A COCKROACH INSPIRED ROBOT WITH ARTIFICIAL MUSCLES

by

DANIEL A. KINGSLEY

Submitted in partial fulfillment of the requirements

For the degree of Doctor of Philosophy

Dissertation Adviser: Dr. Roger Quinn

Department of Mechanical and Aerospace Engineering

CASE WESTERN RESERVE UNIVERSITY

January, 2005

There's no one to take my blame
if they wanted to
There's nothing to keep me sane
and it's all the same to you
There's nowhere to set my aim
so I'm everywhere
Never come near me again
do you really think I need you

I'll never be open again, I could never be open again.
I'll never be open again, I could never be open again.

And I'll smile and I'll learn to pretend
And I'll never be open again
And I'll have no more dreams to defend
And I'll never be open again

Dream Theater
"Space-Dye Vest" (Awake, 1994)

Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank God for the gifts that I have been given that have allowed me to travel so far in life. It is far more than I have deserved.

Thank you to my advisor Dr. Roger Quinn for providing me with the opportunity to explore an avenue of research that I have found exciting, fulfilling, and rewarding.

Thank you to my committee, Dr. Roy Ritzmann, Dr. Thomas Kicher, and Dr. Malcolm Cooke as well as mechanical engineering department chair Joe Prahl; more than professors, they have been teachers who have helped me to better understand the world around me.

To my parents, thank you for your support and understanding through these years of research.

Richard Bachmann, Jong-ung Choi, Gabe Nelson and Brandon Rutter all provided critical support to my work, and I could never have completed it without the help that each provided.

Chris Assad and Mitra Hartmann at NASA JPL both provided a stimulating environment, and a chance to take a break from the research grind at Case for some research at JPL.

My time at Case was financially supported by the Case Prime, NASA GSRP and OAI fellowship programs.

A Cockroach Inspired Robot With Artificial Muscles

Abstract

by

Daniel Adam Kingsley

A robot was developed for the purpose of research into legged locomotion. Legged robots are complex mechanisms, and their development can be greatly aided by insights into the mechanisms—both physical and control—by which animals locomote. This text presents the design methodology used for the development of the fifth such biologically inspired robot at Case: Robot V. The robot is based on the deathhead cockroach, *Blaberus discoidalis*. It has twenty-four degrees of freedom (DOF) distributed amongst task oriented legs; five DOF in the front legs, four DOF in the middle, and three DOF in the rear. Previous research has shown that this joint configuration can closely approximate that of the cockroach, while not being too complex from and engineering standpoint. Actuation is by braided pneumatic actuators, and coupled with a valve system that allows air to be trapped within the actuators, numerous beneficial muscle-like properties were produced. Most importantly, this system enabled rapid response to perturbation, much like "preflexes" exhibited by animals, affording the system a level of passive stability. The robot is controlled by a hierarchical control system, and the operation of the interleg coordination mechanism, a variant of the distributed network of the responsible for stick insect interleg coordination proposed by Cruse, is discussed in detail. The robot

has demonstrated open loop passive stability and locomotion. With the addition of joint angle and actuator pressure sensors, the robot has been able to perform "air walking" motions while suspended from a gantry as well as walking on a treadmill while its weight is partially supported. Closed loop walking has been accomplished, but further development of its locomotion controllers is called for.

Chapter I: Introduction

Introduction

1.1 Background

For decades, researchers have sought to design a fully autonomous, legged walking machine capable of operating in complex natural environments. Development of such machines has been limited by actuators, power sources and control schemes that cannot hope to compete with even some of the "simplest" systems found in the natural world; in contrast, natural organisms have developed a wide range of means by which to locomote through any terrain imaginable. Faced with the challenge of reproducing such capabilities, it is not surprising that more and more researchers are beginning to look toward biological mechanisms for inspiration.

Biology provides a wealth of concepts to assist robot design. There are millions of species of animals that have evolved efficient solutions to locomotion and locomotion control. Insects in particular are well known not only for their speed and agility but also for their ability to traverse some of the most difficult terrains imaginable; insects can be found navigating sparse or rocky ground, climbing vertical surfaces, or even walking upside down. Furthermore, insects almost instantly respond to injury or removal of legs by altering stance and stepping pattern to maintain efficient locomotion with a reduced number of legs (Delcomyn, 1998). Given the ultimate goal of autonomy, this ability to reconfigure locomotion strategies will be crucial to the robustness of autonomous robots (Raibert and Hodgins, 1992).

The recognized benefits of legged locomotion have created intense interest in the features and characteristics of the many successful examples we see in nature,

specifically insects. Walking insects are exceptionally adaptable in the way they move and use their legs. They can walk on irregular surfaces like leaves and branches almost as well as they can on flat, level ground. They can walk forward and backward, and in some cases even sideways or upside down (cockroaches, for example, perform this through the use of a gripping foot called a tarsus) (Zill and Seyfarth 1996). Insects can also walk after injury or damage to their legs or even after suffering the complete loss of one or more legs.

There are of course other mechanisms capable of producing locomotion, most notably wheels and caterpillar treads. While these devices are admittedly much easier to design and implement, they carry with them a set of disadvantages that greatly inhibits their use in military or exploratory applications. Primary amongst these limitations is the simple fact that wheels, and to a lesser extent treads, are not capable of traversing terrain nearly as complex as that which a legged vehicle is capable of maneuvering over (Raibert, and Hodgins, 1993). Even wheeled and tracked vehicles designed specifically for harsh terrains, such as the 2003 Mars Explorers "Spirit" and "Opportunity" constructed at NASA JPL (Figure 1.1) cannot maneuver over an obstacle significantly shorter than the vehicle itself (for example, operators spent days

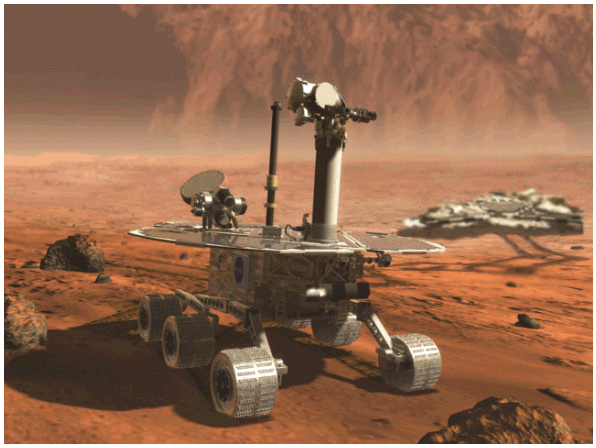


Figure 1.1: NASA's 2003 Mars Explorer

determining the necessary path for Spirit to maneuver around a deflated air bag after landing); a legged vehicle on the other hand could be expected to climb an obstacle up to twice its own height, much like a cockroach can (Watson et al. 2002a, Watson et al. 2002b). This limitation on mobility alone means that in any environment without fairly flat, continuous terrain, a walking vehicle is far preferential to a wheeled or tracked one. Legged vehicles are also inherently more robust than those dependent on wheels or tracks. The loss of a single leg on a hexapod will result in only minimal loss in maneuverability; on a wheeled vehicle a damaged wheel could spell the end of mobility, and a damaged caterpillar tread almost always results in catastrophic failure. Finally, legged locomotion is far more capable of navigating an intermittent substrate—such as a slatted surface—than wheeled locomotion. While wheels could conceivably become trapped on a surface with large spaces between surfaces, such as a girder network made up of thin beams, legs could be placed only on solid substrate, allowing navigation over even the sparsest of terrains (Espenschied et al. 1996).

1.2 Approaches to Biologically Inspired Robots

Given the preceding argument for the use of legged locomotion in certain environments, one is left with the daunting task of actually designing a walking robot. While such a task is enormous to say the least, nature has provided us—literally—with a world full of templates. Animals can be found that are capable of navigation over almost any surface, and it is from these natural solutions to locomotion problems that engineers are more and more often seeking inspiration.

There are two different methods through which biology has been used as an inspiration for robotic systems (Ritzmann et al, 2000). In one strategy, the robot is designed using standard engineering principles, and problems that arise are addressed by looking to the animal world for a very specific solution. That solution is then implemented with no attempt to incorporate any other biological property. Under this strategy, the biological system is simply one source of information for solving engineering problems. With the alternate strategy, the robotic engineer attempts to capture as many of the animal's pertinent properties as possible in the robotic vehicle. To do this the entire behavior of interest and all related mechanical properties are studied in detail. The engineers then use appropriate simulation tools to capture the relevant properties of these systems in the designs of their vehicles. Under this "biology by default" scheme, the engineer accepts the biological properties of interest, such as those for locomotion, as the default unless they will create a serious engineering impediment (Ritzmann et al, 2000).

The Biorobotics group at Case has achieved success in the application of both strategies. Robot III (Figure 1.2) followed the biology as default strategy and was based as closely as possible on the structure and walking strategies that were

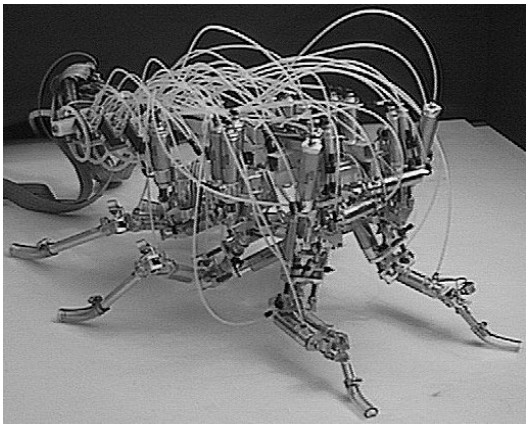


Figure 1.2: CWRU's Robot III

observed in the death head cockroach *Blaberus discoidalis* (Watson and Ritzmann, 1998, Bachmann, 2000, Watson et al. 2002). Each pair of its six legs was designed to capture the dimensions and joint architecture that are necessary for walking and climbing. Because the leg designs of the robot capture the kinematics of the insect particularly well, problems that exist in controlling its movements are relevant to control of the insect's movements. This relationship allows for direct implementation of the insect's control principles into the robot. Through study of biological control systems in nature, we also have achieved robust autonomous controllers capable of directing the actions of our walking robots (Espenschied et al. 1996, Nelson and Quinn, 1999).

1.3 An Overview of Actuation Devices

When designing a robot, especially when basing that design on one found in nature, the selection of actuators plays a pivotal role. The shape, size, weight, stroke, speed, strength and other aspects of an actuator must all be taken into account in the design, and the power source of the actuators often provides the greatest constraint on a robot's potential abilities. Biological organisms have a great advantage over mechanical systems in that muscle, nature's most prevalent actuator, has a favorable force-to-weight ratio and requires low levels of activation energy. Its tunable passive stiffness properties are also well suited for energy efficient legged locomotion. Furthermore, sensors such as tendon organs and muscle spindles provide a wealth of feedback information for the control of joints (Zajac 1989). Lastly, muscle is capable not only of self-repair, but growth in response to repeated use. All of these properties would be extremely valuable if they could be captured for use in robot design. Engineered actuation systems are far from equivalent to their biological counterparts

as currently the most popular mechanical actuators in use are electric motors and pneumatic/hydraulic cylinders.

Electric motors are probably the most commonly used actuation and control devices in modern day robotics, and with good reason. Motors in a wide range of sizes are readily available and they are very easy to control, either through the use of encoders or stepper motors designed specifically for position control (http://www.mpmmaxonmotor.com). These devices are also fairly easy to implement, normally requiring just a few electrical connections. However, electric motors have several disadvantages. Most importantly, their force-to-weight ratio is far lower than that of pneumatic and hydraulic devices, and in a field such as legged robotics, where weight is of the utmost importance, this makes them unsuitable for many applications. Typically, electric systems have a power to weight ratio of 50-100 W/kg, whereas fluid systems have a ratio of 100-200 W/kg (Kingsley, 2003). In addition, when trying to model a robotic system after a biological one, the drastic difference between the rotary motion of most electric motors and the linear motion of muscle can cause complications.

Pneumatic and hydraulic cylinder systems (discussed here together because of their many similarities) eliminate some of the problems associated with electric motors (Song and Waldron, 1989). As a general rule, they provide a significantly higher force-to-weight ratio than motors; an advantage that in itself often leads to their use, even given the increased complexity and weight of control valves and pressurized fluid lines required for operation. These actuators also produce linear motion, which makes them more suitable to serving a role equivalent to muscle. Unfortunately, air cylinders are often intended for “bang-bang” operation; that is,

motion from one extreme to another with mechanical stops to halt motion. Smooth walking motion requires a much larger range of states, and the stiction present in most pressure cylinders makes even course position control difficult. Fluid pressure devices are still quite weighty; for example, almost seventy-five percent of Robot III's weight is composed of its actuators and valves (Bachmann, 2000).

1.4 Braided Pneumatic Actuators

A key element of the research presented here is the braided pneumatic actuator (BPA), also known as the McKibben artificial muscle. Initially developed in the 1950's to serve as actuators in prosthetics, these devices share some important characteristics with actual muscle (Nickel et al., 1963). This as well as other demonstrated advantages suggest that these actuators will have an important place in the future of robotics if some of their limitations—specifically a short fatigue life—can be overcome.

The braided pneumatic actuator consists of two major components: an inflatable bladder around which is wrapped an expandable fiber mesh (Figure 1.3). Both ends are clamped together; one end of the bladder is sealed and the other is ported so that the device may be inflated. The resulting actuator, generally consisting of latex and PET mesh, is significantly lighter than a standard air cylinder made of aluminum and steel construction; however, the braided pneumatic actuator is actually capable of producing greater forces (and thus possesses a much higher force-to weight-ratio) than its heavier counterpart.

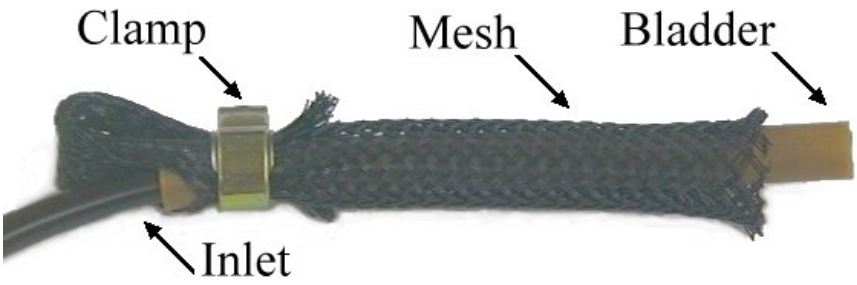


Figure 1.3: Cutaway view of a braided pneumatic actuator

When the bladder is filled with pressurized air, its volume tends to increase. Because of the constant length of the mesh fibers, this can only be accomplished by the actuator expanding radialy while simultaneously shortening (Figure 1.4). The result is a muscle-like contraction that produces the force-length curve akin to the rising phase of actual muscle (Figure 1.5) (Klute et al, 1999; note that the force-length relationship for dimensionless lengths greater than one becomes dependent on the material properties of the actuator, and will continue to rise until rupture). An important property of McKibben actuators to note is that at maximum contraction ($L/L_o \approx 0.69$) the actuator is incapable of producing force; conversely, the maximum possible force is produced when the actuator is fully extended. Therefore, similar to muscle, these actuators are self-limiting by nature. While an electric motor controller could conceivably become unstable and drive a system until failure of either the structure or the motor, a braided pneumatic actuator driven by an unstable controller cannot be driven to the point of damaging itself or the surrounding structure. Because of this property, braided pneumatic actuators are well suited for the implementation of positive load feedback, which is known to be used by animals including cockroaches, cats and humans (Prochazka et al., 1997).

