stabilized the doll. During experiments the tilt and twist angles at launch were kept as close to zero as possible. The laboratory experiments showed that a human-like doll fitted with a carefully selected shoulder spring can consistently perform at least three stable layout somersaults. The doll had insufficient flight time for more somersaults.

The doll was sized to have approximately the same non-dimensional inertial parameters as a typical human performer. These parameters for the doll are $k_1 = 0.45, k_3 = -0.9, I_{rel} = 0.10$. Figure 5–5 shows the theoretically determined values of K_{sh} that stabilize the layout somersault for the doll.

Figure 5–6 shows the average number of stable somersaults performed as a function of the non-dimensional shoulder spring. The average is computed over twenty sequential launches for a single shoulder spring. Five different spring constants were tested. To test the stability of the effective rigid body, we clamped the arms in place. The standard deviation for each condition is shown with error bars. The average somersault rate was 16.6 rad/sec, and $I_{sh} = 3.021 \cdot 10^{-4} kg\ m^2$.

In addition to recording the number of stable somersaults exhibited during each launch we recorded whether or not the doll still appeared stable at the end of the maneuver. If the doll had not exhibited a quarter twist by the end of the maneuver then it was considered

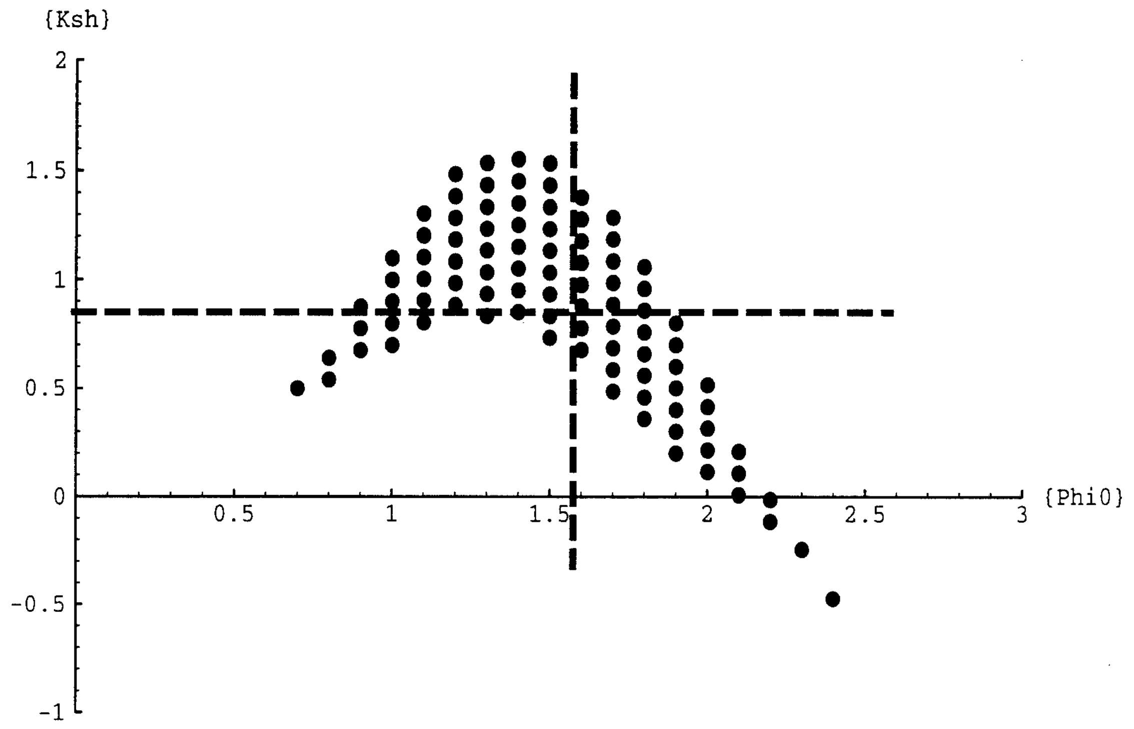


Figure 5–5: The points in this plot indicate the values of K_{sh} that theoretically will stabilize layout somersaults in the doll. The cross-hairs indicate the spring constant that best stabilized the doll during experiments.

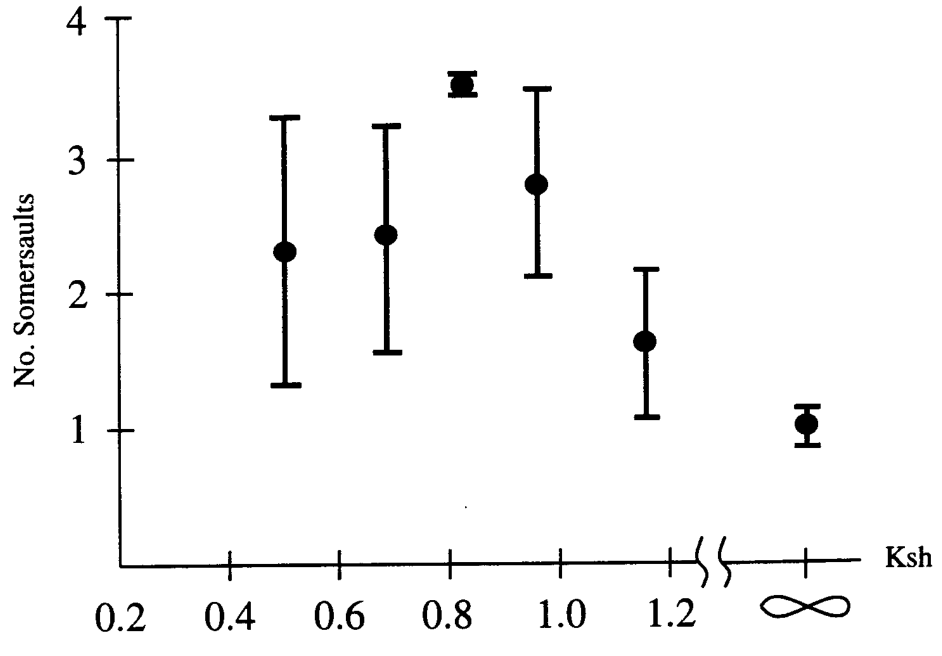


Figure 5–6: Plot of the average number of stable somersaults performed during twenty sequential launches of the doll as a function of the shoulder spring constant, K_{sh}. The standard deviation is indicated by error bars. The data for the effective rigid body is indicated by $K_{sh} = \infty$. The nominal symmetric arm angle was approximately 1.3 radians.

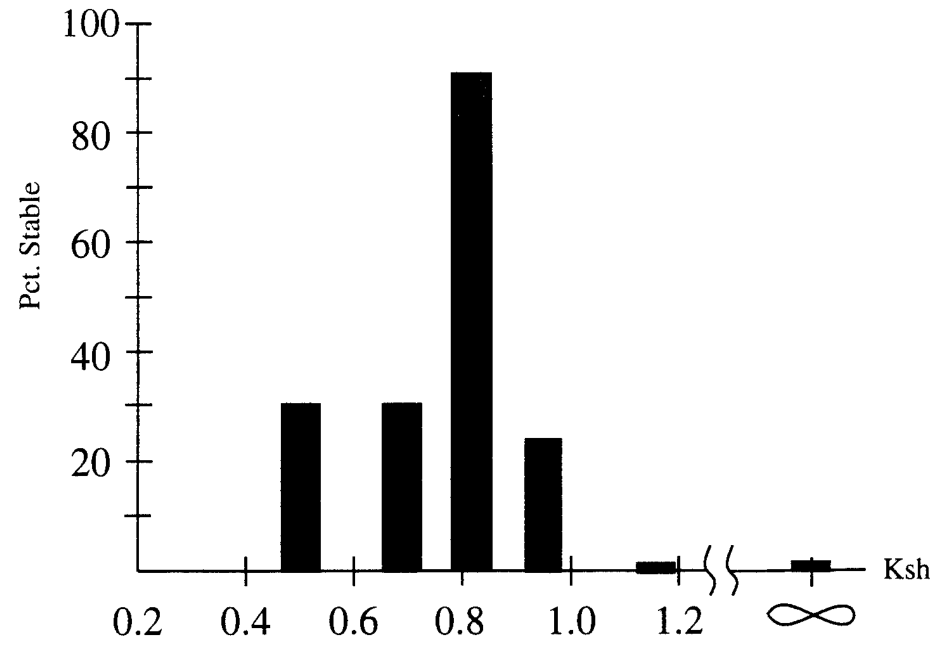


Figure 5–7: Plot of the percentage (of twenty) of completely stable dismounts as a function of the non-dimensional shoulder spring constant, K_{sh}. The two test conditions on the right side of the plot exhibited no completely stable somersaults.

to be a completely stable dismount. The percentage of completely stable dismounts as a function of K_{sh} is shown in Figure 5–7.

Of the six conditions tested, (five springs plus rigidly fixed arms) one shoulder spring value clearly outperformed the rest. The shoulder spring, $K_{sh} = 0.82$, performed 18/20 dismounts without exhibiting the twist instability. This high percentage of completely stable dismounts led to the very small variance of this condition as shown in Figure 5–6. If we could have observed more somersaults, the doll may have exhibited a variance more in accordance with the remaining data. The variance in number of stable somersaults appears to increase with softer shoulder springs. At the other extreme, the doll with fixed arms reliably exhibited the twist instability at about one complete somersault from release.

5.6 Discussion

In this paper, we discuss the control of the layout somersault. We show that the layout somersault can be passively stabilized for at least the two or three somersaults that are typical of human performances. This result suggests that passive dynamic behavior can play a role in aerial maneuvers that was previously attributed to active control by the athlete. It can compensate for errors in initial conditions or imprecise movement in flight and can stabilize maneuvers that were considered to be inherently unstable. We show that passive dynamic behavior can be tuned by the athlete through selection of nominal body configuration and passive mechanical parameters like joint spring constants.

The goal of this research is to develop a theory of passive dynamic aerial maneuvers. By showing that passive dynamics can play a significant role in the performance of reliable aerial maneuvers we argue that they reduce the need for active control from the athlete. This may in turn suggest that rather than focus on active control of each maneuver, athletes may choose a performance strategy that optimizes the use of passive dynamic behavior. Such a strategy might involve finding combinations of nominal aerial trajectories and passive mechanical parameters of the body that together produce the most reliable performances. Despite showing that passive dynamics could play a role in human aerial maneuvers we have not shown that they do.