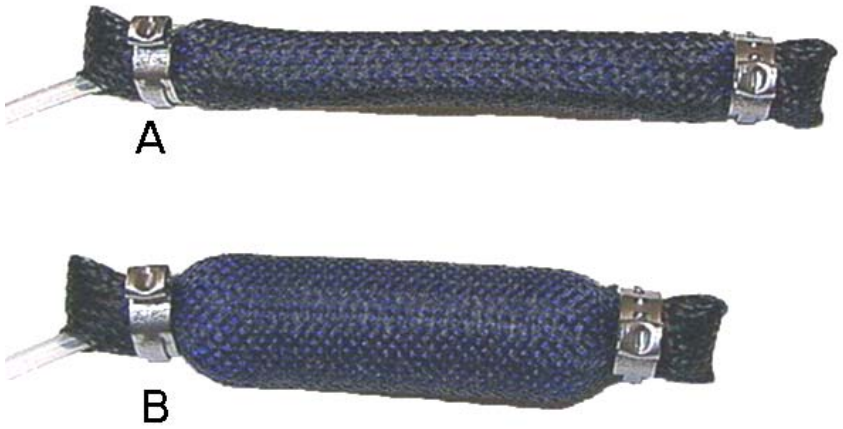
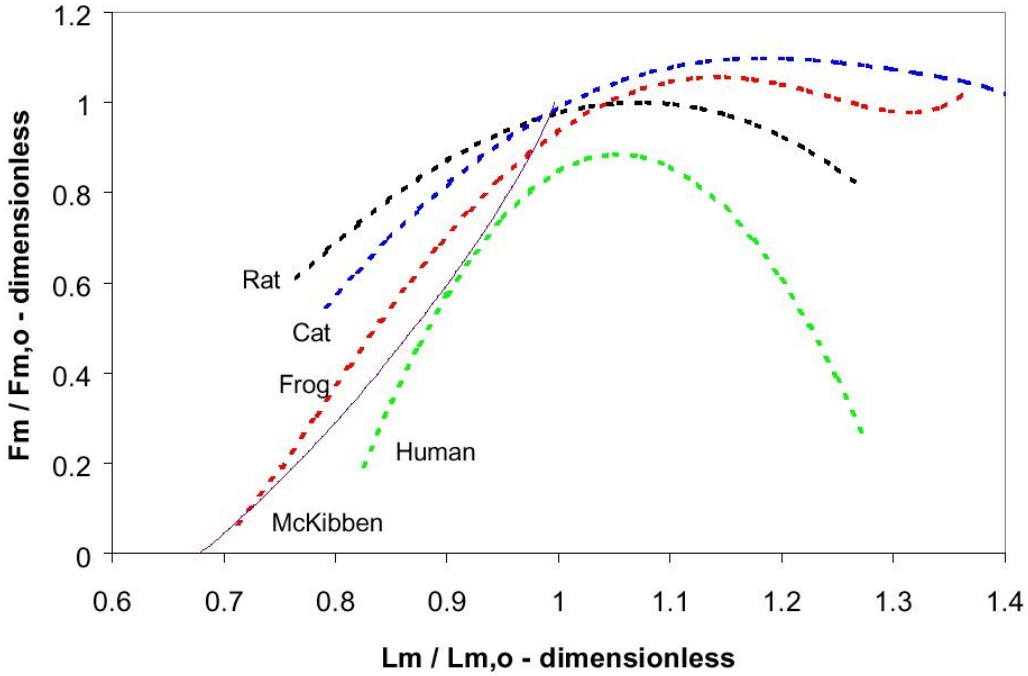


Figure 1.4: A braided pneumatic actuator at rest and inflated.



From: G.K. Klute, J.M. Czerniecki, B. Hannaford, 'McKibben Artificial Muscles: Pneumatic Actuators with Biomechanical Intelligence,' IEEE/ASME 1999 Intl. Conf. on Advanced Intelligent Mechatronics, Atlanta GA, September 19-22, 1999.

Figure 1.5: The force-length relationship of BPAs compared to that of muscle.

Braided pneumatic actuators have many additional characteristics that differentiate them from pneumatic cylinders but that are similar to actual muscle.

They are not capable of a stroke equivalent to half the actuator's length; instead, a practical stroke of only twenty-five percent of the actuator length is possible. While pneumatic cylinders are often ported at both ends to allow the actuator to either push or pull (double acting), McKibben muscles are pull-only devices. This means that they must be used in opposing pairs, as actual muscles are. This last property is of significant importance for useful application of these devices, for although it requires the use of two actuators or sets of actuators at each joint, it allows the muscle-like property of co-contraction, also known as stiffness control. If one considers a joint in the human body, such as the elbow or knee, it should be obvious that whatever position the joint is in the muscles that control that joint can be activated (flexed) without changing the joint angle. As a result, the joint angle remains the same but perturbations, such as the application of an outside force, result in less disturbance. From a practical standpoint, this means that the joint can be varied through a continuum of positions and compliances independently. The resulting joint can be stiff when needed, such as when bearing weight while walking, or compliant, as in cases of heel strike where compensation for uneven terrain may be needed.

1.5 Selected Legged Robots

There have been a great many attempts to construct legged robots, in which a wide range of leg designs, actuators, and controllers have been implemented. Although none have reached the ultimate goal of complete autonomy, many of these robots have demonstrated important advances in the robotics field. An excellent source of information on walking robots is the proceedings of the annual CLAWAR

conference (Virk et al. 1999, Berns and Dillmann 2001, Bidaud and Amar 2002, Muscato and Longo 2003)

The OSU Hexapod was one of the first large (4 ft long) six-legged robots. Each leg had three revolute degrees of freedom (McGhee et al., 1977). From the late 1970's through the 1980's, Hirose led research at the Tokyo Institute of Technology that resulted in many quadruped vehicles that used a pantograph mechanism for each leg (Hirose et al., 1989). These robots include the TITAN series of which TITAN IV was the largest, weighing 352lbs. The pantograph mechanism used in these robots' leg designs was a significant development as it decoupled the actuators used for weight support from those that generated horizontal motion. It also allowed for less massive legs, because all the actuators could be placed on the body.

Carnegie Mellon University's Ambler weighed nearly three tons and was a hexapod that used one revolute and two prismatic joints in each of its legs (Bares and Whittaker, 1993). CMU in collaboration with K^2T inc. also developed Dante and Dante II. These eight-legged robots, known as framewalkers, consisted of a body made up of three frames. The inner four legs were attached to an inner frame and the outer four legs were attached to an outer frame; the frames were connected by a middle frame. This middle frame could translate with respect to the outer frame and the inner frame could rotate with respect to the middle frame.

Several two-dimensional bipeds have been developed at MIT's Leg Lab under the guidance of Gill Pratt (Pratt, 1997). These machines use series-elastic actuators for energy efficient locomotion. Their most recent vehicle is a three-dimensional biped modeled after a human

(http://www.ai.mit.edu/projects/leglab/robots/robots.html). Prior to this, Raibert had built The Quadruped at MIT, and demonstrated that the principles of one legged hopping could be generalized for four legged running with the addition of a low level leg coordination mechanism (Raibert 1985, Raibert 1991).

Smaller hexapod robots with more promising controllers, some of which are inspired by insects, were developed starting in the mid 1980's (Brooks, 1989). Two such robots, Ghengis and Hannibal, were developed in Brooks' AI Lab at MIT. Ghengis weighed about 2lbs and had single segment legs that were moved through two revolute degrees of freedom at the shoulder. Hannibal had three degrees of freedom per leg. These robots demonstrated some of the advantages of distributed control versus centralized control.

The TUM Walking Robot was a hexapod that was modeled after the stick insect (Weidemann et al., 1994). Each of its six legs had three segments separated by revolute degrees of freedom. Its gait controller was based upon a network of mechanisms that were abstracted from observations of the stick insect in biological studies spanning several decades (Cruse, 1990).

A variety of robots actuated by pressurized air have been developed at research institutions across the country. At the Massachusetts Institute of Technology, Binnard constructed the robot Boadicea, modeled after the cockroach, *Blaberus Discoidalis*. Each pair of legs on Boadicea was designed differently to accomplish specific walking tasks. The simple front legs of this robot had two degrees of freedom each, while the others all had three; actuation was produced by pneumatic cylinders. This 10.5 pound robot was capable of carrying a five pound

payload and moving at four inches per second. Although surely a successful robot, Boadicea's most notable deficiency was its speed. This was due in large part to actuator instability; the legs required a pause of approximately half a second after each step to return to steady state (Binnard, 1995).

Robug IV, produced by Portsmouth Technology Consultants Limited (Portech) of Havant, UK, is a much more ambitious air cylinder driven robot. This eight-legged robot is designed with a focus on modularity and reduced cost. As such, all eight legs are the same, and can be easily interchanged and replaced. These legs each have four degrees of freedom. Robug IV weighs about forty kg, and is capable of supporting the weight of two people standing on it. At the time this paper was written, Portech had attained control of individual legs, but had not yet made Robug IV walk (Cooke et al. 1999).

At the University of California at Berkeley, Powers constructed a hexapod with twelve degrees of freedom, driven by braided pneumatic actuators. Although this robot was capable of lifting its own weight, it was not able to walk. This failure may be attributed to the manner in which the actuators were implemented. As previously mentioned, braided pneumatic actuators must be used in opposing pairs, just as biological muscle is. Powers, in an attempt to reduce valve weight, used only one actuator per joint, with a spring to provide a restoring force. Although this did reduce the robot's weight, an important feature of the braided pneumatic actuator was overlooked: its self-limiting properties in both contraction and extension (both cases are attributed to the fibers of the mesh). Furthermore, the joints were configured in such a way that the spring was compressed and extended with the actuator instead of

in opposition; as a result, there was no limiting mechanism on the joints in the direction of actuator contraction. This significant oversight resulted in the legs buckling when walking was attempted (Powers 1996).

The leg geometry of Protobot is based upon the cockroach, *Periplaneta Americana* (Delcomyn and Nelson, 1999). Each of its legs has three degrees of freedom, but the front, middle and rear pairs of legs have different geometries reflecting differences found in the animal. It is actuated with air cylinders.

A number of recent robots have been constructed using abstracted biological principles in an attempt to simplify many of the engineering problems that develop when closely adhering to biological properties of animals. Foremost amongst these are the Whegs (CWRU), RHex (McGill University, University of Michigan), and the Sprawl (Stanford) series of robots. All three of these families of robots use elemental biological concepts, such as hexapod morphology and tripod gaits, to produce easy to manufacture and simple to control robots capable of autonomous locomotion. While in many ways successful, these robots are limited in their capabilities, and many of the advantages gained through rigorous adherence to biological principles are lost.

Case Western Reserve University's Biorobotics group has previously developed three biologically inspired robots, aptly named Robot I, Robot II and Robot III. Robot I (Figure 1.6) was constructed primarily from model airplane plywood. Its twelve degrees of freedom were actuated by DC electric motors, and consisted of one rotational and one telescoping joint on each of the six legs (Quinn and Espencheid, 1993). Although not very biologically inspired mechanically, this robot implemented an insect-like neural network for gait control (Beer et al., 1993)

and was the first reported robot to implement a Cruse controller (Cruse, 1990)(Espenschied et al., 1993). Robot II (Figure 1.7) was also driven by DC motors, which actuated three rotational degrees of freedom on each of its six legs, all of which were kinematically and mechanically the same. This highly successful robot was capable of navigating rough terrain and slatted surfaces consisting of only fifty percent substrate. Sensors on this robot consisted of single turn potentiometers to determine joint angle, and strain gauges to determine whether a leg was in contact with the ground (Espenschied et al., 1996). Robot III (Figure 1.8) was a radical departure from the design of its predecessors, and is the first to have its design based upon a specific animal—*Blaberus discoidalis*. All joints are actuated by pneumatic cylinders, and each pair of legs (front, middle, and rear) are designed to perform specific tasks (Bachmann et al. 1997). The front legs, as on the actual animal, are quite nimble, each with five degrees of freedom. The middle legs are larger, possess four degrees of freedom, and assist primarily in climbing and turning. The rear legs are the largest and most powerful; with only three degrees of freedom they are intended to push the robot forward while walking. This robot also has strain gauges mounted on both the tarsus and tibia of each leg. These sensors enable the robot to not only determine if its legs are in contact with the ground, but to measure the force vector with which each leg pushes against the ground. At the time of this writing, this powerful robot is capable of stable posture and performing "pushups" while lifting a payload equal to its own weight. Furthermore, it can swing its legs smoothly and coordinate them in a tripod gait; however, it has not yet walked smoothly. It is believed that this failure to walk was a result of the limited bandwidth of the

actuators; the robot was not able to respond to perturbations quickly enough to maintain the necessary joint angles for walking.

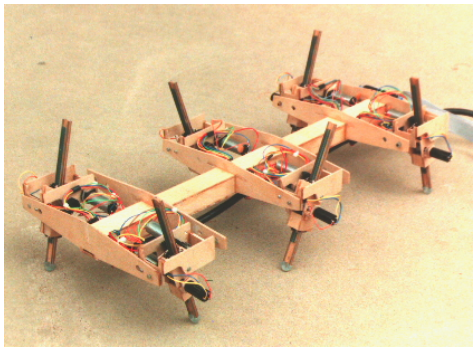


Figure 1.6: CWRU's Robot I

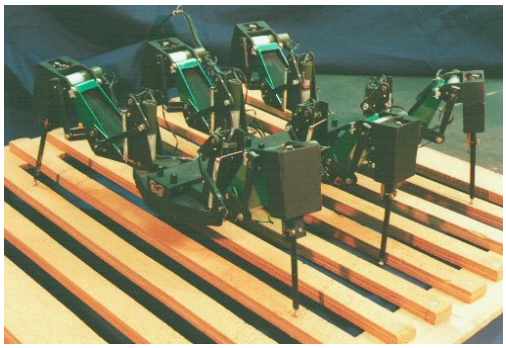


Figure 1.7: CWRU's Robot II

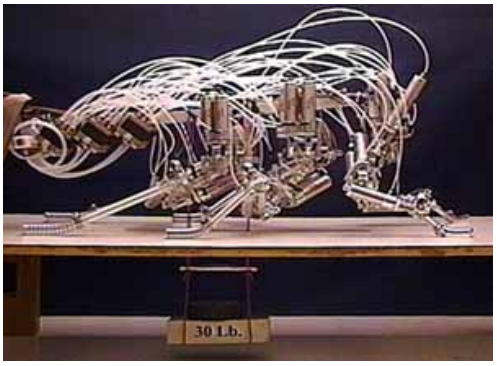


Figure 1.8: CWRU's Robot III lifting a 30 lb payload

A variety of robots have been constructed to demonstrate the viability of braided pneumatic actuators. At CWRU, a simplified planar leg was built with two degrees of freedom (Colbrunn, 2000). This leg was attached to a trolley and was able to move in a sensible, walking motion. Furthermore, this leg demonstrated the efficiency of McKibben muscles, as the leg was found to be ninety percent passive; that is, the valves controlling the actuators only needed to be open ten percent of the time to produce the desired motion. This low use of power and air will play a key role in making an autonomous, mission capable robot.

The Shadow Robot Company (www.shadow.org.uk) has produced a more advanced biped robot using McKibben muscles. Unlike the CWRU leg, this robot is modeled after a pair of human legs. It is capable of standing, suffering minor perturbations, and even small, lurching steps. At the University of Washington, researchers are working to develop a robotic arm closely modeled after that of a human (http://rcs.ee.washington.edu/brl/project/aarm/). The skeletal structure of this arm is made from casts of human bones, and is overlaid with braided pneumatic actuators attached at biomechanically accurate locations.

Researchers at the University of Salford, led by Caldwell, constructed a simplified biped robot that was actuated by McKibben muscles. Only four pairs of actuators were used, one pair moving each hip and one pair actuating each leg. This robot was reportedly able to stand and perform rudimentary walking motions. However, it did not have an active balance controller, and thus was not capable walking for extended periods or over complex terrain (Caldwell et al., 1997). More recently, Caldwell has led research on a torso-and-arm robot that also utilized McKiben muscles for actuation (Costa et al. 2002).

At the university of Karlsruhe in Germany, a hexapod robot called AirBug was constructed using Festo fluidic muscle (described in more depth in Chapter III) for actuation. This robot had three degrees of freedom per leg—all of which were the same—and was roughly based on the morphology of the walkingstick insect. Airbug was approximately three feet long, weighed sixty pounds and was capable of slow, but stable, walking (Kerscher et al, 2002).

1.6 Motivation

At the most basic level, the goal of this research was to construct a legged robot that modeled *Blaberus discoidalis* cockroach – specifically the leg designs and scaled inertia properties – and moved its legs in an agile manner. This is the fifth in a series of robots that each have made progress toward this goal. What differentiates this work from many robots developed throughout the world is the attention paid to the properties of the organism upon which it was based. While previous robots have been built using biologically inspired kinematics and joint structures, an important goal of this project was the implementation of a muscle-like actuation system. Why? It was believed that by having a mechanical system geared toward locomotion, and one similar to those found in nature, implementation of a neurologically inspired controller would become much easier. In short, it was desired to move the lowest level of control—reflexes, postural stability—from a control issue to one that was dealt with, at least in part, by the mechanics of the system. If one considers a biological organism that has been perturbed, for example a human being bumped on a busy sidewalk, very little active control seems necessary to recover from the disturbance; this is instead handled by either low level reflexes or the compliant properties of the actuators and joints themselves. In contrast, many walking robots, even those biologically based, require comparatively more control to maintain posture and deal with small perturbations and control inaccuracies. By using muscle-like actuators, BPA's in specific, it was believed that a robot that was much more energy efficient and easier to control could be developed. Indeed, research indicates that muscle can produce large, nearly instantaneous changes in output in response to

sudden variation in kinematic conditions (Loeb et al. 1999). If this feature could be reproduced in a robot, a significant portion of the control of the system could be provided by the features of the actuators themselves.