5.7 Acknowledgments

We appreciate David Bailey's contribution to this project. Dave helped to design and construct the launcher and dolls and caught, with remarkable grace, many flipping dolls.

References

Hinrichs, R.N. 1978. *Principle Axes and Moments of Rotary Motions.* Masters Thesis. University of Iowa.

Hughes, P. C., 1986. *Spacecraft Attitude Dynamics.* John Wiley & Sons, Inc.

McGeer, T. 1989. Powered flight, child's play, silly wheels and walking machines, *IEEE Conference on Robotics and Automation*, Phoenix.

Mochon, S., McMahon, T. A. *Ballistic Walking.* J. Biomech. 13:49-57.

Nigg, B.M., 1974. Analysis of Twisting and Turning Movements. *Biomechanics IV.* London: MacMillan. 279–283.

Yeadon, M. R., 1990. The Simulation of Aerial Movement I-IV. *J. Biomechanics* 23(1):59–89.

Chapter 6

Twisting Somersaults

Robert Playter, David Bailey

6.1 Introduction

The twisting somersault is a maneuver in which the performer simultaneously rotates about the somersault and twist axes of the body. Multiple twisting, multiple somersaulting maneuvers are among the most exciting and complex aerial maneuvers performed by gymnasts and other athletes. Unlike the pure somersault, the twisting somersault must include non-linear rotational coupling between the different body axes. One effect of this non-linearity is that the effect of the performer's control actions will change during a maneuver making cause and effect relationships more complex than in the pure somersault. Navigation and feedback control of twisting maneuvers are challenging tasks. Does the accurate, reliable performance of this maneuver necessitate a feedback control strategy?

In this chapter we briefly discuss the open loop control of twisting somersaults. Open loop control means the performer's control actions are simply replayed from memory. We would like to know if twisting somersaults could exhibit passive dynamic stability when performed with an open loop control strategy. We discuss one test we performed to look for evidence of passive dynamic stability in the twisting somersault.

We present results from the non-linear dynamic simulation of a 1 1/2 twisting front somersault. This maneuver requires that the performer execute a sequence of limb motions during flight. We found that when a simulated performer used a prescribed set of motions for executing this maneuver, the landing attitude of the performer was sensitive to initial conditions leading to poor landing attitudes. However, when the performer's control movements were compliant, the reliability of the landing attitude was significantly improved. There appears to be an optimal choice for the performer's compliance that leads to the most reliable landings.

We also discuss the control of twisting somersaults by the 3D Biped robot. We programmed a simulated 3D Biped robot to perform a front somersault with half twist. We found that in order to make the robot maneuver look like a front somersault with twist as performed by a human, we had to add weight to the robot to make its moments of inertia more like those of a human. We also had to use stronger actuators than available for the physical robot. Our experiments to make the real 3D Biped robot perform the maneuver in the laboratory were unsuccessful. The physical robot actuators had insufficient actuator power to perform the maneuver.

6.2 The Tilt of Twisting Somersaults

Frohlich (Frohlich 80) described two techniques for performing twisting somersaults. In the **torque twist** the athlete derives rotation about the somersault and twist axes from external forces as he or she leaves the ground, diving board or other apparatus. In the **torque-free twist with angular momentum** the athlete initiates twist from an airborne somersault with an asymmetric movement of the limbs (Figure 6–1). The net effect of either technique is to tilt the principal axes of the body relative to the angular momentum vector. Even a small amount of tilt of the principal axes from the layout somersault position will result in twisting. (This is what makes layout somersaults challenging.) The greater the tilt angle, the greater will be the twist rate.

Figure 6–2 shows a sample map for the possible rotational trajectories of a 'rigid' human body. Each trajectory shown on the sphere corresponds to a different kinetic energy of rotation. The sphere rotates with the body so that its axes remain parallel to the principal axes of the body. The angular momentum vector is shown protruding from the sphere. The sphere (and body) must move so that the (inertially fixed) angular momentum vector remains in the 'slot' that is appropriate for the given kinetic energy.

The stable and unstable axes of rotation are obvious from this map. The stable principal axes are surrounded by trajectories that enclose the axes while the unstable middle principal axis shows trajectories that converge then diverge from that axis. The body motion that results when following one of these diverging trajectories is a twisting somersault maneuver. It is easy to see that when rotating about the middle principal axis, a small amount of principal axis tilt will put the body on a twisting somersault trajectory.

6.3 One and One Half Twisting Front Somersault

Yeadon (Yeadon 84) discusses a "torque-free" twist technique based on the hula movement for initiating twist from a piked front somersault. The hula movement involves a swiveling of the hips not unlike that required to swing a hula hoop about the hips. A quarter cycle of hula movement performed during a pike front somersault will tilt the performer's principal axes relative to the angular momentum vector. This effect is increased if the arms are held in an abducted position during the movement. After the hula movement the performer extends from the pike and the arms are adducted to decrease the inertia about the twist axis. The extension from the hula movement should happen between the 1/4 and 3/4 twist

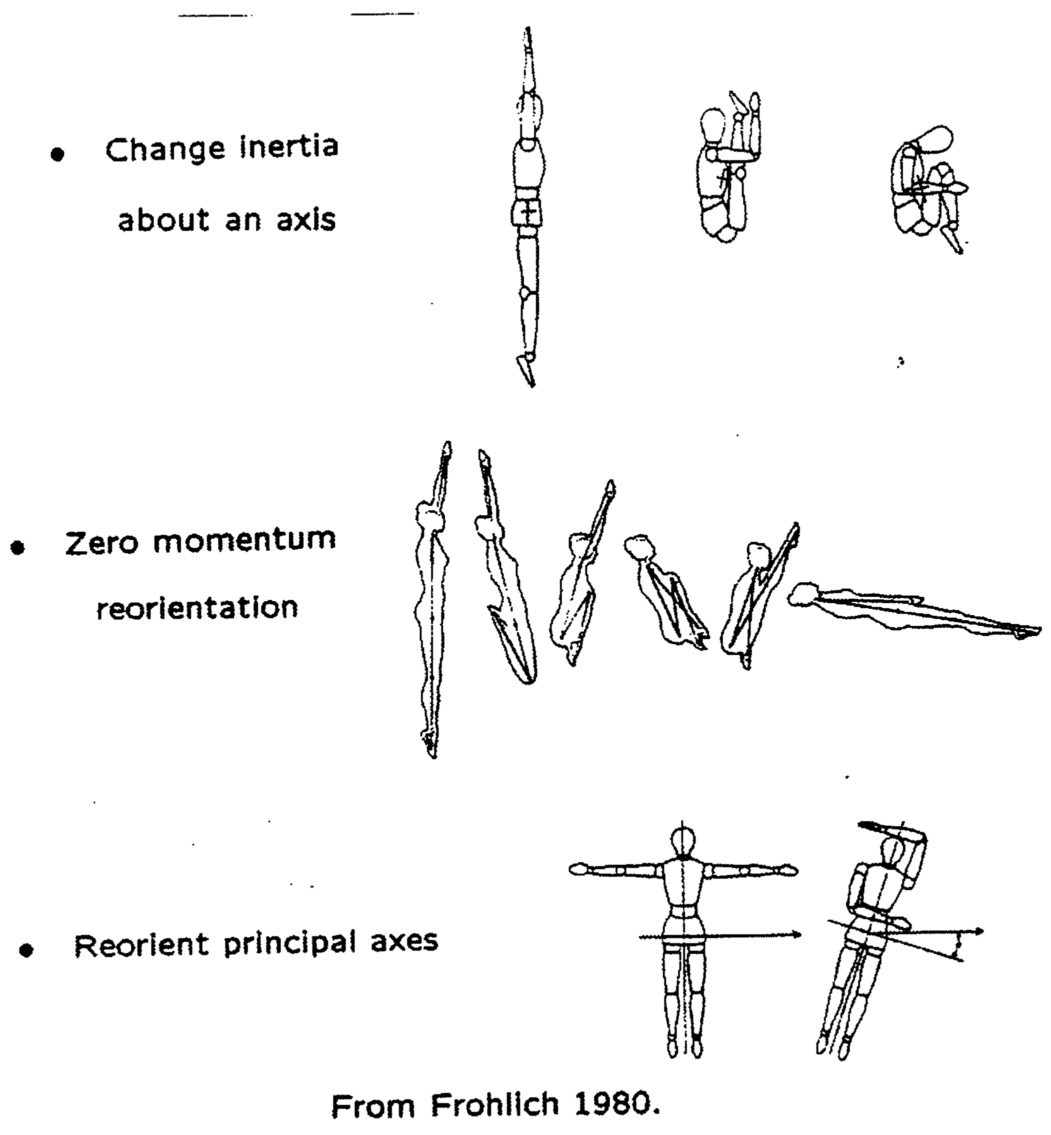

• Change inertia about an axis
• Zero momentum reorientation
• Reorient principal axes
From Frohlich 1980.

Figure 6–1: A limited ability to influence the outcome of a ballistic maneuver arises from the relative movement of the limbs and torso during flight. One way to influence the outcome is to change the moment of inertia about an axis in order to change the rotation rate about that axis. The standard being the ice-skater's spin. A second way to influence the outcome is to use momentum-free rotations (Smith 67; Frohlich 79). This technique allows a structure to be reoriented while maintaining zero angular momentum by performing a sequence of limb movements. A third way is to reconfigure the system so that the principle axes of inertia are reoriented relative to the inertially fixed angular momentum vector. This allows sharing of momentum between principal axes. This procedure is frequently used to introduce or remove twist in somersaults (Batterman 68; Frohlich 79; Yeadon 84). The existence of these mechanisms makes it possible to actively adjust the outcome of an aerial maneuver once it has been initiated.

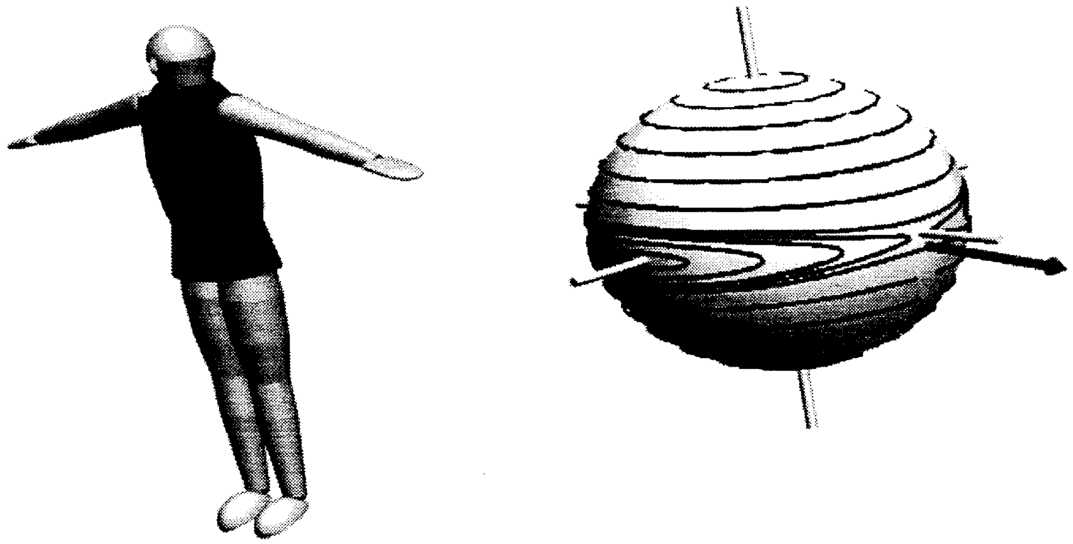


Figure 6–2: A map of rigid body rotation for a human in the layout position. The axes of the sphere and the principal axes of the body remain parallel as the body rotates. The inertially fixed angular momentum vector (black) paints trajectories onto the surface of the sphere as the sphere rotates. Each trajectory on the sphere corresponds to a different rotational energy. The trajectories indicate the tilt and twist angles of the body as it rotates. This map does not include somersault angle and does not show the time dependence of the tilt and twist Euler angles.

positions to maximize the net tilt of the body. However, between these limits, Yeadon claims the timing of the extension is not critical to the final tilt angle.

6.3.1 The Nominal Case

We created a simulation of a human performing a one and one half twisting front somersault. The simulated athlete used the technique described in the previous paragraph to perform the maneuver in a weightless environment (Figure 6–3). The human model has thirteen joint degrees of freedom, including three in each shoulder, one at each elbow, two in each hip, and one in the torso allowing the upper body to twist relative to the lower body. The maneuver was initialized from an upright piked somersault position with the arms held straight out from the side of the body. The maneuver finished in a layout body position with the arms held straight out from the sides.

The control movements used to produce this maneuver were hand programmed so that the model looked natural during the movement and finished with the desired attitude and body configuration. The control movements consisted of a sequence of desired positions of the joints written as functions of time. The purpose of designing this maneuver was to empirically find a sequence of body configurations and joint torques that would produce the desired maneuver under nominal conditions. The nominal somersault rate at the beginning of the maneuver was 12.6 *rad/sec* and the maximum twist rate during the maneuver is 40.2 *rad/sec*.

6.3.2 Off-Nominal Performance, Prescribed Motion

If the simulated performer uses the prescribed set of control movements from a different set of initial conditions than the nominal case then the trajectory of the maneuver will change. How sensitive the maneuver is to variation in the initial conditions is important to the reliability of the maneuver. We tested the sensitivity of this maneuver to initial conditions by running a series of simulations, each starting from a different set of initial tilt (Θ_0) and twist (Ψ_0) angles of the body. We evaluated the reliability of the maneuvers by comparing the landing attitude of the off-nominal simulations to that of the nominal case. (The landing attitude was considered to be the attitude at a fixed time after the start of the maneuver.) The error, e in landing attitude was computed as follows:

$$e^2 = 1/4((\Phi_d - \Phi)^2 + (\Theta_d - \Theta)^2 + (\Psi_d - \Psi)^2 + 1/13 \sum_{i=1}^{njoints} (qd_i - q_i)^2)$$

where Φ, Θ, and Ψ are the body attitude at landing q_i refers to the i^{th} joint position and the subscript d refers to the desired value. This equation for the error emphasizes the body attitude over body configuration. An error of 1.0 is large; it could mean the twist angle or somersault angle was off by 4.0 radians or about 270°.

The results of a series of simulations that varied the initial tilt and twist attitude of the body over a range, $-0.1\,rad \leq \Theta_0 \leq 0.1\,rad$, $-0.1\,rad \leq \Psi_0 \leq 0.1\,rad$ are shown in Figure 6–4.

The nominal maneuver, $\Theta_0 = 0$, $\Psi_0 = 0$, corresponds to the center grid point of this figure. The height of the surface there is zero. Away from the nominal the landing attitude

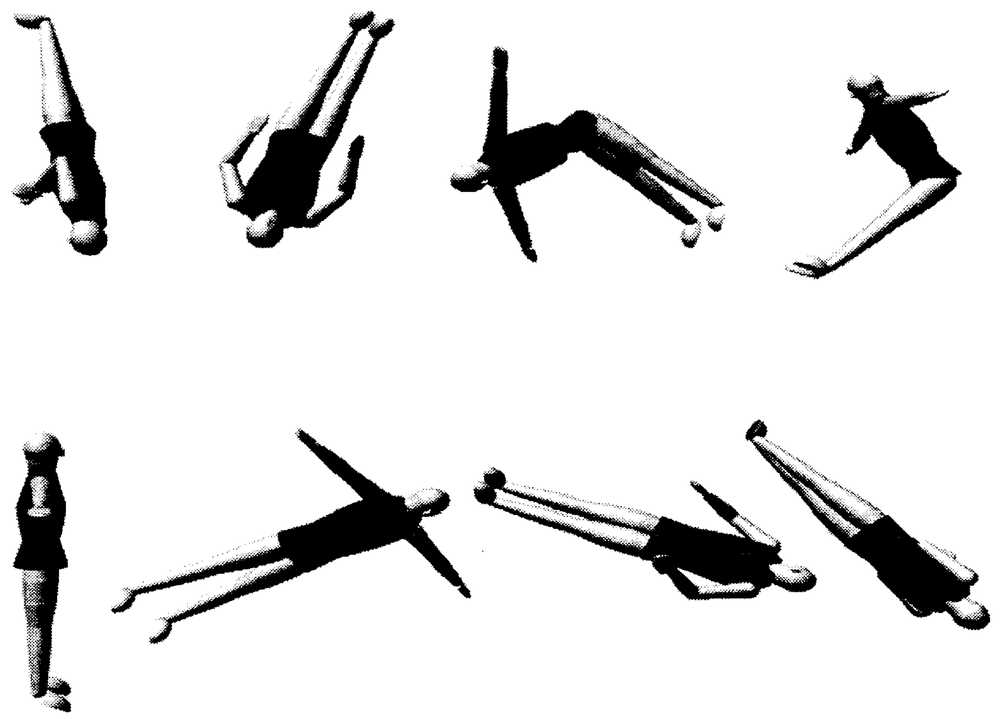


Figure 6–3: Images arranged in **right-to-left, top-to-bottom order** from a dynamic simulation of a 1 1/2 twisting front somersault. The maneuver was initialized from somersault rotation in the piked position. The control movements for this maneuver are a hand programmed sequence of joint angles written as functions of time.

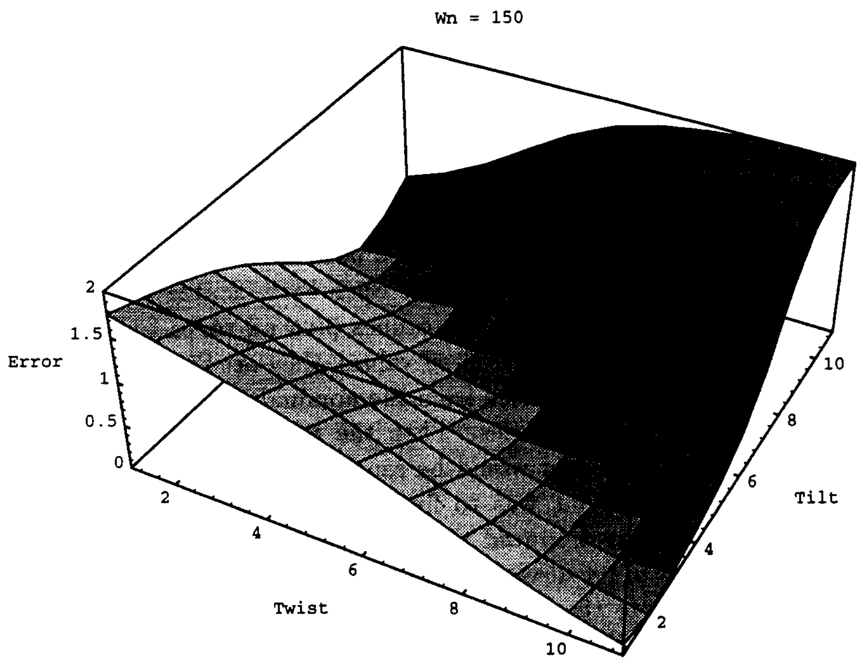


Figure 6–4: This figure shows the accuracy of a 1 1/2 twisting somersault in the presence of off-nominal initial conditions. The simulated performer used a prescribed set of joint angles to perform the maneuver. The two grid axes correspond to initial tilt and twist angles of the body, $-0.1\,rad \leq \Theta_0 \leq 0.1\,rad$, $-0.1\,rad \leq \Psi_0 \leq 0.1\,rad$. The height of the grid indicates the accuracy of the landing attitude of the simulation starting from the corresponding set of initial conditions.

error increases except for a narrow valley of initial tilt and twist attitudes along which the landing error remains small. It appears that a prescribed motion strategy would not produce reliable 1 1/2 twisting somersaults. Is there a simple open loop strategy that can improve this performance?