1.7 Overview

This work details the design and control of a 24 degree of freedom, cockroach-like robot built at Case Western Reserve University called Robot 5, or "Ajax". The paper can roughly be broken into three sections. The first focuses on the design, fabrication, and modification of Robot 5. Chapter II specifically addresses the design methodology used in the development of the robot. Chapter III explains in detail the operation and properties of the actuators used onboard the robot. Chapter IV covers the manufacturing and assembly processes used to construct Robot 5. Chapter V presents the initial tests that were used to evaluate the performance of the robot in its early stages. Chapter VI is a review of modifications that were necessary to bring the robot to an operational level. Chapter VII highlights two very different methods of dealing with slack or "kinking" problems inherent in the actuators. Chapter VIII ends the first section by discussing the proprioception sensors implemented onboard the robot. The second section of this paper focuses on a portion of the control system implemented onboard the robot. Because the overall task of control was divided between three different researchers, only the interleg coordination is detailed in this work. Chapter IX covers the basics of the leg coordination mechanism, the Cruse Controller. Chapter X expands on this controller, explaining how the one dimensional model was expanded to produce planar motion. In the final section of this paper, the results of the research, including future

suggestions are presented. Chapter XI presents the capabilities of the robot as of the completion of this work. Chapter XII, on somewhat of a tangent, is an explanation of an onboard compressor system that could be used to make future robots autonomous. Chapter XIII compares the performance of the robot to that of the incest. Finally, Chapter XIV contains suggestions for future design concepts, both mechanical and control, as well as a summary of the performance of this robot. Following these chapters is an appendix of the code utilized to produce the Cruse Controller, which should be flexible enough to be implemented on multiple platforms without too much difficulty. A second appendix contains a listing of the actuators in use onboard the robot.

Chapter 2: Design of Robot V

2.1 Design Aspects of Previous Robots

A discussion of the design of Robot V must be prefaced with a discussion of important design aspects of its predecessors, specifically Robot II and Robot III. This series of robots represents an iterative design process, with each one offering improvements over the last. Indeed, it may be difficult to understand some of the improvements made by Robot V without first understanding the deficiencies of its forbearers.

Robot II (figure 2.1) was considered to be a very successful robot. It had three degrees of freedom per leg. This provided a reasonable range of motion for its feet while maintaining a single solution kinematic chain in each leg. The robot was capable of navigating a variety of complex terrains, and clearly demonstrated the robustness and adaptability of a biologically inspired controller. Indeed, much of the current research in the field is deeply based in the mechanical and control architecture of Robot II. Despite its capabilities, Robot II had some distinct disadvantages, the most noticeable of which was its lack of power; it was barely capable of lifting its own weight, and was not capable of carrying an onboard controller or power supply. As discussed earlier, this was in large part a result of the poor power-to-weight ratio of the electric motors used to actuate the robot (Espenschied et al. 1996).

Figure 2.1: Robot II walks over a slatted surface.

Much of the work done prior to the construction of Robot III was focused on determining the minimum number of joints needed to reasonably reproduce the motions of a cockroach's limbs. Although the animals have a total of seven joints per leg, this is a challenging number of DOF to implement on a robot, as both the system mechanics and control become cumbersome. The necessary distribution of DOF was arrived at through the careful study of live cockroaches during locomotion. The Ritzmann Lab at Case has the capability for both high-speed video and EMG recording of the animals. This analysis showed that cockroach walking and climbing motions could be reproduced with only 24 DOF (Watson and Ritzmann, 1998, Bachmann et al. 1997, Watson et al. 2002, Ritzmann et al. 2004). Robot III (figure 2.2) implemented this reduced joint structure, distributed through the task-oriented legs. The front legs each had five degrees of freedom, resulting in agile, yet relatively weak limbs. The middle legs had four degrees of freedom; they were less agile but more powerful than the front legs. These limbs were used mainly for weight support, rearing and turning. The rear legs needed but three degrees of freedom; they are powerful yet not very dexterous (Bachmann 2000).

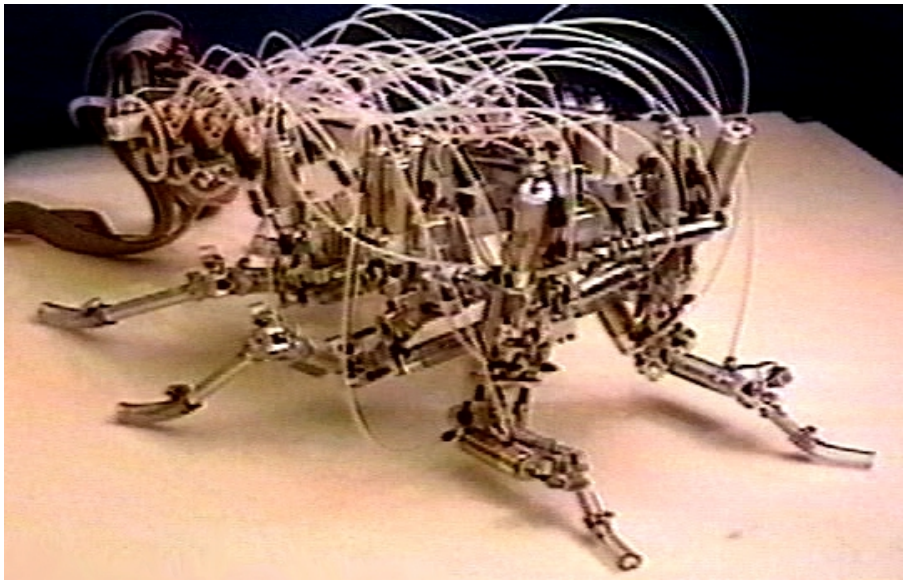


Figure 2.2: Robot III standing.

Robot III made many other significant design improvements over Robot II. An aluminum construction provided greater durability and strength. Actuator power issues were addressed by switching from electric motors to pneumatic cylinders. Although this greatly increased the loads the robot was capable of carrying, these actuators proved difficult to control. This difficulty in actuator control ultimately proved to be the robot's greatest failing; although it was capable of smooth, coordinated motion while suspended above the ground and maintaining posture after perturbation, it was not capable of combining the two for robust, agile walking. This failure is attributed in large part to the low bandwidth of the pneumatic cylinders—the controller simply could not respond quickly enough to make the robot stably walk (Nelson 2002).

Robot IV (figure 2.3) was very closely based on Robot III, with two significant changes: the pneumatic cylinders were replaced with BPAs and the number of valves was doubled to enable air to be trapped in the actuators. This robot

suffered significant strength problems; it was barely able to lift itself while carrying the valves necessary to control the actuators. Even with this lack of power, the robot was able to walk in a wave gait without the use of any sensors. Further problems were caused by the relatively short fatigue life of the BPAs, as well as the difficulties inherent in replacing them because they had to be custom manufactured in house.

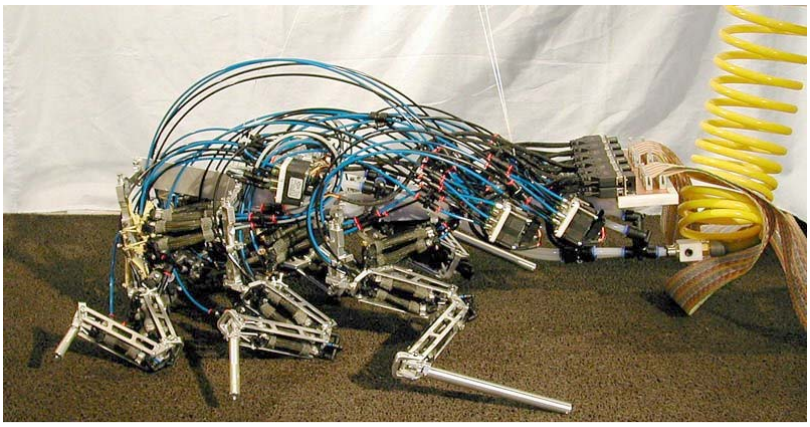


Figure 2.3: Robot IV.

2.2 Stance Bias

The actuators onboard a legged robot can generally be subdivided into two classes: those used to move the limb through the swing phase and those required to maintain stance and generate locomotion. One of the fundamental differences between these two types of actuators is the load that is required of them. The swing actuators need only provide the force necessary to overcome the weight and inertia of the limb, whereas the stance actuators must support not only a significant portion of the entire mass of the robot, but also provide the force necessary for locomotion. This disparity between operational demands can potentially lead to large, powerful stance actuators and small swing actuators (as can be seen in many quadrupeds with powerful biceps muscles which maintain stance, and the respectively weaker triceps

muscles, which are used for swing); however, because of limited options for robot actuator sizes, it is more often the case that the swing actuators are overpowered, whereas the stance actuators are either underpowered or just capable of meeting the demands placed on them.

One of the most noticeable problems demonstrated by Robot IV was its lack of strength. All of the actuators onboard the robot had the same diameter, and thus produced the same force; furthermore, in most cases the robot used the same number of actuators for both flexion and extension of a joint. Additionally, most of the stance actuators onboard the robot were shorter than those placed in opposition to them. Although actuator force is independent of length, longer actuators have longer stroke lengths; use of a longer actuator allows an increased moment arm to be used, while moving the joint through the same angle. Conversely, shorter actuators have a smaller displacement and must therefore be used in conjunction with smaller moment arms to provide the desired range of motion, and are thus incapable or producing as much torque at a joint.

On Robot V these problems were resolved through careful actuator selection (described in more detail in the following sections) and the placement of torsion springs at some critical load bearing joints (specifically the coxa-femur and β joints) to provide a bias in the direction of stance (figure 2.4). As a result, the forces required of the stance actuators are significantly reduced and while the swing actuators must produce greater forces, they still remain within their operational range.

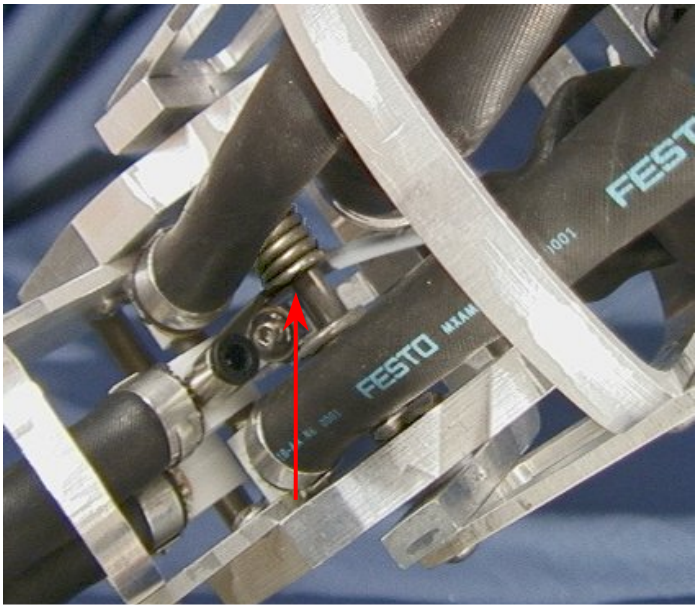


Figure 2.4: Close up of a middle leg coxa-femur joint on Robot V. The arrow points to a torsion spring.

2.3 General Aspects of Leg Design

Ajax uses the same joint architecture as Robot III: five DOF in the front legs, four DOF in the middle legs, and three DOF in the rear legs. The data that were used to determine these configurations were gathered in the Ritzmann Lab by James Watson and analyzed by Gabriel Nelson in the Biorobotics Lab. Although each pair of legs on the robot was designed to perform specific tasks, there are many elements inherent in their designs that are shared by all legs. Before delving into the specifics of the individual leg pairs, these common traits will be discussed to form a base for further, detailed discussion about specific leg designs.

Although the legs have different numbers of DOF, the same scheme is used to refer to leg segments and joints on all legs (figure 2.5). The most proximal leg segment is called the coxa, the next segment is the femur, followed by the tibia, and most distal is the tarsus. The body-coxa joint can consist of up to three DOF. These

are called the body-coxa α, which has an axis normal to the coronal plane (in the X direction), the body-coxa β, which has an axis normal to the median plane (in the Y direction) and the body-coxa γ, which has an axis normal to the transverse plane (in the Z direction). The coxa and femur are connected by a single joint, as are the femur and tibia. The complex tarsus joints were not modeled on the robot, but instead a semi-rigid passive element was used to make the tarsus. As mentioned earlier, the middle and rear legs had reduced DOF. The middle legs were designed without an active γ joint; instead, the assembly was constrained at a constant angle of 30º. The rear legs also froze the active γ joint at 30º and in addition constrained the α joint to a constant position of 75º. These values were determined to provide motion reasonably close to that of the animal while reducing the total DOF, and thus the complexity of the robot.

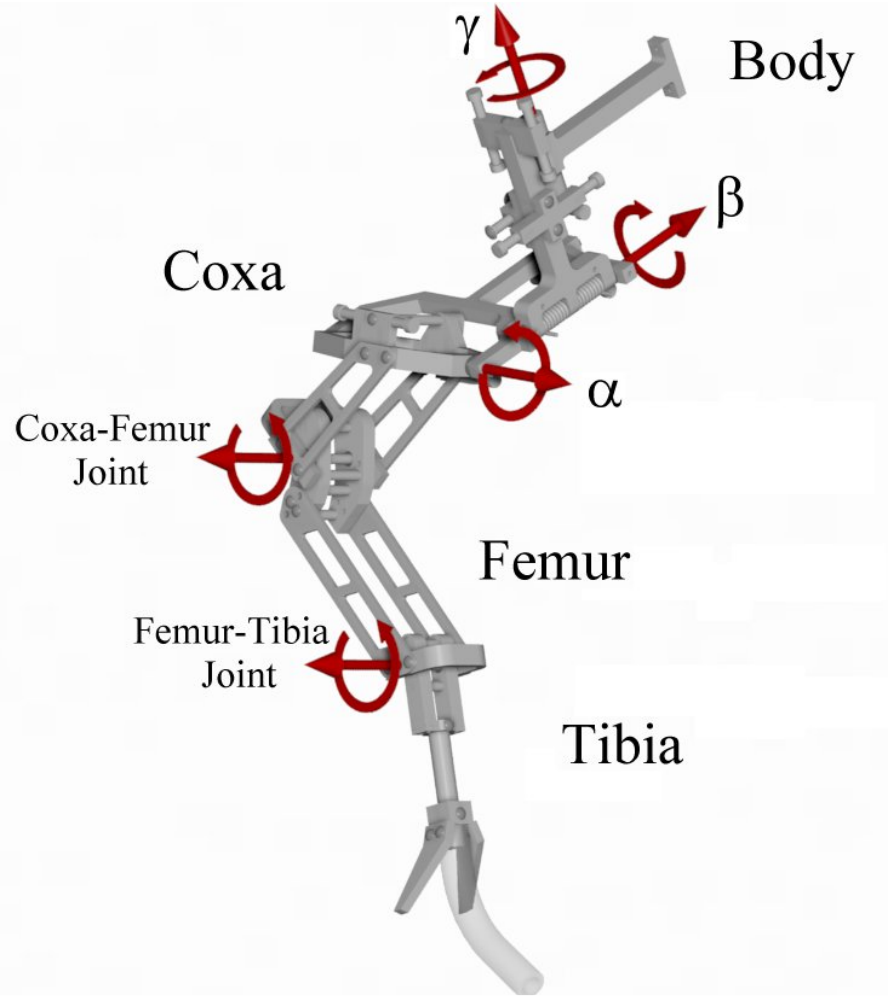


Figure 2.5: Schematic of the front leg showing joint axes.

Before any of the physical structure was designed, it was first necessary to determine the length of limb segments, approximate joint torques, and approximate range of motion for each joint. The lengths of limb segments were chosen to remain kinematically similar to Robot III; in turn, the lengths of limb segments on Robot III were chosen to be kinematically similar to those of the cockroach (Bachmann 2000). For comparison, leg segment lengths are presented as ratios with respect to the rear tibia; Robot V was scaled to be approximately twenty times larger than the animal.