6.3.3 Off-Nominal Performance, Compliant Motion

It seems unlikely that people could accurately reproduce prescribed motions in a dynamic movement like the 1 1/2 twisting somersault. People use springy muscles and tendons to position their limbs. It seems likely that the change in environmental forces that would accompany a change in maneuver trajectory would mean that the limb movements change even if the athlete *tried to execute the exact same motions.* This idea is the basis for the following set of simulation experiments.

We ran a series of simulations of the 1 1/2 twisting front somersault that used the joint torques from the nominal maneuver as feed forward commands. We also used the prescribed joint motion from the nominal maneuver as the commanded positions for a set of compliant, position plus derivative (pd) servos at the joints. In this open loop strategy, the net joint torques would be a combination of the feed forward torques plus pd servo torques. Since the pd servos use a prescribed set of positions as desired values, they act like passive springs and dampers at the joints. If the maneuver started from the nominal initial conditions then the joints would follow the prescribed trajectories. Then the torques from the pd servos would remain zero and the nominal maneuver would be reproduced.

Our goal in this part of the experiment was to find a set of pd servo gains (spring constants and damping coefficients) that produced the most reliable performances. Rather than search over the gain parameters of pd servos at thirteen joints, we chose to search over one parameter. Therefore, we compute the gains of all joint servos according to a single parameter, the body clamped natural frequency, ω_n. The intent behind this choice is that all body joints have a similar compliance or natural frequency of operation. We will then look for a body natural frequency that produces reliable 1 1/2 twisting somersaults.

To compute the servo gains at a given joint as a function of ω_n, we assume that the body inboard (towards the torso) from the joint is inertially fixed and all out-board joints are immobilized. This way the model simplifies to a single d.o.f. joint between the limb in question and ground. The equations reduce to a simple second order system of equations as follows

$$I\,\ddot{x} + c\,(\dot{x} - \dot{x}_d) + k\,(x - x_d) = 0$$

where I is the apparent inertia of the limb at the joint, c is the damping coefficient, and k is the spring constant of the joint. This simple system can be rewritten in the canonical form

$$\ddot{x} + 2\zeta\omega_n(\dot{x} - \dot{x}_d) + \omega_n^2(x - x_d) = 0$$

therefore $c/I = 2\zeta\omega_n$ and $k/I = \omega_n^2$. We choose $\zeta = 0.7$ to achieve a well damped system response (Ogata). Then choosing ω_n allows us to compute the stiffness and damping constants in a consistent manner.

We systematically varied the value of ω_n between 150 rad/s and 20 rad/s. At each of these values of the body natural frequency we performed a series of simulations starting

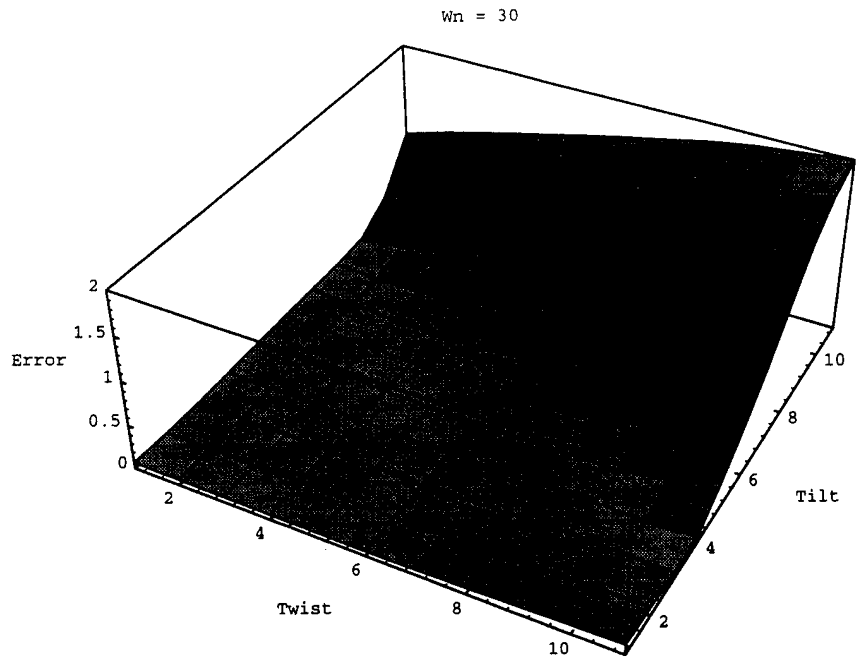


Figure 6–5: This figure shows the accuracy of a 1 1/2 twisting somersault in the presence of off-nominal conditions. The simulated performer used feed forward torques plus passive pd servos at the joints to perform the maneuver. Pd servo gains were chosen according to $\omega_n = 30 rad/s$. The pd servos used the prescribed joint angles from the nominal maneuver as desired values. The two grid axes correspond to initial tilt and twist angles of the body, $-0.1\,rad \leq \Theta_0 \leq 0.1\,rad$, $-0.1\,rad \leq \Psi_0 \leq 0.1\,rad$. The height of the grid indicates the accuracy of the landing attitude of the simulation starting from the corresponding set of initial conditions.

from initial conditions just as in the case of prescribed motion at the joints. We plotted the results in the form of the 3D plot shown in Figure 6–4 for each frequency and subjectively evaluated them. There was a clear choice for the best natural frequency. Figure 6–5 shows the simulation results for the case of $\omega_n = 30\,rad/s$. This plot shows that the reliability of the 1 1/2 twisting somersault performed with the open loop strategy shows marked improvement over the prescribed motion case. Now, nearly half the set of initial conditions results in small final attitude errors. Furthermore, the landing attitude errors were worse for either smaller or larger values of ω_n. These basic results held true for variations in the angular momentum, h, of the maneuver as well $(0.9\,h \leq h \leq 1.1\,h)$.

Figures 6–6 and 6–7 show data from two simulations using the nominal value of angular momentum and with $\Theta_0 = -0.1$, and $\Psi_0 = -0.1$. The data of figure 6–6 shows that when prescribed joint motion is used for an off-nominal maneuver, the body attitude error becomes large at the end of the maneuver. In contrast, the data of figure 6–7 shows that while the joint angles incur some error during the maneuver the body attitude is close to the desired value at the end of the maneuver.

Since the timing of this maneuver will scale with somersault rate it is instructive to show how the body clamped natural frequency compares to the nominal somersault rate,

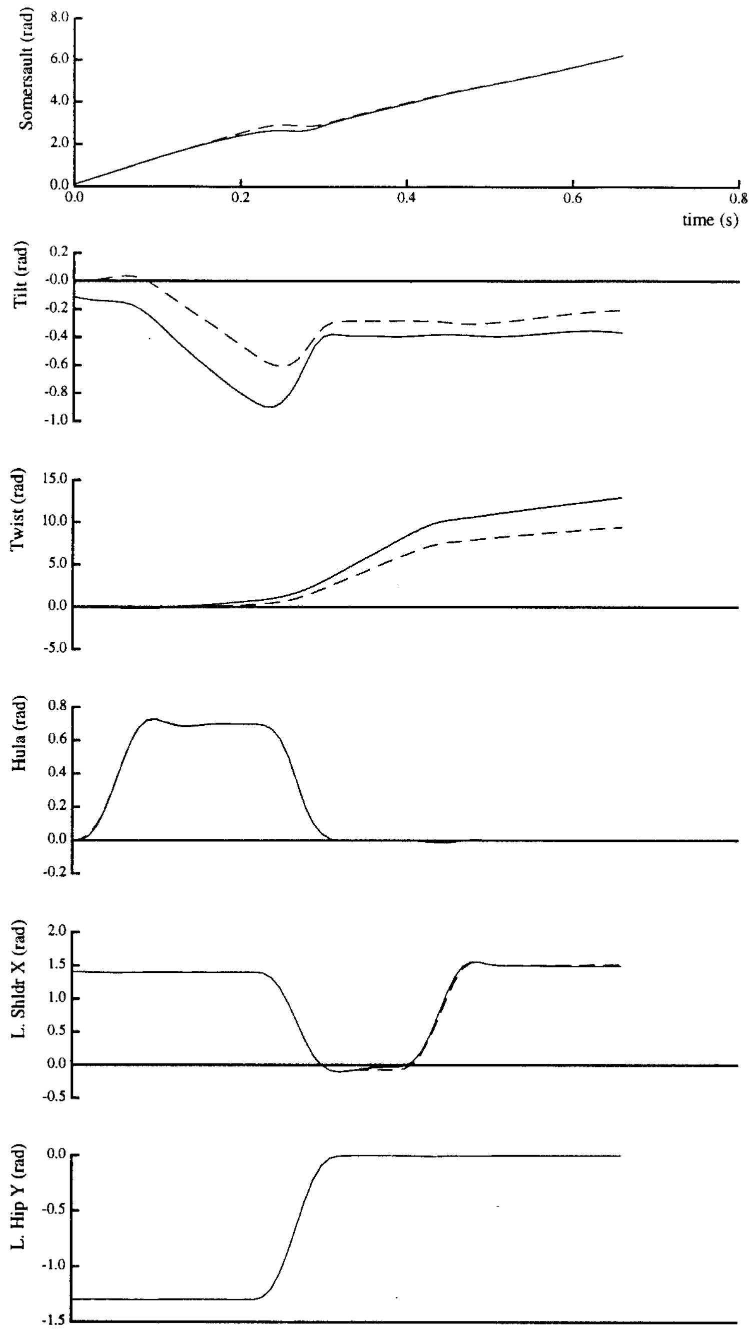


Figure 6–6: Simulation data from a 1 1/2 twisting somersault that used prescribed control motions for initial conditions $\Theta_0 = -0.1$, and $\Psi_0 = -0.1$ are shown with solid lines. The body attitude and joint angles of the nominal maneuver are shown with dashed lines. While the prescribed joint positions are accurate for the off-nominal case (bottom three graphs), the body landing attitude error is large. (The traces stop at the landing time.)