Table 1.1: Relative limb segment lengths.

Leg segment length with respect to rear tibia length

Front Leg	**Insect**	**Robot**
Coxa	0.591	0.619
Femur	0.599	0.619
Tibia*	0.347	0.452
Middle Leg		
Coxa	0.681	0.667
Femur	0.731	0.714
Tibia	0.611	0.619
Rear Leg		
Coxa	0.751	0.714
Femur	0.800	0.762
Tibia	1	1

*The front tibia of the robot was later modified to a length ratio of 0.595

Desired ranges of motion (ROM) were again based on those of Robot III (Bachmann 2000), which in turn were based on studies of the animal during walking, but inflated to provide a margin of safety for the design of the robot (Watson and Ritzmann 1998, Watson et al. 2002).

Table 1.2: Desired robot ROM.

Anticipated Joint ROM (degrees)

	Front	**Middle**	**Rear**
γ	40	xxx	xxx
α	40	40	xxx
β	30	30	40
C-F	50	50	50
F-T	75	75	75

At this point, an assumption was made that the robot would weigh approximately thirty pounds; this was the weight of Robot III, and although Robot V would be larger than its predecessor, the much lighter actuators were expected to offset the increased weight due to the larger structure and increased number of valves.

Using this, the length of the limb segments and approximate joint angles during stance, it was possible to approximate the torques at each joint if the robot were in a tripod stance (front and rear leg on one side and middle leg on the opposite side contacting the ground) and weight were equally distributed between the legs. These torques were then doubled to account for and inaccuracies in design approximations. This process quickly revealed that certain joints, specifically the coxa-femur, α and β joints, would be much more heavily loaded than others. On the other hand, the femur-tibia joints would see relatively little load.

Table 1.3: Anticipated joint torques.

Anticipated Joint Torques (in-lbs)			
	Front	**Middle**	**Rear**
γ	100	xxx	xxx
α	120	150	xxx
β	75	150	200
C-F	180	200	225
F-T	40	40	50

A great deal of care was taken to determine the lengths of actuators required to drive each joint. This was in most cases an iterative process, due to the number of variables that had to be factored into each actuator/joint design. Longer actuators produce the same force as shorter ones, but longer actuators have a proportionally longer stroke. Greater actuator stroke allowed a larger moment arm to be used for actuator insertion at the joint while producing equivalent joint range of motion, and as a result, greater torques could be produced at the joint. However, using longer actuators come at the cost of having to build a larger robot. These issues were compounded by the fact that it was not possible to utilize the entire stroke of the actuator; this is because at maximum contraction, BPAs produce no force (Klute,

Hannaford 1999). Thus, a minimum actuator length had to be found that was still capable of supporting the loads (stance or swing) that would be placed on the actuator. Desired actuator lengths were found assuming an actuator that behaved as a theoretical BPA. Safety margins were added by increasing the desired actuator length by one-half to one inch whenever the dimensions of the robot allowed, and in some cases—again, where robot dimensions allowed—doubling the number of actuators used.

The majority of the structure was designed as an exoskeleton. Not only did this reduce overall weight while maintaining strength, but it also afforded a level of protection to the actuators. Stress analysis was only performed for what were deemed critical stress points, and in all cases the design parameters were found to be within or far exceed reasonable safety margins; in all situations a safety factor of three or more was required, and no analysis resulted in a safety factor below ten. More often that not, component dimensions were constrained not by stress levels but by dimensions of parts purchased from outside suppliers, specifically as bearings and the torsion springs placed in some joints. Steel shafts ¼" in diameter were used as joint axles and actuator mounts, and the smallest nylon journal bearings available for these were 3/8" O.D. and 3/8" long. The torsion springs used to bias joints were 2.1" long, requiring members containing these to be spaced at least that far apart.

Foot design of all legs was fundamentally the same. This consisted of a semi-rigid member in place of the tarsus and a pair of claws. The flexible tarsus, made of wire reinforced PVC tube, was intended to provide traction on slippery terrain. The

claws were angled differently on each leg such that they would dig in to a substrate while that leg was in stance.

Custom ends were designed for the Festo actuators that would allow easy mounting on the robot; specifically, it was important to keep the ends as short as possible to preserve the working length of the actuators. Furthermore, because leg segments would rotate with respect to each other, by designing ends that would rotate on a circular mounting shaft actuators would always produce forces directed along their axis, and actuator kinking would be reduced. Whenever possible, actuators were mounted directly to a joint, and tendons were not used as these would simply reduce the available stroke length of an actuator. The only case where tendons were incorporated into the design were the body-coxa β joints, where the use of tendons allowed the design of a robot with a much lower profile and did not detract from the available actuator length.

2.4 Design for Assembly and Disassembly

Ease of assembly and disassembly is an important, but often overlooked design aspect. This is especially true for a prototype/research platform such as Robot V, which was expected to undergo a great deal of modification during its life. An effort was made in the design phase to use the same parts in different applications whenever possible; for example, the components of the coxa-femur joint are the same for the middle and rear legs, and some of those parts are also used in the front legs. For consistency and to further increase ease of assembly/disassembly, as few sizes of machine screw as possible were used. This had the added advantage of reducing the number of tools needed during the manufacturing process as well. Most structural

connections were made using 8-32 thread, button head machine screws. Although three different lengths were required, all of these could still be loosened and tightened with the same size allen wrench. In some cases, design parameters and component sizes did not allow the use of 8-32 screws, and 4-40 button head screws were used instead. To prevent rotation of axels with respect to their mounts (those from which they were not separated by a bearing) 2-56 set screws were used.

2.5 Rear Leg Design

The rear legs (figure 2.6) of the robot are mechanically the simplest, as they have only three degrees of freedom; however, they are required to bear the majority of the forces generated in stance and locomotion (Bachmann 2000). Because of this, these legs must have very powerful actuators, specifically at the body-coxa β and coxa-femur joints. It was possible to incorporate long β actuators into the design without making the robot unmanageably tall by mounting them in a plane parallel to the body (A), and attaching them to the joint via steel cable tendons. To produce forces in the desired directions, these tendons wrapped around axles attached to the mounting armature of the leg (B). The coxa-femur joint is one of the most critical load bearing and propulsive joints on the robot; on Robot IV it was also one of the most underpowered, due in large part to extensor actuators that were far too short. This issue was addressed on Robot V by extending the actuator mounting elements further into the joint (C). To further increase the load bearing abilities of these two joints, torsion springs were attached to the joints and preloaded so that they would provide approximately 50% of the moment required to maintain stance—approximately 100 inch-pounds. To ensure enough torque was available at these

joints, pairs of actuators were used instead of single actuators. Another problem exhibited by Robot IV was yielding of the coxa and femur exoskeletons due to the large bending moments placed on the material during stance (Bachmann 2000). The design of Robot V alleviated this problem by using thicker exoskeleton material and the inclusion of ribs (D) throughout these two segments.

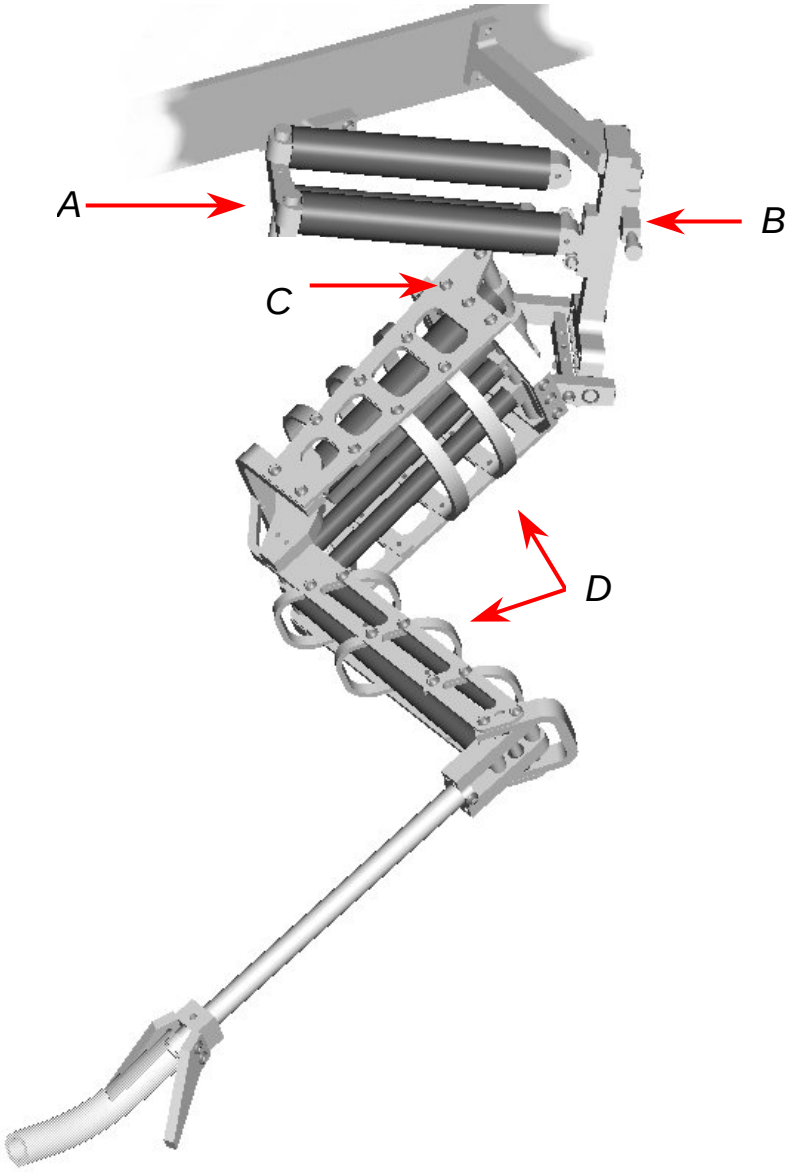


Figure 2.6: Rear leg of Robot V

2.6 Middle Leg Design

The middle leg (figure 2.7) utilized many of the same design elements as the rear leg, although some significant changes had to be made to allow the incorporation of the body-coxa α joint. With the exception of some length differences and the mounting of the claws, the rear and middle leg coxa-femur joints and all more distal elements are identical in design. Furthermore, the same mounting scheme was used for the β actuators. Torsion springs were also attached to the body-coxa β and coxa-femur joints of the middle leg, just as they were on the rear leg. The coxa of the middle leg is not as long as that of the rear leg, and because of its orientation during stance, it experiences lower bending moments (150 vs. 200 inch-pounds). This allowed for a lighter exoskeleton design using only one spacer (A), thus reducing the weight of the limb. The most noticeable difference between the rear and middle legs is the inclusion of the body-coxa α joint (B). Much like the coxa-femur joint, this joint was critically underpowered on Robot IV. Given the joint orientation, this is specifically true for the extensors of the joint, which are, responsible for bearing very large loads during stance. Although the design did not allow easy attachment of a biasing torsion spring at this joint, it was possible to space the legs slightly further out from the main body than on the actual animal, allowing more space between the legs for actuators. In short, the legs were extended far enough from the body so that actuators of sufficient length could be attached. This required the proximal arms of the β joints to be separated by ten inches, while if the robot were built using the relative spacing on the animal, the separation would have been only 6 inches.

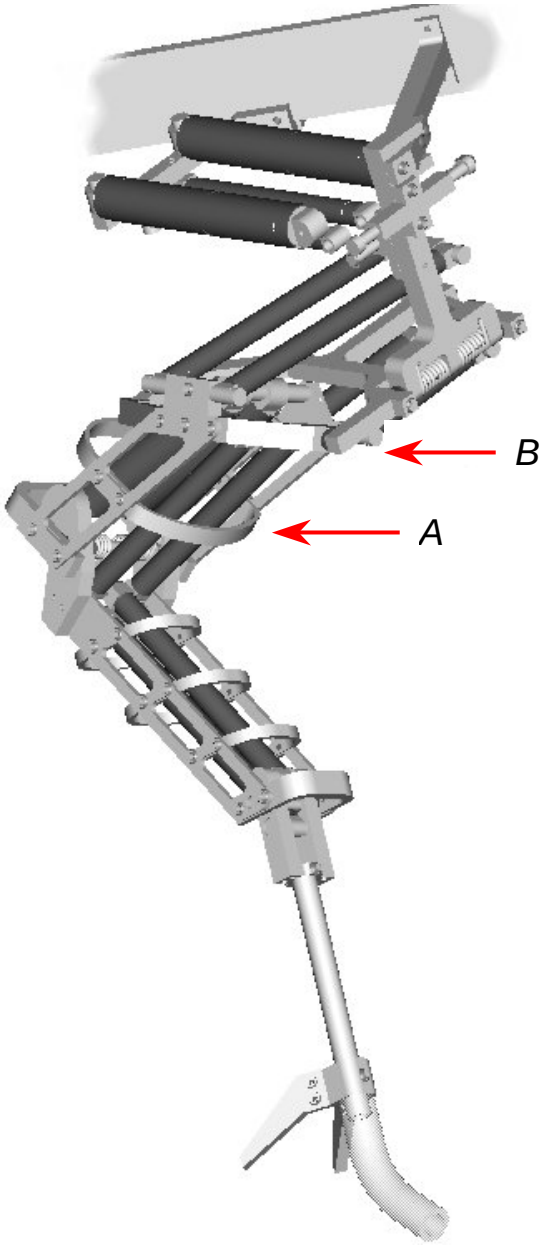


Figure 2.7: Schematic of Robot V's middle leg.

2.7 Front Leg Design

The front legs of the cockroach are the most dexterous on the animal, and to capture this property it was determined that three joints between the body and coxa would be necessary (Watson and Ritzmann 1998, Watson et. al. 1999). A variety of biological solutions to this problem exist, such as the ball-and-socket, or spherical, joint such as in the shoulder and hip of a human, or a series of trochantins, as can be found in the cockroach. These versatile joints can rotate about three axes, but have many drawbacks from a mechanical point of view. Actuation in one direction can stretch actuators responsible for other motions, making coordinated movements very difficult. This is further compounded by the fact that most conventional sensing means, such as potentiometers, cannot easily be attached to such joints, as they are capable of rotation about only one axis.

The most common solution to this problem is to design a mechanism such that three separate axes of rotation intersect at a single point, resulting in a mechanism that acts like a spherical joint about that point. The front leg body-coxa interface of Robot V (figure 2.8) was designed in this way, but because of space limitations for actuators, the axis of the α joint (A) had to be offset slightly from the intersection of the β and γ axes (B). This portion of the design was further complicated by the mounting scheme of the actuators for this joint. Because the joints form a series of linkages, actuators should be attached directly to the two links they actuate; if they cross additional links, motion of those links can result in changes in length between the origin and insertion of the actuator without the actuator being activated. Alternately, activation of the actuator may result in motion of joints beyond the one it

was intended to drive. This process occurs in nature and is called tenodesis; this can be seen quite clearly in the wrist, where flexion of the wrist causes extension of the fingers, and conversely extension of the wrist causes flexion of the fingers (Jenkins 1998). In the design of the robot, it was easiest to array the body-coxa joints of the front leg in order of proximal to distal as γ-β-α, but in keeping with the β actuator mounting scheme of the other joints (C), some tenodesis would occur at that joint. The alternative would have been to mount the actuators vertically, but not only would this have significantly increased the height of the robot, but there would also have been additional problems involved in the mounting of the actuators in this orientation. In the end, it was felt that the effects of the tenodesis would be minimal, and with foreknowledge of this issue, steps could be taken to eliminate any problems that could occur.

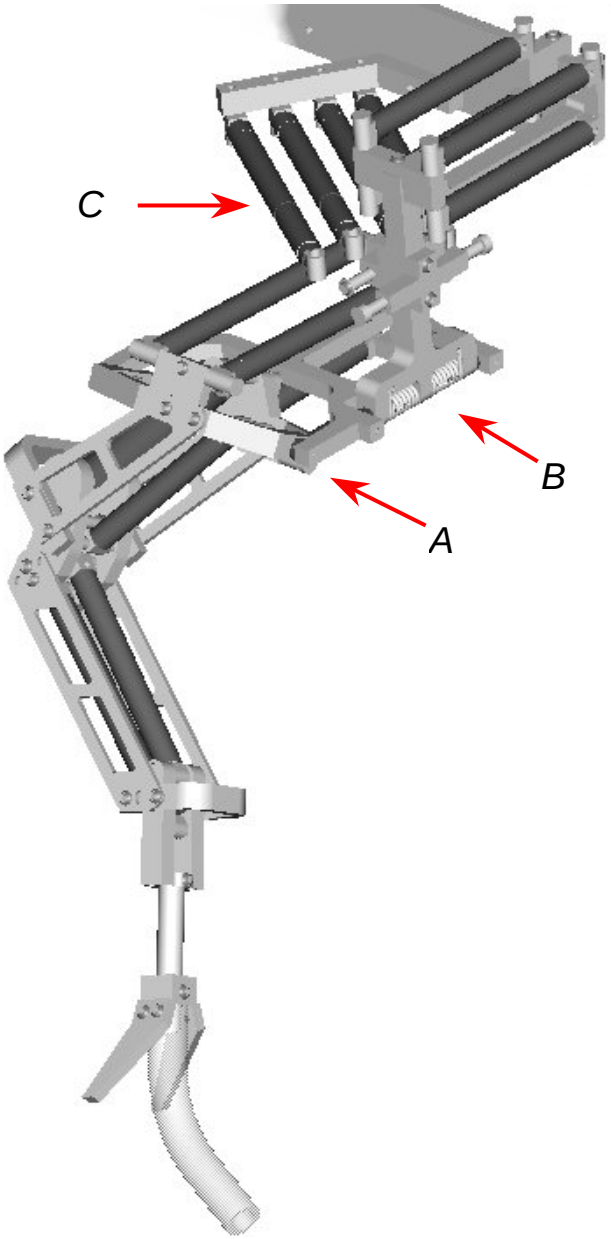


Figure 2.8: Schematic of Robot V's front leg.

Because the orientation of the front leg resulted in much lower joint torques, torsion springs were not placed at any of the joints. Removal of the spring at the coxa-femur joint allowed the entire joint to be designed much thinner, as the length of the spring had been the critical dimension for this joint on the other legs. Again, because the front legs were not required to carry as much load as the others,

calculations showed that the coxa-femur extensor, which on other legs consisted of a pair of 20mm actuators, could sufficiently be driven by a single 20mm actuator. Although this would reduce the overall power of the front legs, it also allowed for a smaller, lighter front leg as well as reducing the compressed air requirements to drive that leg. Lastly, calculations showed that this was in fact not the weakest joint on the leg; the body-coxa α joint had the least "surplus" torque, and in fact this joint later had to be modified to account for this failing (see chapter VI).

Chapter 3: Modification of Festo Fluidic Muscle Actuators

3.1 Festo Fluidic Muscle

The Festo corporation, located in Germany, has developed a variant of the braided pneumatic actuator that they have dubbed “fluidic muscle” (Figure 3.1). These devices function using all of the same principles as the BPA, but possess the important characteristic of having the fiber mesh embedded inside the expandable bladder, much like fiber or metal reinforcements are embedded in a tire or high-pressure hose. This results in a number of advantages.

Figure 3.1: A Festo Fluidic Muscle Actuator.

Due to its construction, the Festo actuator is significantly longer lived than either commercially available BPA’s or those manufactured in-house; whereas actuators made at Case lasted at best 15,000-20,000 cycles at 100 psi, Festo reports lifetimes in excess of ten million cycles.

Festo fluidic muscle is a hose—a specially made hose—but a hose nonetheless. Because it is available in continuous lengths of up to 30 feet, it is not only easy to cut actuators to a desired length, but repair and replacement of damaged actuators is a quick and easy process, unlike the time consuming manufacturing process required for in house production of standard braided pneumatic actuators.

The latex used as the expandable bladder of BPA's manufactured in house is notoriously susceptible to environmental damage, specifically from chemicals (solvents and oils) and ultraviolet light. The actuators on Robots IVa and IV had to be periodically replaced due to degradation of the latex (Kingsley 2001). Festo fluidic muscle, on the other hand, is intended for industrial conditions, and is much less susceptible to environmental damage.

As a result of its construction, the Festo actuator has a significant restoring force, as the equilibrium position of the rubber is in the fully extended state. Although this has many effects on design implementation (which are discussed elsewhere) the most apparent effect is a slight loss in the force produced compared to a BPA of equivalent size. This slight loss is deemed acceptable due to the aforementioned benefits, and in itself results in one small advantage. Previous research has shown that the limiting factor of BPA bandwidth is the time necessary to exhaust air from the actuator; the inlet phase generally has a large back pressure motivating inflation of the actuator, while the exhaust phase has only the decreasing pressure of the air contained within the actuator to motivate exhaust (Kingsley 2001). The added restoring force of the Festo actuator will thus decrease the exhaust time of the actuator, resulting in an increase in bandwidth. This does however result in a

disadvantage over the standard BPA design, in that the actuator is effectively inextensible. This means that both actuators on a joint must be cut to their maximum length, and when the joint is at rest both actuators will have some slack length. This resulted in a number of control issues and potential solutions, which are addressed in Chapter VII.

Festo fluidic muscle exhibits one additional significant disadvantage: the ends provided by the company are intended for high pressure industrial applications, where failure can be both dangerous and costly. Although the ends provided by Festo are "safe" they are quite bulky, not only adding an unjustifiable amount of weight to the device (which, among many other features is valued for its high power to weight ratio) but also seriously reducing the working length of the device, and in turn, the available stroke. Replacement of these ends became paramount for implementation of these actuators.

3.2 Replacement End Plugs

To provide a reliable seal a large compressive force between mating surfaces is required. This is normally accomplished through one of two means: heavily loaded threaded connections, such as between a piston and its head on an IC engine, or permanent deformation of a material (likely a metal) such as in a rivet. Although threaded connections have the advantage of being removable, they generally require more material and therefore are heavier and larger than those relying on material deformation for a clamping force. Many of the problems with the ends designed by Festo are attributable to their use of threaded connections. Because size and weight were of the utmost importance for this application, it was decided that relying on

material deformation to provide a seal was the most suitable choice. Furthermore, because only the outer portion of the plug would be required to undergo this deformation, the majority of the plug, which would reside inside the actuator, would be reusable in the event of a failure. Lastly, the polymer Delrin was chosen as the material from which the plugs would be made; this would provide a lightweight, easily manufacturable component.

The final design for new end plugs of the Festo actuators consisted of a simple cylindrical drum flanged on one end. The end opposite the flange was squared and contained a through hole for mounting. Under normal operating conditions of these actuators on a robot, the ends would be required to rotate about fixed axes so that they could maintain a linear connection between two limb segments rotating with respect to each other; designing the plugs to be mounted directly to these axes reduced the number of components that would be required for assembly, as well as reducing the overall length of the plug. The flanged end of the plug was ported for inlet of air to the actuator. To allow for easy delivery of air, a hole was drilled and tapped for connection with a quick disconnect pneumatic fitting. Because an inlet was only required at one end of the actuator, a reduced plug was designed that was not ported for placement on the opposite end of the actuator; this allowed a slight reduction in overall plug length (Figure 3.2). The ported ends of the plugs were not expected at restrict air flow, as they were 0.175 inches in diameter, compared to the 0.09 inch diameter pneumatic line in use throughout the robot, and the 0.065 inch diameter orifice in the valves.

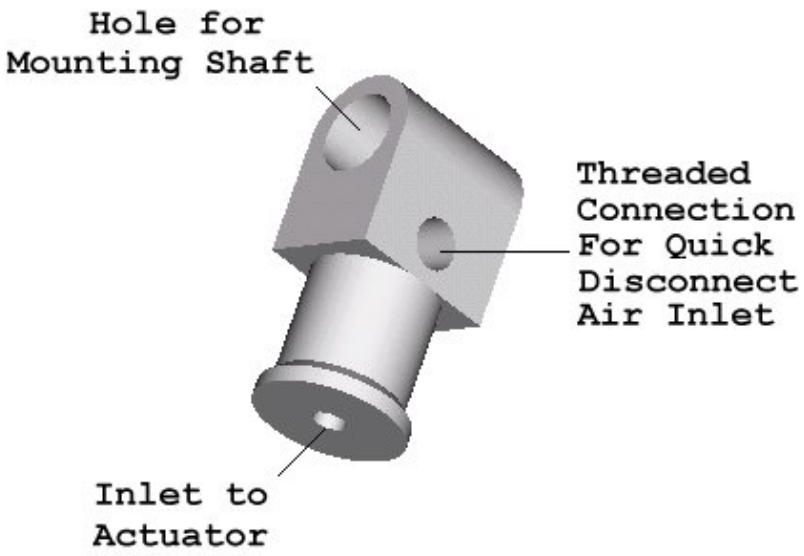


Figure 3.2: The inner portion of a replacement plug.

Assembly was completed by placing the plug inside the actuator, placing a step-less ear hose clamp (Figure 3.2) over the thinner portion of the plug and then compressing the clamp (Figure 3.3). Because the clamp keeps constant pressure over the entire circumference of the plug (thus, it is step-less) it provides a reliable seal. In the event of a plug failure, the clamp can be quickly and easily replaced by reinserting the plug in the actuator, and attaching a new clamp. Rates of plug failure varied widely in application; early actuator designs lasted approximately 100 cycles, those that were too short for their opposing mate—and thus were stretched to their maximum length during operation—would often fail within ten cycles due to the high operational loads they experienced.

Figure 3.2: Step-less hose clamp.

Figure 3.3: Close-up of an assembled replacement end plug.

Festo manufactures three sizes of fluidic muscle; two of which were used on Robot V. The plugs were initially designed for the smaller 10mm version, and then modified for the 20mm actuator. Because a total of 152 plugs were required for the robot, it was necessary to produce them in volume on a CNC machine. Racks of 40 10mm plugs or 16 20mm plugs were designed so that many plugs could be made at one time (figure 3.4 and 3.5).

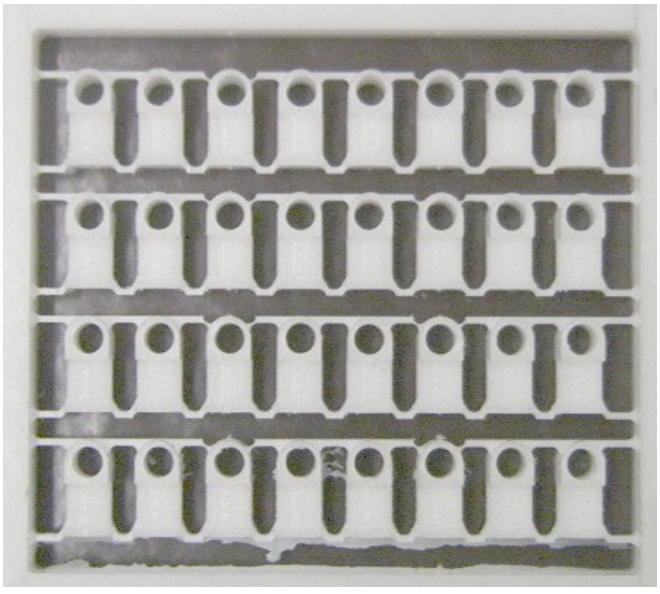


Figure 3.4: A rack of unported 10mm plugs.

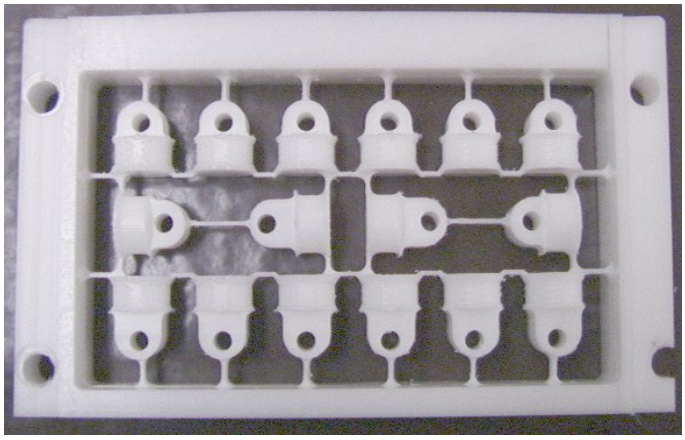


Figure 3.5: A rack of assorted 20mm plugs.

In practice, it was found that a layer of latex tube placed between the thinned portion of the plug and the actuator acted as an excellent gasket and also increased the material diameter, and thus the available clamping force. Initial trials showed a need for a slight increase in both plug and plug flange diameters. Although the plugs fit quite well into the actuators, the clamping force was at times insufficient, causing the plugs to either slowly work free or, more spectacularly, blow off (but they didn't go anywhere, as the mounting elements kept them firmly in place). This was due in part to diameter range limitations of the clamps—their design allowed only a finite amount of compression—but was easily overcome by increasing the diameters of both the plug drum and flange. Plugs that were thusly modified have performed excellently in service; most have exhibited no failures. Those failures that have occurred were again due to actuators of improper length, which were heavily loaded during operation even when at their maximum length.

Chapter IV: Manufacture and Assembly

4.1 Component Materials

Reducing the weight of the robot was of course of great interest, given that its predecessor was barely able to lift itself. One of the easiest ways to accomplish this was to use lighter materials in the construction of the robot. The most common engineering material is steel, and although it offers great strength, it is also quite heavy. Aluminum alloys have strengths comparable to some of the softer steels (ultimate strength of 1018 steel is 49.5 ksi, compared to 45 ksi for 6061-T6 aluminum) while weighing one third as much (Juvinall, Marshek, 1991). Although Aluminum does cost more than steel, the modest price increase was seen as a worthwhile trade-off for the reduced weight and easy machinability of Aluminum.

For the reasons cited above, the vast majority of the robot was manufactured from 6061-T6 aluminum; however, axles and actuator mounting shafts were made of 1018 steel, although in some cases alloy steel shoulder screws were used. Steel was chosen for these applications for two reasons. First, these elements were exposed to much greater loads than the structural elements, so strength was of the utmost importance. Second, these components were required to undergo a great deal of rotational motion. Aluminum is known to experience excessive surface wear in these circumstances (some refer to it being "gummy"), leading to increased joint friction and eventual part failure (Juvinall, Marshek, 1991). All axles were supported by nylon journal bearings which were press fit into one of the two elements which rotated upon the axel; the axel was rigidly attached to the other element. These

bearings were flanged, so that a layer of nylon would reduce wear not only between the axel and the structure, but also between the two elements of the joint.

4.2 CAD/CAM

As has been the standard for at least the last decade, the entire robot was designed using a computer aided design, or CAD, package; in this case, ProEngineer. This powerful parametric modeling program allows the design of components in a virtual three-dimensional space, making visualization of the complete design almost effortless. A number of supporting tools for this program further simplified the design process.

The completed robot design was imported to Interactive Product Animator (IPA) by Immersive Design. Although this was purely a kinematic modeling program, it did allow visualization of the robot's movements, as well as a means by which to determine if there were any interferences between components during operation.

The use of computer numeric controlled (CNC) manufacturing can decrease the time required for machining components, as well as allow much more complex parts to be made in a process known as computer aided manufacturing (CAM). A CNC machine (in the case of this project an EMCO 55 milling machine) functions much like a manual machine, but the axes are all driven by electric motors, and encoders provide exacting position feedback with a tolerance of one ten-thousandth of an inch. Contained within the ProEngineer suite of programs is ProManufacture, a program which, given a geometry and machining parameters (cutting rates, depth, etc…) can determine the necessary cutting paths to produce that component. These

cutting paths can then be processed into the necessary language for the CNC machine to produce the desired part. To decrease the number of different parts and programs that would be run on the EMCO, components of the robot were grouped together into sprues of parts of similar thickness (figure 4.1); for example, a number of parts half an inch thick could be grouped together onto a single sprue. Webbing was placed between the parts so that after the machine had finished cutting, they could easily be separated from each other. The end result was a number of plates—each containing various robot parts—that looked very much like a sprue of plastic pieces for a model airplane or car. Some additional processing of parts—such as drilling and tapping holes outside the cutting plane of the machine—was required, but this was minimal compared to the amount of work saved by having the CNC machine perform most of the manufacturing. To further simplify this process, an effort was made to design one-sided parts; although it is possible to flip a sprue over to machine both sides, there is invariably some positioning error in such a process, leading to not-quite-even components.