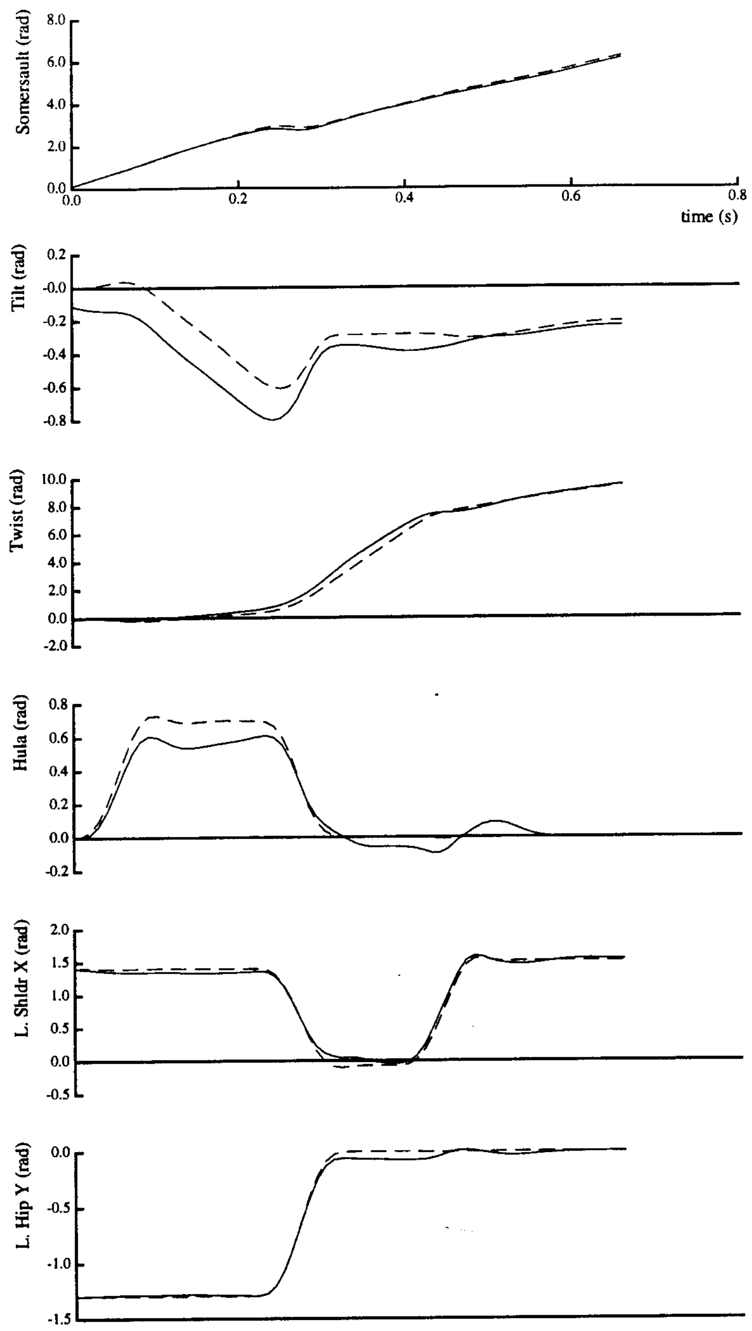


Figure 6–7: Simulation data from a 1 1/2 twisting somersault with $\omega_n = 30, \Theta_0 = -0.1$, and $\Psi_0 = -0.1$ are shown with solid lines. The desired values of the body attitude and joint angles are shown with dashed lines. Joint angles incur significant tracking errors but the landing attitude closely follows that of the nominal maneuver. (The traces stop at the landing time.)

$\frac{\omega_n}{\Phi_0} = \frac{30}{12.6} = 2.38$. It appeared in simulations that the compliance that allowed the arms to open prematurely, thus increasing the body inertia about the twist axis, was important in off-nominal simulations. For this reason, one might consider scaling the body clamped natural frequency by the maximum twist rate of the nominal maneuver, $\frac{\omega_n}{\Psi_{mx}} = \frac{30}{40.2} = 0.746$.

The body clamped natural frequency was chosen to be a simple parameter that described the natural frequency of the whole body. However, since the computation of this parameter assumed that parts of the body were inertially fixed, the actual eigenfrequencies of the system will differ from this body clamped frequency. The actual eigenfrequencies of the body for the initial configuration (pike) and the twisting configuration (wrap) of the Rudi, are shown in Table 6.3.3.

6.4 Twisting Somersault of the 3D Biped

We programmed a simulated 3D Biped robot to use the "torque twist" method to initiate a twisting somersault. To use this technique the simulated 3D Biped robot accelerated its torso in somersault and twist during the stance phase just prior to take-off. These two components of momentum should result in an angular momentum vector that is tilted with respect to the principal axes of the robot.

Figure 6–8 shows the map of rotational motion of the 3D Biped. When compared to the map for the layout somersault in the human (Figure 6–2) we see that the robot requires significantly more tilt of the principal axes (relative to the angular momentum vector) to achieve a twisting somersault. This is partly due to the fact that the robot somersault axis is the major principal axis but also due to the significantly different shape of the regions on the two maps. The qualitative difference in shape of the maps is due to the difference in rigid body inertia ratios of the robot and of the human. To put it simply the human is long and skinny and the robot is short and fat. This difference makes the twist harder to achieve (more tilt required), and it means the maneuver will not look much like that of a human.

In its current configuration, the twisting somersault mode of the 3D Biped would force the robot to lay on its side some time during flight. This body orientation is not what we identify with a twisting somersault in a human. The human's long axis stays closer to the vertical during a twisting somersault. In addition, this inertial orientation of the robot is undesirable from a practical standpoint because this orientation is coincident with gimbal lock in the gyros used to measure body attitude. During laboratory experiments, the physical robot achieved this horizontal position which resulted in damage to the gyroscopes. To correct this situation we changed the robot inertia, and thus its rotational modes, to look more like that of a human.

To change the robot inertia we added weight to increase the major and intermediate principal inertias without increasing the minor principal inertia. We did this by adding weight along the '3' axis of the robot. Figure 6–9 shows the new map of rotation for the 3D Biped with a 4.0 kg weight added 0.8 m above the hips of the robot. This map looks much more like that of a human now and the twisting somersault should bear this resemblance as well. Figure 6–10 shows a sequence of computer graphic images of a simulated 3D Biped

Table 6–1
Rudi Eigenfrequencies

Pike		Wrap	
ω_n *rad/sec*	ζ	ω_n *rad/sec*	ζ
921.0	0	210.0	0
709.2	0	188.3	0
359.4	0	159.2	0
161.2	0	107.7	0
160.6	0	93.1	0
88.0	0	39.7	0.92
69.0	0	39.7	0.92
36.7	0.85	38.9	0.90
36.7	0.85	38.9	0.90
31.3	0.73	30.1	0.70
31.3	0.73	30.1	0.70
31.0	0	30.0	0.70
30.1	0.70	30.0	0.70
30.1	0.70	27.8	0
30.0	0.70	26.8	0.62
30.0	0.70	26.8	0.62
28.3	0	26.7	0
24.7	0	26.5	0.62
24.7	0	26.5	0.62
22.9	0.53	24.9	0.58
22.9	0.53	24.9	0.58
22.7	0	24.8	0.58
22.0	0	24.8	0.58
21.9	0	24.7	0
16.0	0.37	24.1	0
16.0	0.37	23.8	0

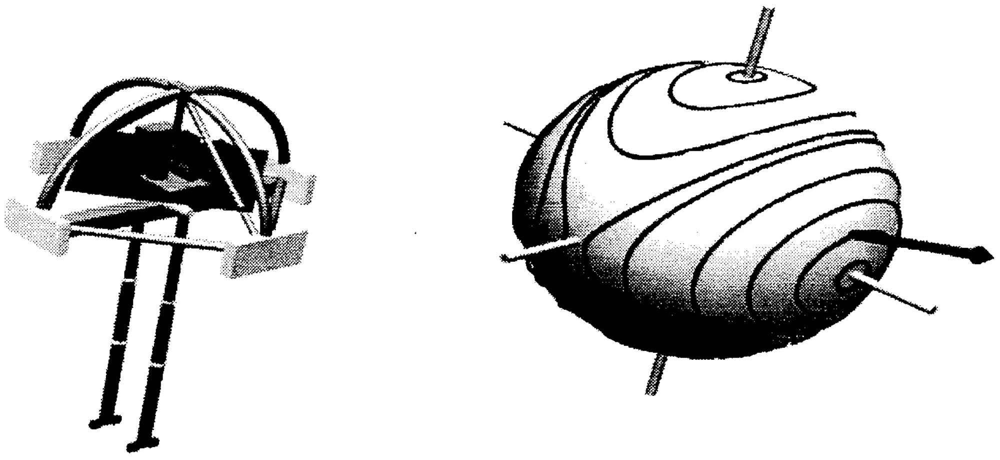


Figure 6–8: The somersault axis of the 3D Biped is coincident with the major principal axis of inertia. With the legs in the fully extended position, as shown here, the minor principal axis is the 'head-to-toe' axis.

with added weight (and strong actuators) performing a front somersault with one half twist. Notice that the simulated robot lands the twisting somersault facing the opposite direction it started from. Data from the simulation is included in Figure 6–11. The simulated running robot regained balance on landing to continue stable dynamic running.

We tried this maneuver with the physical 3D Biped robot in the laboratory. The added weight and inertia were too large for the robot to achieve sufficient flight time or angular momentum to produce the maneuver.

6.5 Summary

In this chapter we presented simulation results of a 1 1/2 twisting front somersault performed by a model gymnast with thirteen joint degrees of freedom. The maneuver is initiated from a piked front somersault. The twisting maneuver results from a sequence of movements of the limbs and torso made during flight. We found that prescribed limb movements could produce reliable 1 1/2 twisting somersaults only for a small set of off-nominal initial conditions. In contrast, a control strategy that used feed forward joint torques in a tuned passive dynamic model of the performer could produce reliable maneuvers for a much larger

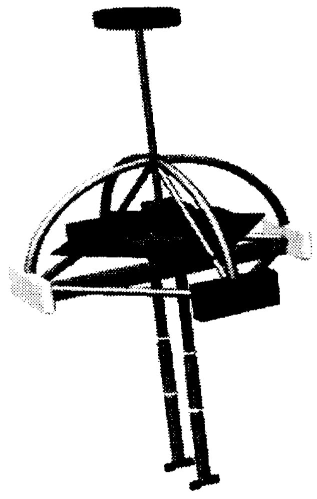
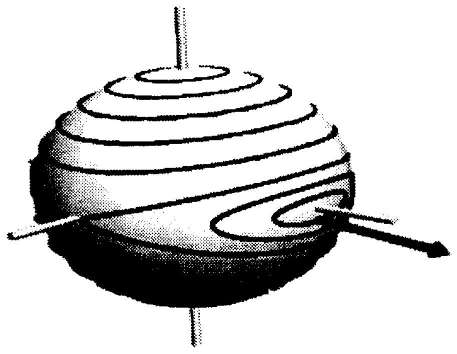



Figure 6–9: Spherical coordinate map of the tilt and twist Euler angle trajectories for the 3D Biped with $4\,kg$ of weight added at a distance of $0.8\,m$ above the hip.

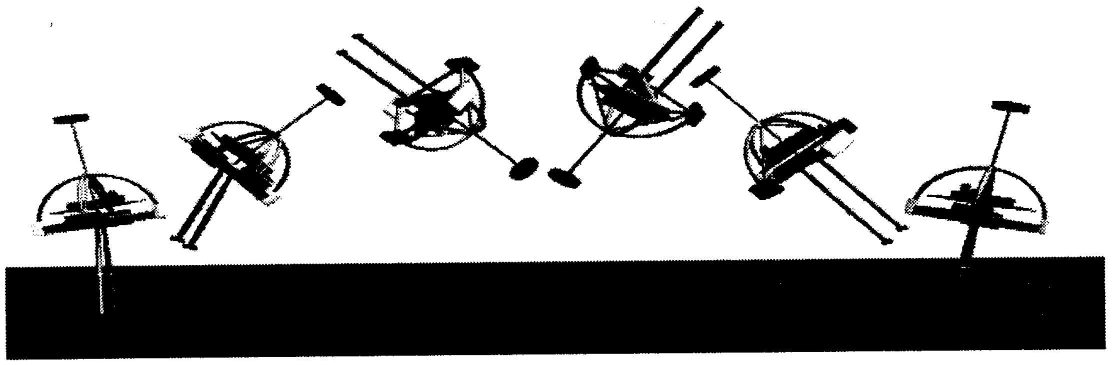


Figure 6–10: Sequence of images of the simulated 3D Biped somersault with twist.

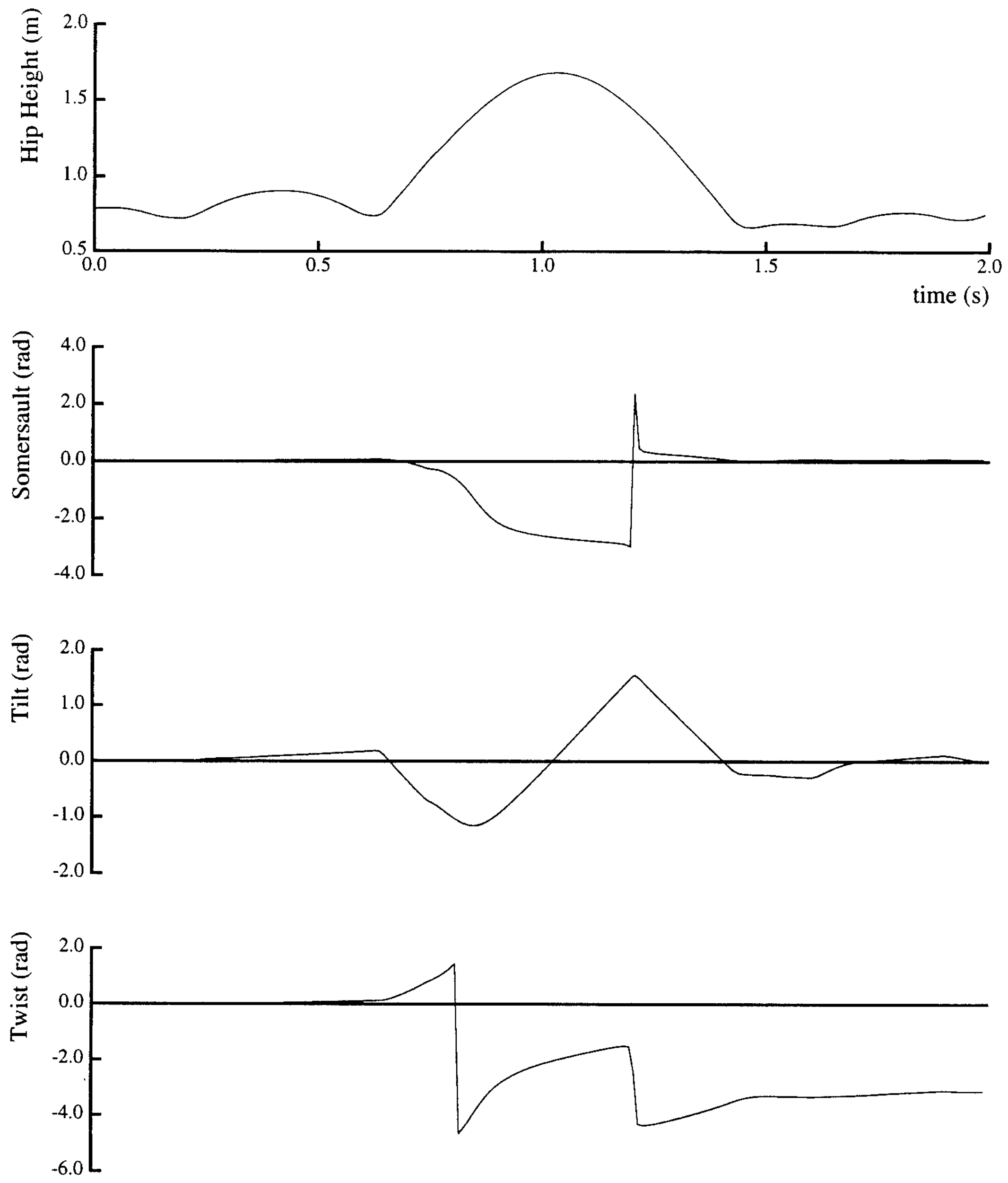


Figure 6–11: Hopping height, somersault, tilt and twist Euler angles of the 3D Biped during a simulated front somersault with 1/2 twist. The simulated robot passes through an Euler angle singularity at approximately 1.2 sec. causing discontinuties in the data. This singularity did not affect the dynamic simulation however as a different set of attitude parameters were used.

set of off-nominal initial conditions.

We do not yet know the significance of this result. There are many degrees of freedom of this model. Too many to easily distinguish between important and unimportant effects. Simplification of this model may reveal salient features that can be analytically confirmed. These simulation results from the rudi are potentially interesting because it suggests that a passive dynamic approach to twisting somersaults may be capable of producing reliable maneuvers as was the case with the layout somersault. It is interesting that many of the eigenfrequencies of the most reliable, tuned compliant system for the rudi were in the neighborhood of the maximum twist rate of the maneuver. This appears to be analogous to the ideal choice of arm-body oscillation in the passive layout somersault. In this case the best shoulder spring resulted in an eigenfrequency of the arm-body oscillation that was equal to the somersault rate.

In this chapter, we also described simulation experiments with 3D Biped twisting somersaults. The distribution of mass in the 3D Biped robot makes a twisting somersault particularly difficult. In order to enter a rotational mode that could produce the twist angles desired, the robot had to assume body attitudes unlike those associated with a human somersault with twist. This extreme body attitude could lead to mechanical difficulties with inertial instruments on the physical robot. We added weight to the robot so its natural rotational modes were more like those of a human. A simulated 3D Biped robot was able to perform a 1/2 twisting front somersault. The simulated robot landed the maneuver and continued running stably afterwards. The added weight made twisting somersaults of the physical 3D Biped robot impossible due at least in part to insufficient actuator power.

References

Batterman, C., 1968. *The Techniques of Springboard Diving.* The MIT Press, Cambridge, Ma.

Frohlich, C. 1979. Do Springboard Divers Violate Angular Momentum Conservation? *American Journal of Physics* 47(7):583–592.

Frohlich, C. 1980. The Physics of Somersaulting and Twisting. *Scientific American* 242(3):154–164.

Smith, P. G. and Kane, T. R., 1967. *The Reorientation of a Human Being in Free Fall.* Stanford University, Division of Engineering Mechanics, Technical Report No. 171.

Yeadon, M. R., 1984. *The Mechanics of Twisting Somersaults* Ph.D Thesis, University of Loughborough, Loughborough, United Kingdom.

Ogata, K., 1990. *Modern Control Engineering.* Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

Chapter 7

Automatically Tuning Control for Legged Creatures

Robert Ringrose

7.1 Introduction

Within the domain of actively balanced legged locomotion, it is necessary to tune control systems to reflect physical alterations of the robot. I have designed and implemented a tuning algorithm which will tune an existing control system to control a different robot, or to exhibit different behavior. This algorithm has successfully tuned the control system of a planar quadruped simulation to accommodate a reduction in leg length by a factor of two, an increase in body mass by a factor of three, and changes in the commanded speed while trotting. [1]

Any control system has some set of control parameters, numbers which determine how it performs. For example, a parameter might control how rapidly it tries to accelerate to a desired speed or how much energy it injects at each step. For a complicated system, the appropriate values for these control parameters are not obvious. There are several advantages to having a computer search for a set of parameters which minimizes an evaluation function rather than having a human tune the control system directly. Automatically tuning the control system does not require that a human with experience tuning invest a large amount of time. One can also specify the desired behavior without considering the interactions between any specific input parameters to the control system. Additionally, it is easier for a computer to optimize for something which is not obvious to a human, such as minimal energy consumption. Properly specifying the desired behavior is not a trivial task, but it seems easier than manually tuning the control system.