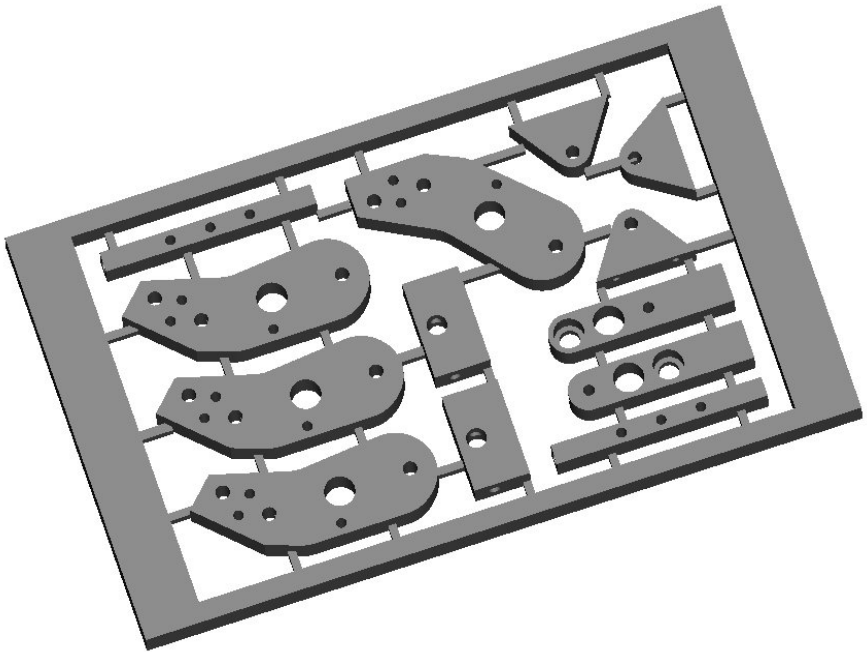


Figure 4.1: An example of a sprue of parts for Robot V.

4.3 Actuator Mounting

Actuators were not mounted during the initial assembly phase; instead the entire mechanical system was built as a series of freely moving joints. Once this was assembled it was then possible to manually move the joint through the desired range of motion, and thus determine the necessary actuator lengths to achieve this motion. This process was necessary because in many cases only the required length of the stance actuator was actually calculated for a joint, as not only was much less force demanded of the swing actuators, but they were also longer than the opposing stance actuator as a result of the structural design. Furthermore, even where desired actuator lengths were known, it was important to not only double check the design, but to ensure that the proper additional length was added to compensate for the length of the end plugs.

4.4 Valves

One of the most important properties of BPA's is their ability to control joint stiffness and absorb and release energy—effectively acting like springs. This is however an ability that can only be fully realized through the use of independent inlet and exhaust valves, which allows air to be trapped in the actuators. Furthermore, the speed of the robot is essentially limited by the rate at which air can be moved into and out of the actuators, which is a property greatly dependent on the response time and flow properties of the valves.

Previous robots developed by the CWRU Biorobotics lab implemented valves made by Matrix S.p.A. of Italy, and these same valves were used onboard Robot V. Inlet was provided by a series 850 nine channel (each valve block contained nine independent valves) 2-way normally closed valve (figure 4.2). These valves are basically the same as 3-way valves made by Matrix, but the company has modified them by closing the exhaust port, making the valve capable of only inletting and trapping air (Colbrunn, 2000). Air is supplied to the valve from a reservoir (1), and shutters are opened and closed by solenoids (coil) allowing air to pass into the actuators (2). An O-ring provides both seal against leakage and a restoring force to close the shutter. The manufacturer states that these valves have opening and closing times of 2ms and 5 ms, respectively.

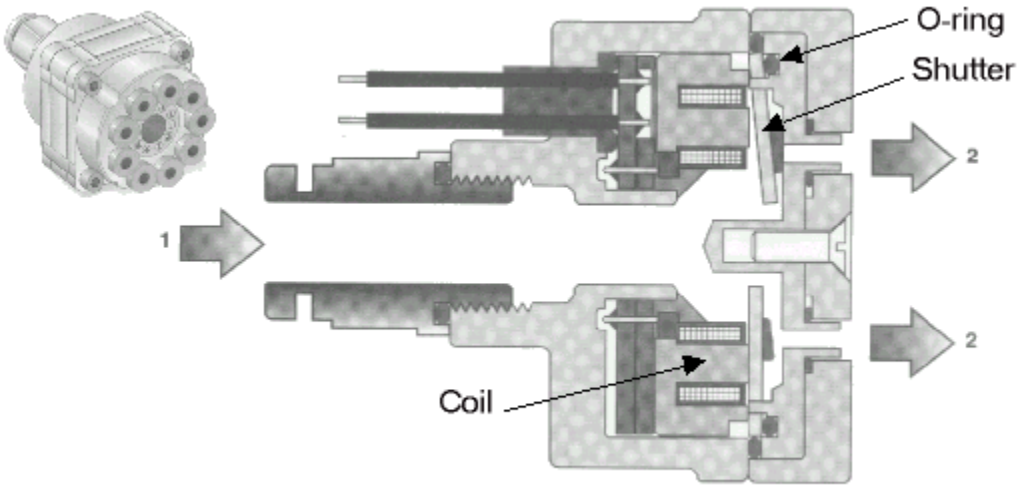


Figure 4.2: A schematic of a Matrix series 850 valve.

Air was exhausted from the actuators through a Matrix series 750 eight channel, 2-way normally closed valve. Problems were encountered on previous robots because these valves are intended to function as inlet valves, and are spring-loaded to provide a closing force against pressurized air. In this application however, the valves were being run in reverse, so that the high pressure air was on the opposite side of the shutter. This originally was remedied by Rob Colbrunn by disassembling the valve, removing the springs, and filling their cavities with epoxy (Colbrunn, 2000). To further reduce valve weight, the Aluminum end plates of these valves were replaced with ones manufactured from Delrin; these plates produced too many leaks however, and the original ones were kept in service. Air was supplied from the valves to the actuators through 5/32" nylon tubing, with connections provided by nylon push-in fittings for easy assembly and disassembly. To make air line identification easier, blue hose was connected to extensors, and gray hose was connected to flexors.

Chapter V: Initial Tests

5.1 Experimental Philosophy

Most mechanical devices are incredibly complex, and no amount of modeling will be able fully predict all aspects of the system. Such was the case with Robot V. Although most reasonable actions were taken to understand, model, and account for issues involved in the operation of the robot, many unknowns remained. It was therefore decided to move the development of Ajax forward in incremental steps, addressing problems and proving capabilities at each stage of development.

5.2 Stage One: Construction of the Middle Right Leg

Although the nature of the CNC machining process led to parts for different legs being simultaneously manufactured, an effort was made to focus on those parts required for the middle legs, specifically the right one (in reality, the components for the legs are the same, but the orientation of the workspace made it easiest to build and test a right leg). The middle legs were chosen to be the first built because they combine aspects of all three leg pairs—although weaker than the rear legs, they must still produce large stance forces, while maintaining a range of motion close to, but not as great, as the front legs.

This first leg (figure 5.1) served as a test bed for a variety of manufacturing and assembly processes, most importantly the CNC, assembly, and actuator mounting methods described elsewhere. This last was especially important, as this was a very new procedure for which there was no previous experience. Upon assembly, the leg was cycled through simple motions using an open loop controller developed by Gabe Nelson. This allowed the range of motion (ROM) of each joint to be measured.

Although the values attained were not as great as desired, they were still deemed to be in a range suitable for walking, and construction continued.

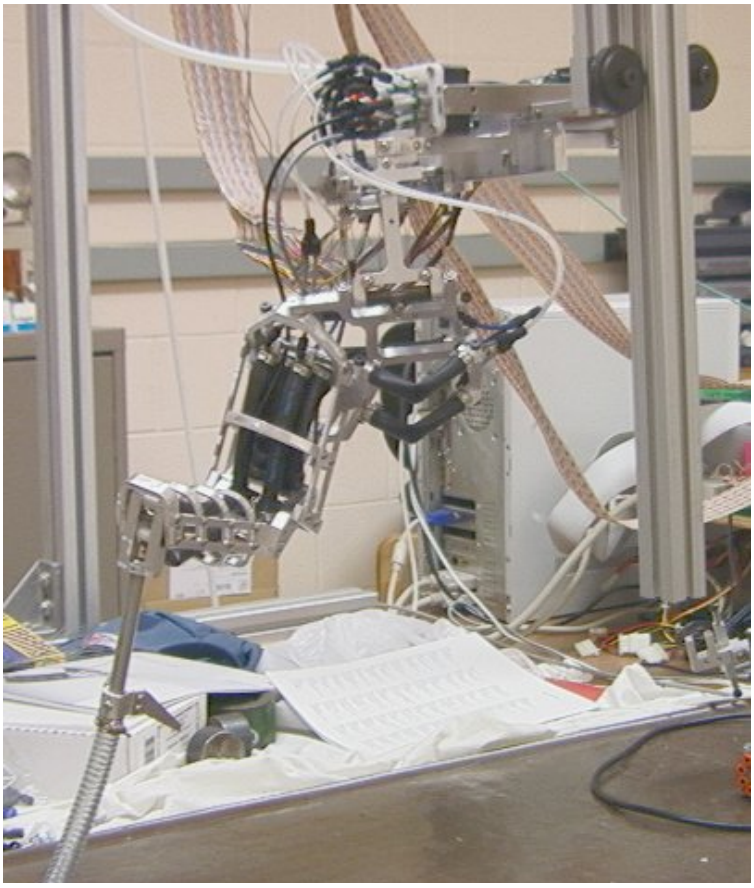


Figure 5.1: The middle right leg of Robot V.

5.3 Testing of Both Middle Legs

Although it was possible to measure the joint ROM's of a single leg easily, it was not possible to set up a simple rig for measuring the ground reaction forces—specifically the vertical forces—easily. This was mainly due to the fact that a single leg would produce large moments about its test stand, and as a result forces would be produced normal to the desired direction of motion. The addition of a second leg would produce lateral moments in the opposite direction as the first, providing a

balance. Thus, the middle left leg was constructed, and attached along with the middle right leg to a rig which allowed free motion both vertically and forward; in effect the legs were allowed to move along their line of force without interference from the rig, which only provided stability to keep the legs from tipping over (figure 5.2). Using an open-loop controller, the legs were made to perform "push-ups" lifting their weight as well as the weight of the rig, which included valves for the actuators. Although the motion was not exactly the same as that of the animal during locomotion—the beta joints were flexed about 15 degrees too far—the difference was acceptable, as the main goal of this was simply to demonstrate that the legs were capable of supporting large loads. As these trials were successful, additional trials were conducted adding five (figure 5.3) and ten pound weights to the legs and again performing push-ups. The legs easily lifted the additional five pound weight to a full stance position, but were not able to lift the ten pound weight quite as high, falling about one inch short of full stance. This series of tests was considered quite encouraging, and the two legs together weighed only six pounds, and the rig added another five pounds of weight, beyond what was added in the later trials. This suggested that the robot would indeed be capable of standing and walking even under the weight of its valves and some payload.

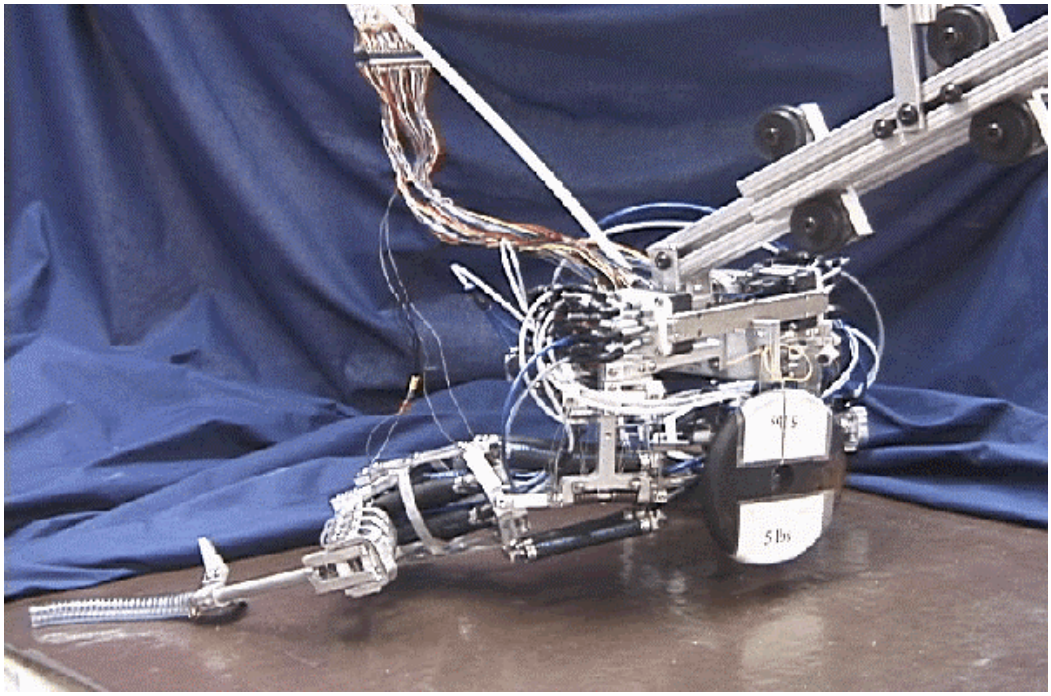


Figure 5.2: The middle legs of Ajax prepare to perform a push-up.

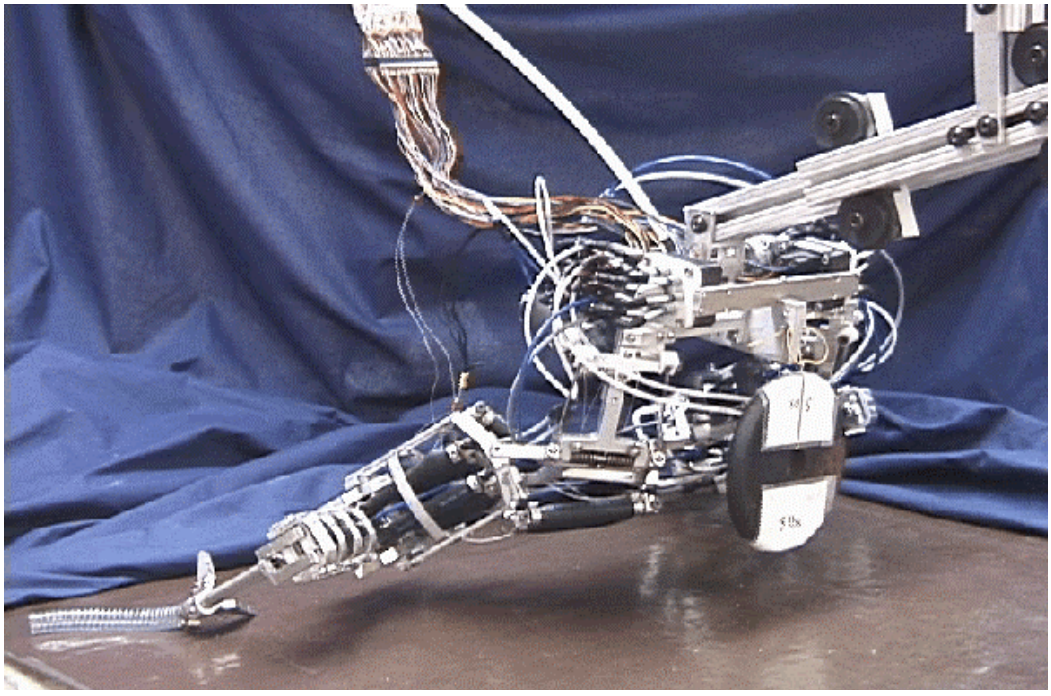


Figure 5.3: The middle legs successfully lift themselves, a rig, and a five-pound weight.

5.4 Assembly and Testing of the Entire Robot

The tests performed on the middle legs showed that they had nearly the intended range of motion, and seemed capable of generating forces sufficient for achieving stance. Although it would have been easily possible to conduct similar tests on the front and rear legs, it was deemed more useful to complete the entire robot and study the system as a whole. The remaining four legs were constructed and assembled onto the robot. Mounts were made to attach all six inlet and six exhaust valves. These were mounted toward the rear of the robot in an attempt to move the center of mass closer to the rear legs; the center of mass is located between the middle and rear legs on the animal, and it was desired to keep it in the same location on the robot. Fully assembled, the robot's center of mass was 13 inches rearward of front leg mounts; 6 inches behind the middle leg mounts and 4 inches in front of the front leg mounts. This is approximately where the CoM of the insect is located.

The first and simplest test to be performed on the robot in this state was a simple demonstration of the stance bias of the legs. Although the front legs did not have torsion springs in any of their joints, and were thus not expected to (and did not) support any static weight, it was clear that the middle and rear legs did lift themselves off the ground without any pressure in the actuators (figure 5.4). Although the robot did not achieve the full stance position in this manner—it was not designed to—it did clearly show that the torsion springs would greatly reduce the load requirements placed on the stance actuators.