[1]This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

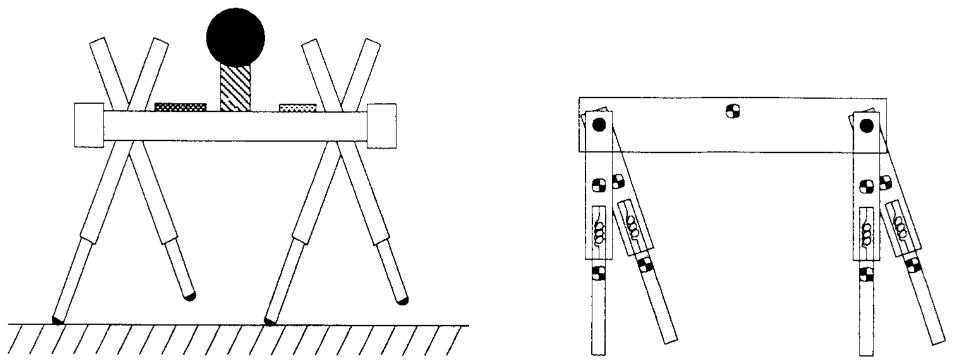


Figure 7–1: Illustration of the simulated quadruped and the associated model. The actuators at the hips are implemented as torque sources. The leg actuator is implemented as a spring with controllable rest length and different constants in compression and extension. The simulation is a planar rigid-body model.

Other work on self-tuning controllers (Helferty, Collins, & Kam 1988), which frequently used searching techniques such as spacetime constraints (Witkin & Kass 1988) or genetic algorithms (Pearse, Arkin, & Ram 1992), has addressed similar problems. Tuning controllers for dynamically balanced legged systems is particularly challenging because there is typically only a small "sweet spot" near the global minimum where one can effectively evaluate the robot's behavior. A parameter set outside this sweet spot will generally make the robot fall over or not take any steps, while a parameter set inside the sweet spot will make the robot run well enough that its performance can be evaluated objectively. The vast majority of the possible parameter sets for dynamically balanced legged locomotion lie outside of the sweet spot. As a consequence, general search methods like genetic algorithms and simulated annealing will take a long time to find any working solution, and searches which follow a local slope will only find a useful minimum if they start in the sweet spot. A frequently used method for getting around this challenging search space is to simplify the problem so as to drastically increase the sweet spot's size (for example, one can add constraints to the model that prevent the robot from falling over). Instead of modifying the search space to suit my algorithm, however, I have attempted to adjust my algorithm to fit the search space.

In the event that there is a parameter set which is in the sweet spot, a simple gradient descent search can find a local minimum. Additionally, most of the time a small change in the robot's configuration will result in only a small change in the sweet spot. The tuning algorithm presented here uses this characteristic to break the search for a new set of parameters into a series of smaller searches for which minima are easier to find. For example, assume the control system for a quadruped running simulation has been tuned to run with legs of a particular length. To find a set of control parameters for a quadruped running with legs half as long, the tuning algorithm gradually reduces the leg length and optimizes at several leg lengths between full and half length. As the leg length changes, the location of the sweet spot will change. For small changes the sweet spot's motion will usually be slight enough that the control parameters for the unchanged leg length will still be within the new sweet spot. The tuning process may fail if gradient descent cannot find a local minimum (the sweet spot might not be continuous), if the sweet spot changes dramatically with a small alteration in the robot, or if there is no way for the given control system to control the robot.

To find appropriate values for the control parameters, the tuning algorithm described in this paper starts with an existing control system, simulation, and parameter set. It finds out how far it can modify the simulation and still get acceptable behavior, makes that modification, and then uses a gradient descent search to improve the performance of the modified simulation. This process of modifying the simulation and re-tuning the control system is repeated until you have the desired final simulation.

7.1.1 The Simulation

I have used a planar quadruped simulation to test the tuning algorithm. It retains enough complexity to illustrate most of the problems that come up, but is simple to visualize and easy to explain. The simulation is based on a physical robot described by Raibert (Raibert, Chepponis, & Brown 1986)(Raibert 1990).

The simulation is a rigid body simulation, the dynamics of which are generated using a commercial dynamic modeling program (Rosenthal & Sherman 1986)(Ringrose 1992a). Simulation creation is automated so that it is possible to change and re-create any simulation as part of the tuning process. The planar quadruped which I used is illustrated in figure 7–1.

The simulated robot is controlled by a planar variation of the finite state controller for the Raibert trotting quadruped (Raibert *et al.* 1992). The control system uses measurements which could be sensed or calculated on a physical robot, such as position, velocity, actuator lengths, and ground contact. The control system's behavior can be modified through 20 parameters, including maximum acceleration, desired speed, spring constants, and desired leg length during different running phases. Some previous investigations into robotic running are described in references (Hodgins & Raibert 1989)(Hodgins & Raibert 1991)(Playter & Raibert 1992)(Raibert *et al.* 1992). Further details of the controller and model are available in (Ringrose 1992b).

7.2 Searching for Solutions

In order to search for an appropriate set of control parameters, you need a way to compare the behavior generated by different sets of control parameters. I use the results from an evaluation function which simulates the creature and returns an objective measure of the creature's performance. Most evaluation functions for dynamically stable running motions result in a search space whose structure makes global searching algorithms ineffective. However, once you have a reasonably good solution, a gradient descent search (Press *et al.* 1988) modified to take into account local minima (Ringrose 1992b) will frequently be able to find a better solution.

Most evaluation functions for legged locomotion have a nearly pathological search space because if the parameters are out of a small sweet spot around the good solutions the simulated creature fails catastrophically, usually by falling over or not taking any steps.

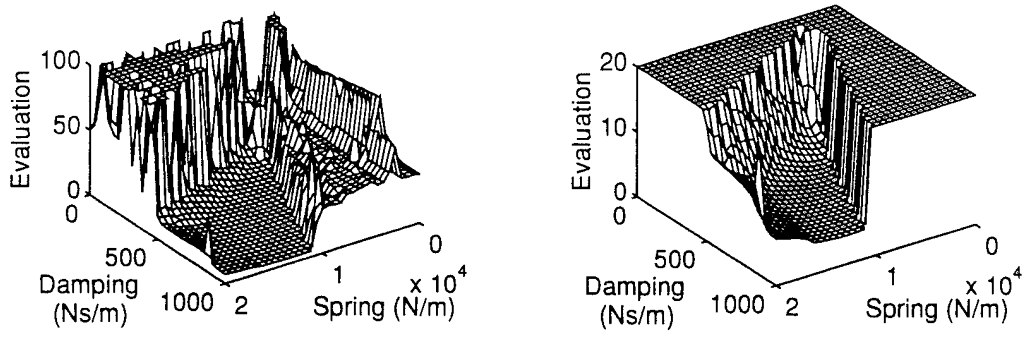


Figure 7–2: Graph of evaluation results over changes in leg spring and damping constants for the planar quadruped, using the evaluation function described in the text. The noisy, high-valued regions are outside the sweet spot. The right graph is the same as the left, with a lower maximum value imposed to emphasize the structure of the sweet spot.

When such a failure occurs, a meaningful evaluation of performance is difficult since the causes of the simulation's failure to trot are difficult to determine. The creature could fall over if it stubs its toe, the leg springs are not strong enough, the swing legs do not come forward fast enough to catch the robot, or some other reason. It is not difficult to make an evaluation function which recognizes when it is out of the sweet spot, but it is difficult to ensure that when outside the sweet spot the gradient of the evaluation function leads towards the sweet spot.

Because of the inherent difficulty evaluating a catastrophic failure, most evaluation functions only have a useful section near the global minimum and the rest is noise. Evaluation functions usually have more dimensions than can be readily visualized, but cross sections can give an idea of the search space's general structure. A two dimensional cross-section of the evaluation function is somewhat like a smooth canyon (the sweet spot) in noise, with the noise being uniformly higher-valued than the sweet spot. There are many parameter sets that do not generate information except to indicate that the creature failed (in the noisy section) and there is a smaller number of parameter sets where the creature may actually run well. Figure 7–2 illustrates the general shape of evaluation functions, using data from the simulated quadruped.

7.2.1 Evaluating Performance

In order to evaluate the performance of a set of control parameters, I created an evaluation function which reflects the fact that a control algorithm for a trotting quadruped needs to do more than propel the quadruped forward. Raibert's experimentation in quadruped control suggests that it should (Raibert 1990):

- control the forward velocity.
- regulate the body attitude.
- put reasonable constraints on the forces and torques applied.

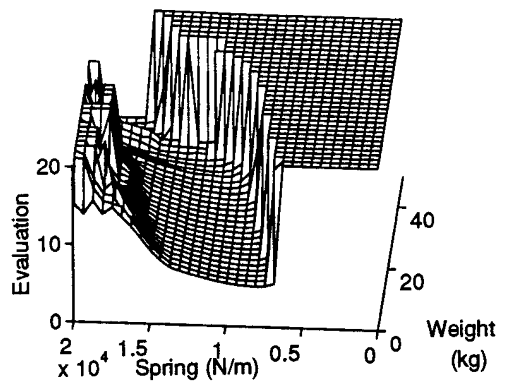


Figure 7–3: Leg spring constant in compression as weight on the quadruped increases. Note how by staying in the minimum as weight is added to the quadruped's trunk it is possible to find a spring constant for 50 kg which is within the sweet spot. Parameters other than the leg spring constant in compression are optimized for the appropriate weight.

- limit the vertical motion of the body.
- keep the running cycle stable.

The evaluation function I used is the integration of the departures from these goals over the course of seven simulated seconds (Ringrose 1992b). Seven seconds, the length of time over which the behavioral error is integrated, is several times the length of the step cycle, allowing transients to die out. This evaluation function does not guarantee that the running cycle is stable beyond the seven seconds of running actually simulated. In practice, however, if the simulation successfully trots for that length of time it is stable enough that it is unlikely to fail later.

7.3 Getting a Close Solution

A goal of this work is to be able to automatically tune the control system when there are large changes in the physical characteristics of the creature. When the simulation changes by a small amount, a good parameter set may no longer be locally optimal, but it may still be within the sweet spot. Figure 7–3 shows a cross section of a sample evaluation function and how it changes as the simulation is altered. If the simulation's change is small enough that the new location of the sweet spot still overlaps the old set of control parameters, those original control parameters can be used with a local search such as gradient descent to find a new set of acceptable control parameters. Generally, if there is a large physical change, the control parameters which were originally acceptable give poor results because the sweet spot moves too far. However, by splitting the large change into a series of smaller changes one can follow the motion of the sweet spot as the simulation changes.

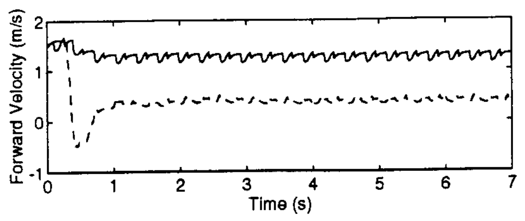
In order to have the tuning process work efficiently, it is desirable to take large steps when possible. I use a divide-and-conquer algorithm which splits large changes in half if necessary and recursively solves each half. Set up the simulation with a fraction f which goes from 0 to 1, where 0 is the original configuration and 1 is the final configuration. Let $F_a(P)$ represent running the simulation with the fraction $f = a$ and the parameter set P, applying the evaluation function to the run, and returning the result. Let a and b be numbers between 0 and 1. Let P_a be a parameter set such that $F_a(P_a)$ is "acceptable" (less than a user-defined constant). The algorithm used to find some P_b, a parameter set such that $F_b(P_b)$ is acceptable, is:

- If $F_b(P_a)$ is "good enough" (less than a constant supplied by the user), $P_b = P_a$.
- Otherwise, if $F_b(P_a)$ is "acceptable", P_b is the set of parameters arrived at by a gradient descent search with P_a as a starting point and using the model with fraction b until the result $F_b(P_b)$ is "optimized" (less than another user-defined constant).
- Let $c = (a + b)/2$.
- Recursively use this algorithm to find P_c, a parameter set such that $F_c(P_c)$ is acceptable, from P_a, a, and c.
- Recursively use this algorithm to find P_b from P_c, c, and b.

Setting the constants "optimized", "good enough" and "acceptable" requires some care. If the level at which the simulation is considered "optimized" is too low, the gradient descent search will take a long time finding a good parameter set (recall that low evaluation results correspond to desired behavior). On the other hand, if "optimized" is too high, the gradient descent search could return a parameter set which is close to the edge of the sweet spot, reducing efficiency. The constant "good enough" corresponds to the point at which the control system performs well and does not need further optimization. This means that the conditions for beginning optimization are less rigorous than the conditions for ending optimization, so that each time the gradient descent search is used it is required to perform a non-trivial amount of work. Finally, "acceptable" corresponds to the edge of the sweet spot. If the result of the evaluation is too high, it is considered to be in the area where the evaluation function is essentially useless.

Note that the fractions at which the gradient descent search is used increase monotonically from 0, and the only time the parameter set P is modified is when the gradient descent search is used.

There are restrictions to this procedure. It must be possible to gradually change the parameter set and configuration from the initial parameter set and configuration to the final one, without leaving the sweet spot. Also, the control system must be able to control the new configuration. For example, the person designing the control system might neglect one of the moments of inertia, and in a new configuration that moment could be extremely important for stability. Since this tuning method will not alter the control system, it will not be able to address this type of problem. Additionally, the behavior for the initial configuration must be similar to the behavior required in the final configuration. If there is a drastic change in strategy involved, such as a change in gait, it may not be able to find suitable parameters. Finally, if the global minimum is on the edge of the sweet spot, this type of tuning will be inefficient, although still functional.



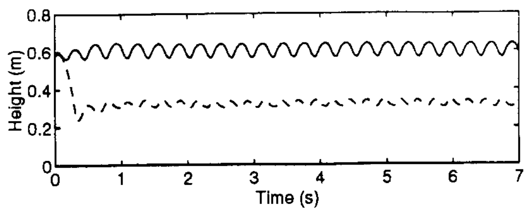


Figure 7–4: Height above ground and forward velocity over time, included to illustrate that the trotting achieved is stable before and after tuning. Solid lines indicate original leg length and original parameter set and dashed lines indicate half leg length and the corresponding tuned parameter set. Note that the initial conditions remain the same, so the quadruped with shorter legs actually falls to the ground, stops, and begins trotting.

Parameter	*Initial*	*Final*	*Units*
Maximum acceleration	0.31284	0.36085	m
Leg spring constant, compression	7803.57	8731.33	N/m
Leg damping coefficient, compression	471.971	630.748	Ns/m
Leg spring constant, extension	21183.7	19637.5	N/m
Leg damping coefficient, extension	462.931	1071.51	Ns/m
Stance leg length	0.62217	0.35498	m
Stance leg length increase	0.08635	0.07645	m
Swing leg length	0.45252	0.28027	m
Increase in swing hip torque with speed	-0.0042	-0.0058	N/s
Acceleration rate	0.65024	0.05992	s
Hip servo	282.093	182.354	Nm
Hip damping	21.3930	17.0266	Nms
Desired forward speed	1.50000	0.34164	m/s

Table 7–1: Parameters modified while decreasing the quadruped leg length.

7.4 Results

In order to evaluate the algorithm described in the previous chapters, I applied it to adjusting the control of the simulated quadruped running machine shown in figure 7–1. I used a planar quadruped simulation because it allowed the solution of interesting problems with a reasonable amount of processing time. I tested the algorithm for variations in leg length, body weight, and desired speed; it performed well on all of these. Due to space considerations, only the data on variations in leg length is included here.

The tuning algorithm was used to reduce the leg length to half its initial value, while maintaining the trotting gait. Interestingly, the problem of getting the quadruped to trot with half-length legs was more difficult than expected because of the very short travel allowed for the leg actuators.

The initial configuration was the quadruped simulation and control system mentioned

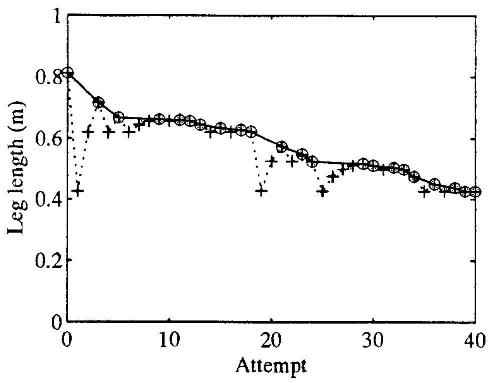


Figure 7–5: Leg lengths where the tuner tried to optimize (dotted line) and leg lengths where it succeeded (solid line).

earlier. The final configuration was the same quadruped simulation and control system, with legs half as long and leg moments of inertia and masses scaled as cylinders. The tuning experiment took three and a half days on an IBM RS/6000 model 550 to find the result listed in table 7–1. Figure 7–4 illustrates the stable running elicited by the original and final parameter sets when used with their respective simulations. Figure 7–5 shows the leg lengths at which the optimization occurred. Note that that the original control parameters for full length legs will not work on the final quadruped.

7.5 Conclusions

The tuning algorithm presented here has successfully solved several optimization problems relating to dynamically stable legged locomotion. All of these problems involve a planar trotting quadruped simulation and vary the amount of weight on the body, the leg length, or how closely it tracks a desired speed. I have also used this tuning algorithm to increase the amount of weight on the quadruped's feet and to increase the running speed of a kangaroo-like robot.

Many searching methods fail when dealing with dynamically balanced legged locomotion because easily created evaluation functions tend to result in a search space which is only tractable near a solution. The tuning algorithm presented here succeeds because it makes the simplifying assumption that the tractable area moves slowly as the simulation is altered. This simplification allows the use of a fairly simple search within the tractable area.

Because of the assumptions behind it, there are limitations to the usefulness of this tuning algorithm. It must be possible to gradually change the parameter set and configuration from the initial parameter set and configuration to the final one, without leaving the sweet spot. If there is a drastic change in strategy involved, such as a change in gait, it may not be possible to gradually change the control parameters. Also, the control system must be capable of controlling the new configuration, as the tuning algorithm will not alter the structure of the control system. Finally, if the global minimum is on the edge of the sweet spot, this tuning algorithm will be inefficient. Some of these limitations can be overcome by carefully constructing the evaluation function.

Even with its limitations, this tuning method will prove useful for modifying simulations and eliciting desired behaviors. I believe that tuning methods such as the one presented here will turn the art of tuning a simulation into the art of constructing an evaluation function—still an art, but one which is a little easier.

References

Helferty, J. J.; Collins, J. B.; and Kam, M. 1988. A learning strategy for the control of a mobile robot that hops and runs. In *Proceedings of the 1988 International Association of Science and Technology for Development.* IASTED.

Hodgins, J. K., and Raibert, M. H. 1989. Biped gymnastics. *International Journal of Robotics Research.*

Hodgins, J. K., and Raibert, M. H. 1991. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation* 7(3).

Pearse, M.; Arkin, R.; and Ram, A. 1992. The learning of reactive control parameters through genetic algorithms. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* 1:130–137.

Playter, R. R., and Raibert, M. H. 1992. Control of a biped somersault in 3d. In *IFToMM-jc International Symposium on Theory of Machines and Mechanisms.*

Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. 1988. *Numerical Recipes in C.* Cambridge University Press. chapter 10, 290–352.

Raibert, M. H.; Hodgins, J. K.; Playter, R. R.; and Ringrose, R. P. 1992. Animation of maneuvers: Jumps, somersaults, and gait transitions. In *Imagina.*

Raibert, M. H.; Chepponis, M.; and Brown, Jr., B. 1986. Running on four legs as though they were one. *IEEE Journal of Robotics and Automation* RA-2(2).

Raibert, M. H. 1990. Trotting, pacing and bounding by a quadruped robot. *Journal of Biomechanics* 23.

Ringrose, R. 1992a. The creature library. Unpublished reference guide to a C library used to create physically realistic simulations.

Ringrose, R. 1992b. Simulated creatures: Adapting control for variations in model or desired behavior. Master's thesis, Massachusetts Institute of Technology.

Rosenthal, D. E., and Sherman, M. A. 1986. High performance multibody simulations via symbolic equation manipulation and kane's method. *Journal of Astronautical Sciences* 34(3):223–239.

Winston, P. H., and Shellard, S. A., eds. 1990. *Artificial Intelligence at MIT: Expanding Frontiers*, volume 2. Cambridge, MA: MIT Press. 149–179.

Witkin, A., and Kass, M. 1988. Spacetime constraints. In *Computer Graphics*, 159–168.