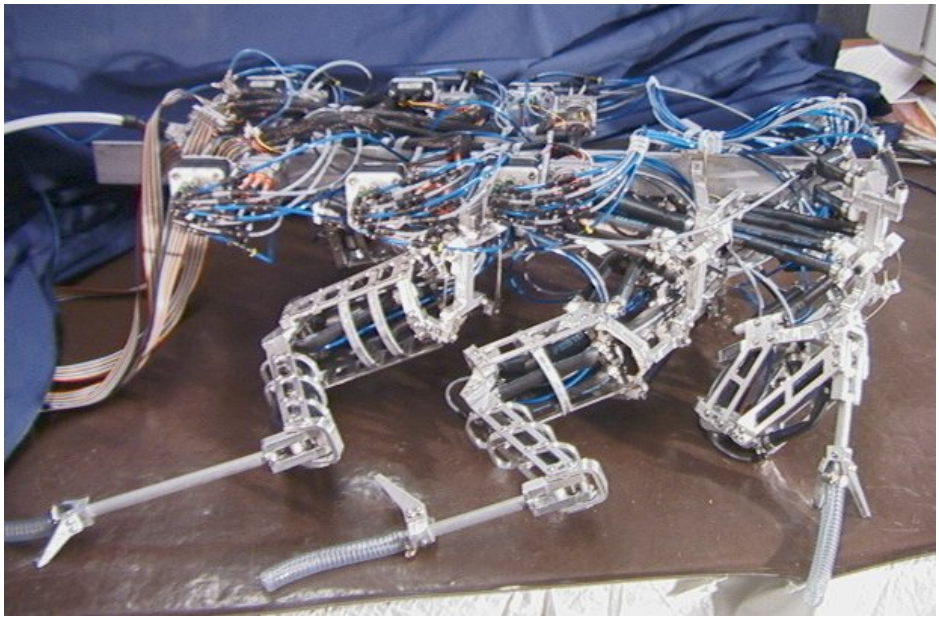


Figure 5.4: Ajax maintains a near stance position without pressurization of actuators.

Using an updated open-loop controller written by Gabe Nelson, the joints of each leg were then made to move through their full range of motion, and these values were measured by hand and recorded as:

Table 5.1: Measured and desired ROM.

JOINT	ROM	DESIRED ROM
Front Leg		
γ	-15 - 5°	-20 - 20°
β	-85 - -45*	- 90 - -60
α	40 - 55	35 - 65
c-f	-5 - 35	0 - 40
f-t	-35 - 40	-30 - 40
Middle Leg		
β	-50 - -70	-90 - -60
α	40 - 55	20 - 60
c-f	0 - 40	0 - 50
f-t	-35 - 40	-35 - 40
Rear Leg		
β	-46 - -25	-40 - 0
c-f	-10 - 40	0 - 50
f-t	-40 - 40	-35 - 40

*The range of motion of the front β joint was dependant on the position of the γ joint

The desired ranges of motion were based on those used on Robot III (Bachmann 1997) which in turn were derived from motion studies of cockroaches walking (Watson and Ritzmann 1998, Watson et al. 2002)

Although in most cases the desired range of motion was not attained, these target values had been chosen in excess of those displayed by the animal, and the achieved ROMs were assumed sufficient for locomotion, so testing was carried on to the next phase.

Again utilizing a simple open-loop controller, the stance actuators were pressurized, in an attempt to move the robot into a standing position. There were some initial difficulties as the rear actuators, specifically the rear β actuators, were

significantly overpowered, and when fully pressurized caused the entire body to pitch forward, effectively driving the front of the robot into the ground. This problem was easily remedied by reducing the allotted inflation time for these actuators, resulting in them not reaching full pressure. Careful adjustment in this parameter eventually resulted in limb orientations closely approximating that of the animal during stance (figure 5.5). It should be noted that the problems encountered with the rear β actuators were met with optimism, as this suggested that the robot would be able to carry some additional payload; in fact, by placing payloads behind the rear legs, it would be possible to reduce the forces required from the weaker front and middle legs while increasing the loads on the stronger rear legs.

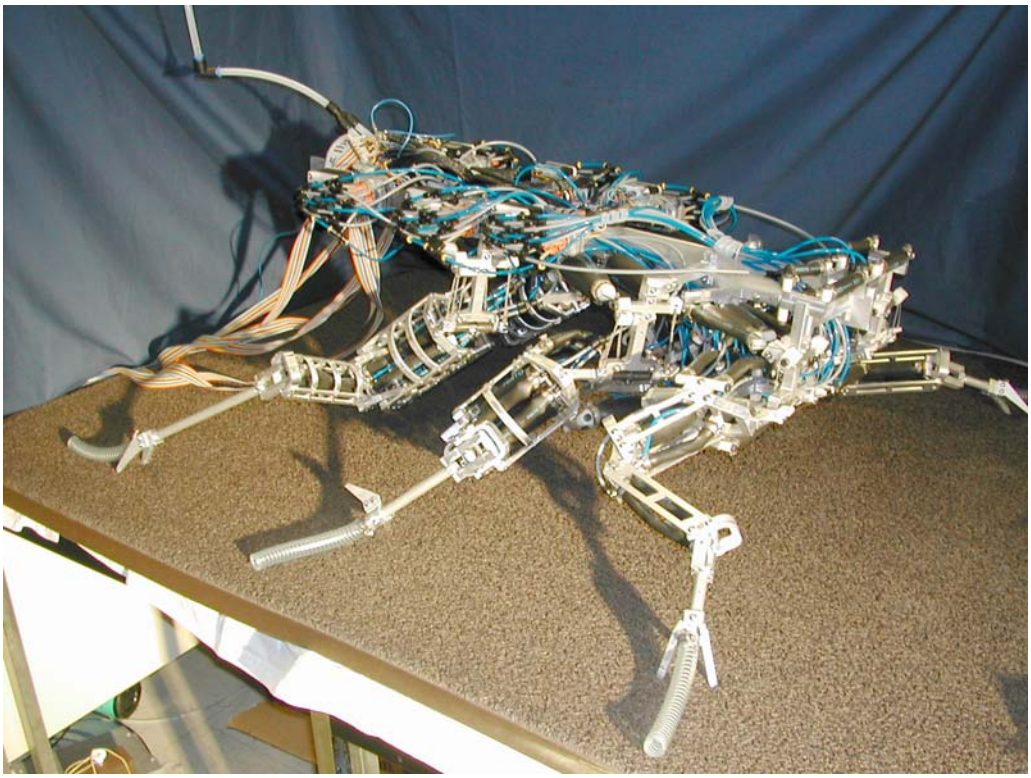


Figure 5.5: Ajax stands using an open loop controller.

Biological systems behave very differently from most mechanical ones, and one of the contributing factors to this are the differences in properties between muscle most actuation devices. Muscle responds very rapidly to sudden kinematic changes, while at the same time they respond sluggishly to changes in neuronal activation (Loeb et al. 1999). This ability to rapidly respond to perturbations is believed to be driven not by any control system, but by the physical properties of muscle itself, and has thus been termed "preflexes" (Jindrich and Full, 2002). One of the many important properties of BPA's is their passively stable nature; one can think of them as simply being springs, with stiffness that varies with pressure and length. When one stretches a spring and then releases it, the spring of course returns to its equilibrium position. This property carries over to the robot as a whole; when the actuators are pressurized, they will move the joint that they drive to some equilibrium position. When disturbed, this passive property will force the system to return to its default position. The result is that after the robot was made to stand, it could be perturbed (hit with a stick) and maintain its posture without the use of a controller. This occurs in much the same way as in animals, which are believed to not need active control to deal with small perturbations (Loeb et al. 1999). Many other robots, such as Robot III, have demonstrated active stability (Nelson 2002), using a controller to maintain posture (in fact, this is necessary for locomotion). It is believed that these passive properties that arise from the actuator and valve configuration on Robot V will not only provide a level of passive stability in response to perturbations of the entire robot, but will also reduce instability of joints that results from the sudden load changes during initiation and termination of swing.

Standing, passively stable robots are interesting to a point, but the goal of this project was to produce a walking robot, so attempts were made to make the robot walk. This first stage of this was to demonstrate that Ajax could support its weight in a tripod stance where the front and rear legs on one side and the middle leg on the opposite side were lifted off the ground. This of course means that the robot has half as many legs to support its weight, and thus each joint experiences much greater loads. Again utilizing the open loop controller, the robot was made to perform this task (figure 5.6), and in addition, to switch from tripod to tripod, something that would eventually be required for walking. This same task was repeated, but with an additional 5lb weight added to the rear of the robot (not only did this demonstrate some payload capacity, but it moved the center of mass toward the rear of the robot, and eliminated some of the pitching forward visible in the figure.

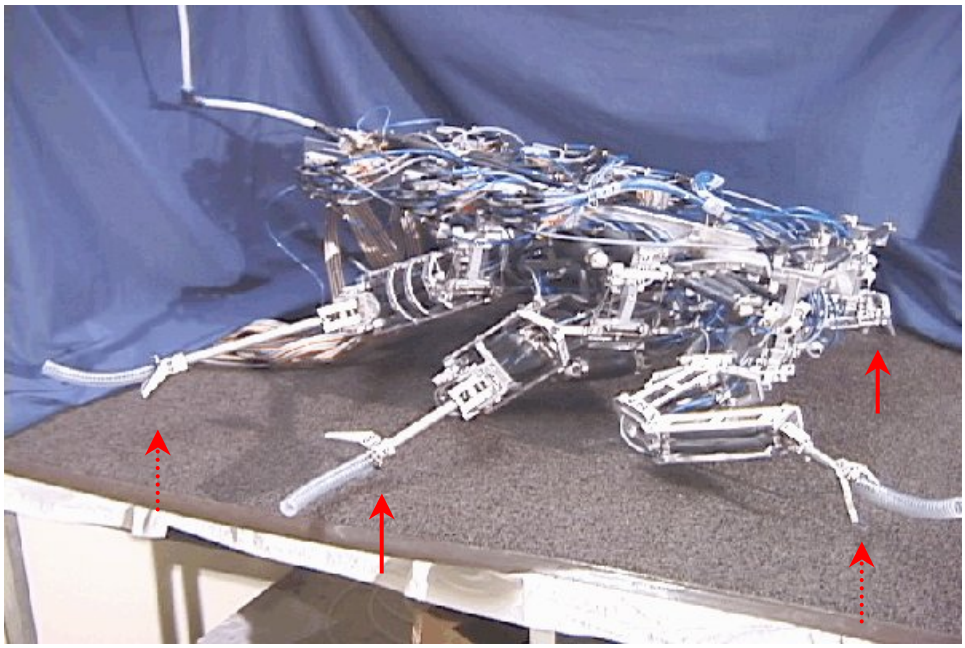


Figure 5.6: Robot V achieves a tripod stance. Solid arrows indicate feet contacting the ground, dashed arrows indicate legs in the air.

The final stage of open loop testing was to make the robot walk. It should be noted that open loop walking is for all intents and purposes impossible for most legged walking machines; and it is easy to understand why, for this is akin to a blind, deaf person with no sense of touch, or even the ability to tell where their joints are positioned or how much their muscles are being activated, trying to walk. However, using the feed forward controller, Ajax was able to produce rudimentary walking motions while carrying a 5 lb. payload. Although this was by no means the agile cockroach-like walking that was the ultimate goal of this project, it was a clear demonstration of not only the robot's capabilities, but also the advantages offered by the BPA's coupled with trapping air in them. The ability to move using only an open loop controller was in large part a result of the passive properties of the actuators, which provided compensation for any instabilities in the controller itself and immediate response to perturbations without the need for controller intervention. This can be contrasted with Robot III, which, even with kinematic and force feedback, was not able to walk. This failure of Robot III was attributed to its 3-way valves that did not permit air to be trapped in its pneumatic cylinders and its posture controller's inability to deal with the sudden changes in load associated with locomotion (Nelson 2002). In short, the BPA's acted as filters, providing immediate response to perturbation; a task a slow centralized controller is incapable of. This same process occurs in biological muscle, which responds nearly instantaneously to perturbation, but only slowly to neurological input (Loeb et al. 1999).

Chapter VI: Modifications To Robot V Design

6.1 Impetus For Modifications

Completion of Robot V allowed for testing and observation of its capabilities. Of greatest concern were the ranges of motion (ROM) and available moment of each joint, as these two factors were most important for achieving stance and cockroach-like walking. Tests of these properties were carried out through the use of a simple open-loop controller written by Gabe Nelson. This allowed the duty cycles of the valves controlling inlet and exhaust to be established for a set series of motions.

6.2 Changes In Joint ROM

Initial tests of the middle legs showed that although the beta joints moved through a satisfactory angle, they did not move through the desired angles—that is, the motion reached too near the body, but did not allow the desired amount of extension. This was easily remedied by lengthening the tendon attached to the flexors and shortening the tendon attached to the extensors; their range of motion was shifted to being between -90 and -60 degrees.

The front gamma, front alpha and middle alpha joints all failed to provide the desired range of motion. In all three cases, this was due to the actuators themselves being too long. Although longer actuators do have a greater stroke, the increased stroke is in the same proportion as the stroke/length ratio of the rest of the actuator. For example, a four-inch BPA has approximately one inch of stroke. An additional inch of length adds only one quarter an inch of stroke. The result of this is simple: an actuator one inch too long gains one quarter of an inch of stroke and three-quarters of an inch of slack that must be taken up before the actuator actually begins moving the

joint. To attain satisfactory ROMs for the aforementioned joints, the front gamma extensor and font alpha flexors were shortened by half an inch, and the middle alpha flexors were shortened by one quarter of an inch. As a result, the front gamma ROM became -15-20°, the front and middle alpha both were improved to 30-55°

6.3 Alpha Joint Strength Enhancements

Observation of early attempts at standing and tripod stance showed that the middle and front alpha flexors were not capable of supplying the necessary force to maintain these positions; this was most apparent during the higher loading states in tripod stance. Fortunately, a simple solution presented itself. One of the two 10mm actuators initially mounted at each joint was removed and replaced with a 20mm actuator (note that doubling the actuator diameter quadruples the available actuator force). Although the design did not allow enough space for both actuators at each joint to be so replaced, further tests showed that only one larger actuator was necessary to achieve and maintain tripod stance.

6.4 Redesign of Coxa-Femur Joints

Although the initial design of the coxa-femur joint on all three leg types was serviceable, it did not allow for easy replacement of actuators which was important not only for modifying actuators for ROM adjustments, but also to repair failed actuator plugs while those devices were still being tested. This was because the actuators were mounted on stepped shafts that, while preventing the elements of the joint from moving too close together, were not removable without disassembly of the entire joint (figure 6.1).

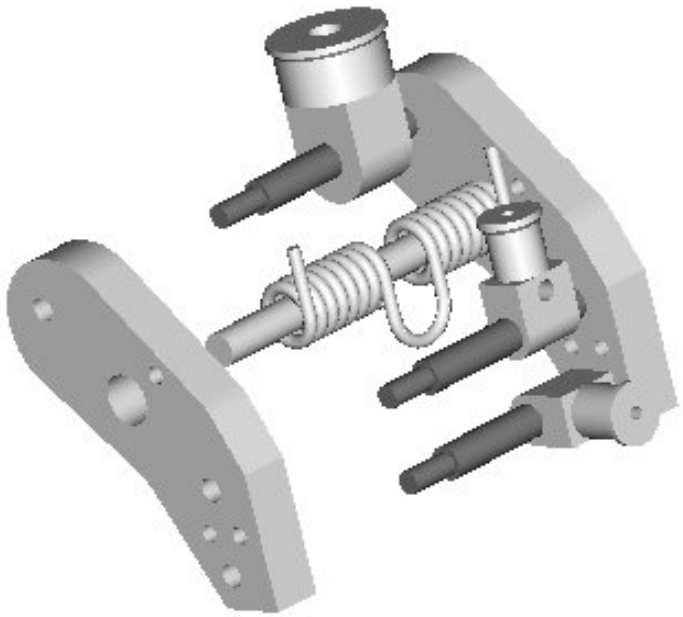


Figure 6.1: Exploded view of the original C-F joint.

The stepped shafts were replaced by shafts with internal threads; although these did not maintain separation between the main elements of the joint, this task was already being accomplished by the torsion spring mounted inside the joint as well as the ribs mounted in the femur. The new shafts were held in place by screws passing through the main joint element and threaded directly into the shaft. To prevent interference between this inner portion of the joint and the outer half, it was necessary to chamfer the mounting holes of the shafts to allow the mounting screw to sit below the surface of the main joint element (figure 6.2).

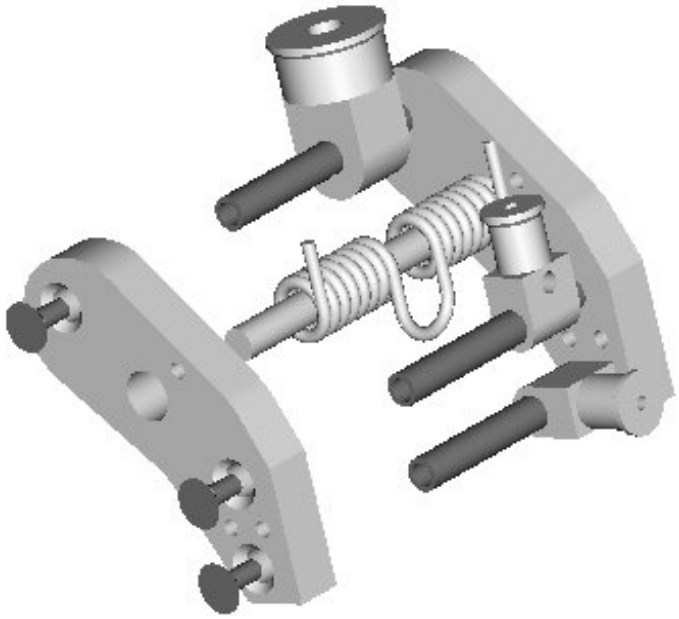


Figure 6.2: Exploded view of the new C-F joint.

6.5 Repositioning of the Rear Legs

During operation, it was noted that there was a slight interference between the body-coxa β joints of the rear legs and the coxa-femur joints of the middle legs. Not only did this cause a small, but undesirable, amount of wear between the parts involved, but it also occasionally caused the middle leg to snag on the rear, preventing it from extending as quickly as it should. This problem was easily remedied by moving the mounting points of the rear legs back three-quarters of an inch. Not only did this alleviate the interference problem, but is also served to move the center of mass of the entire robot more toward the rear legs, which was beneficial as the rear legs are designed to bear much more weight than any of the others. Although the original positioning had been at the same ratios as in the animal, the small (7.5%) difference was acceptable in lieu of the advantages gained.

6.6 Middle and Rear Leg Deflection

The middle and rear legs of the robot were all attached to the body with a single armature, which was held in place via two screws. These screws were placed above and below the armature, so that they could better support the vertical forces and moments about the robot's x-axis produced during standing and walking. An issue arose with this design as a result of the β-actuator mounting position. These actuators were placed horizontally, and deliver torque to the joint via a tendon that wrapped around posts attached to the leg armature. The β extensors in particular were very powerful, consisting of a pair of 20mm actuators, and thus producing a large bending moment on the mounting armature; although no exact measurements were made, it was estimated that the armatures bent approximately 10° when the β extensors were fully inflated. Not only did this have the potential of causing damage to the structure of the robot, but the deflection of the armature also served to shorten the distance between the actuators and tendon posts, increasing tendon slack. To reduce these bending effects, supporting struts were added diagonally between the armatures and the body (figure 6.3). Note that this problem was not apparent on the front legs because of both the inclusion of the γ joints and the use of smaller, 10mm actuators on the front β extensors. This design did however result in the previously mentioned tenodesis issues.

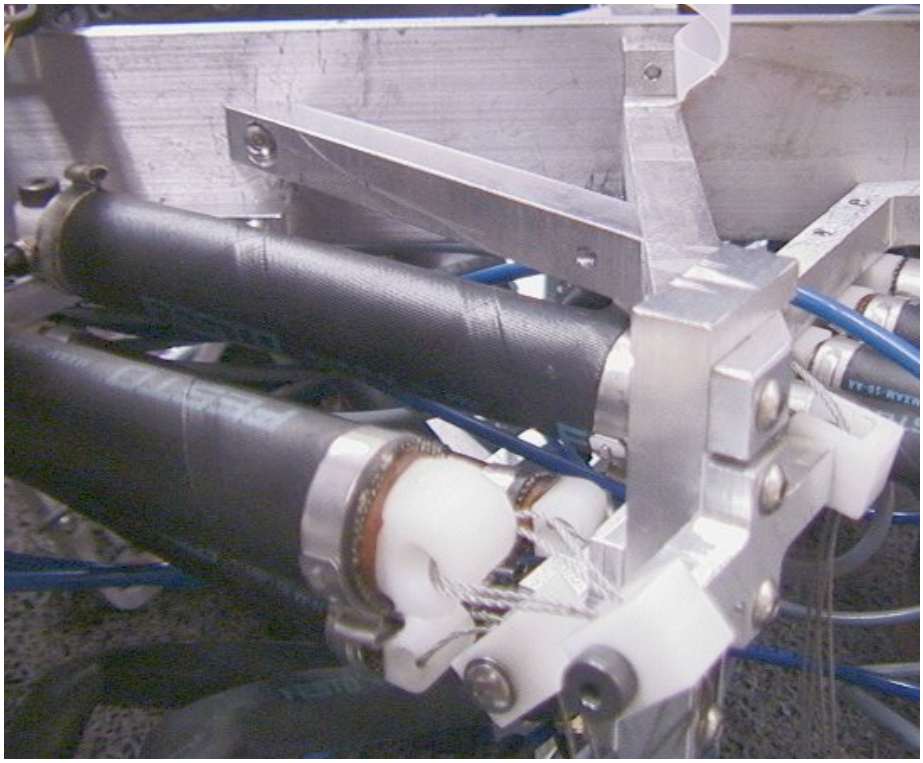


Figure 6.3: β actuators and supporting spar on the middle right leg.

6.7 Tibia Mount Modifications

The same element was initially used on all six legs to attach the hollow Aluminum tube that served as the tibia of each leg; however, the loading conditions experienced by each tibia were quite different. Most significantly, the rear leg tibias were much longer and oriented closer to parallel with the body than the tibias of the front and middle legs. This resulted in much larger moments being placed on the mounting element, and eventual loosening of the tibia. This problem was remedied by making a longer mounting piece, with a slightly undersized hole into which the tibia was press fit. In addition, the original mounting element allowed for a single bolt to pass through the tibia; the new element allowed two screws to be used to secure the part (figure 6.4).

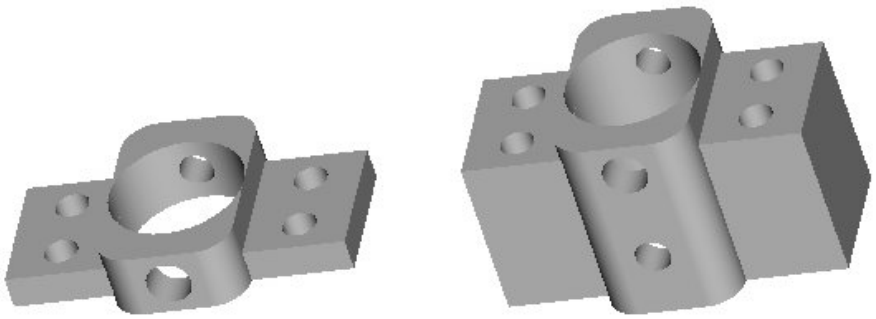


Figure 6.5: The old (left) and new (right) rear leg tibia mounts.

6.8 Redesign of Beta-Gamma Interface

Perhaps the most difficult portions of this design were the body-coxa interfaces of the front legs. This three degree of freedom connection presented numerous problems, specifically because of the way in which the beta-gamma interface was designed. Because the tendons for the beta actuators were wrapped around members of the gamma joint, it was possible for the state of one joint to influence the other. This problem was compounded by an error in the design of the main element immediately distal to the gamma joint (figure 6.6, left) which held the mounting points for the pulleys of the beta tendons. The design flaw was that the axis for the gamma joint (marked "A" in the figure) was not aligned with the shaft of the part, on which the mounts for the beta tendon pulleys were mounted (marked "B" in the figure). As a result, the tendons for these actuators had uneven moment arms on the left and right side of the part, leading to two problems. First, as the gamma joint was rotated, the necessary length of tendon to maintain tension changed, resulting in varying ranges of motion for the beta joint depending on the orientation of the gamma

joint. Secondly, when the beta actuators were pressurized, this uneven moment arm produced a net torque on the component, causing it to rotate about the gamma axis.

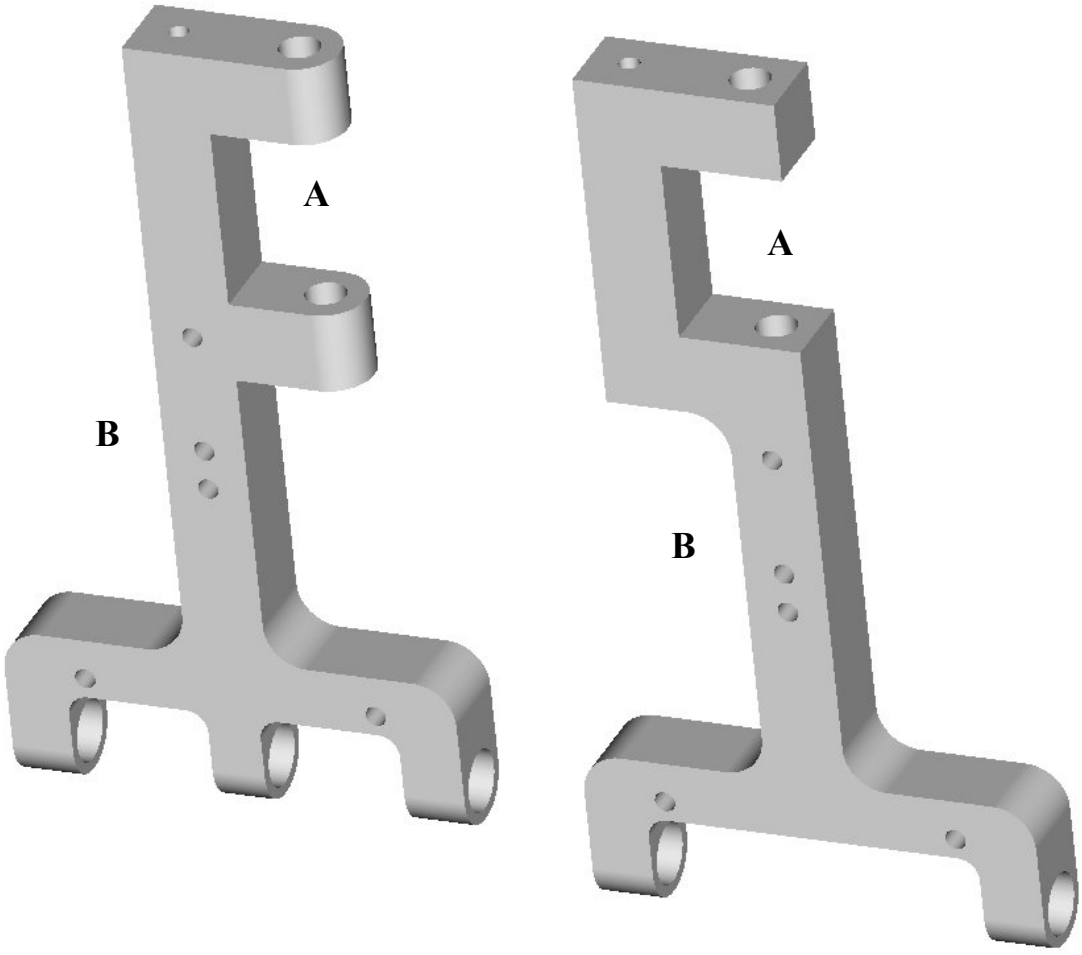


Figure 6.6: Original (left) and modified (right) front left armature. The gamma axis passed through hole A, and pulley attachments were mounted at the three holes marked B.

These issues were addressed by redesigning the part (figure 6.6 right) so that the mounting points for the beta pulleys were aligned with the axis of the gamma joint. Implementation of this change resulted in a dramatic improvement of not only range of motion of the beta joint, but also near elimination of the cross-talk issues between the beta and gamma joints. However, the cross-talk problems were not

completely eliminated, and in specific, it was found that extreme beta positions severely limited gamma range of motion; this was due to the high tensions exerted by the tendons on their pulleys. At full gamma extension, the beta range of motion was restricted from -85 to -65° and at full gamma flexion, the beta range of motion was restricted from -45 to -55°.

6.9 Redesign of Beta-Gamma Interface Redux

Ultimately, it was decided that although the aforementioned modifications significantly improved the front leg beta-gamma interface, they did not result in the necessary ranges of motion for walking. More dramatic changes were needed; specifically the complete uncoupling of these two joints. This had not been done in the initial design stages because of concerns of actuator packaging; that is, actuators of sufficient strength for actuation of the beta joint would significantly raise the profile of the robot and interfere with sensor and valve placement, especially on the middle and rear legs, and potentially interfere with operation of the gamma joint on the front legs. Given the limitations of the current design however, redesign of the front leg beta joints became necessary. These legs were the most viable for this modification, as the front leg beta joint did not require torques as high as the others, and thus would retain functionality with shorter and smaller diameter actuators. In addition, because no structures were placed forward of the front legs, there was ample room for actuator mounts without the fear of interfering with other components of the robot.

Actuator mounting elements were added to the modified armatures detailed in the previous section using the same attachment points used by the pulley mounts in

the original design. Likewise, the tendon attachment points in the main coxa element were used to attach shafts that would serve as the insertion points of the new actuators (figure 6.7). Care had to be taken to ensure that the beta extensors did not interfere with the gamma actuators or sensors.

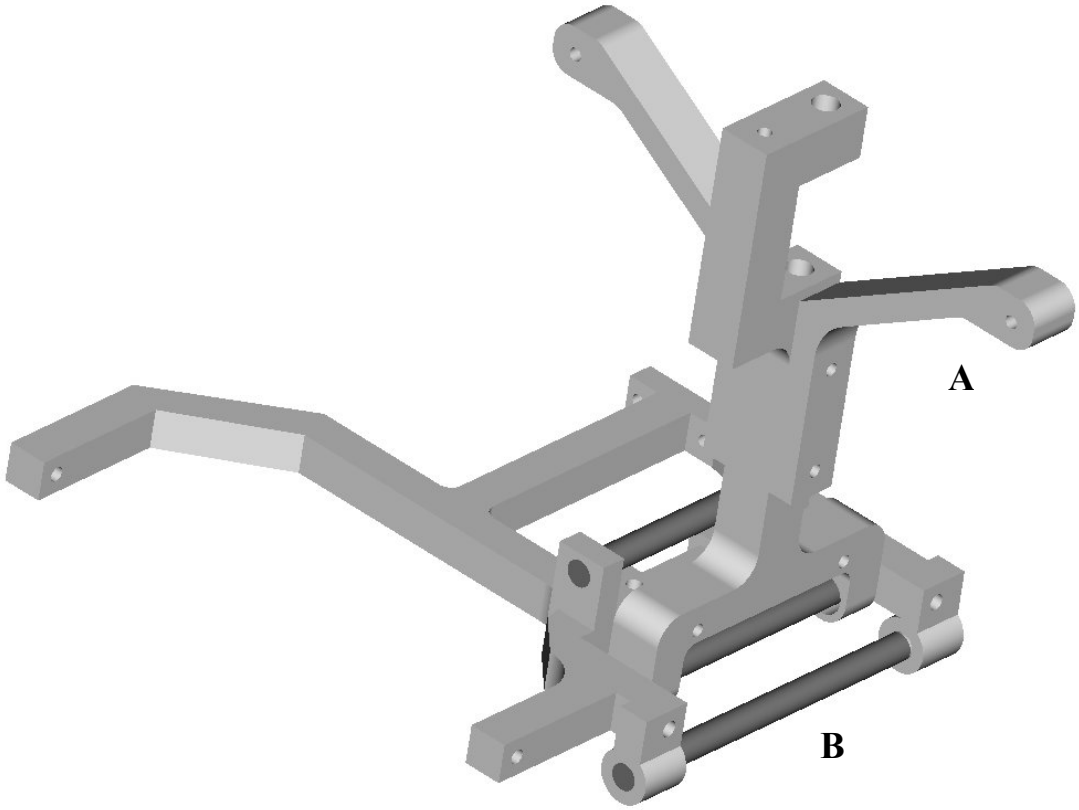


Figure 6.7: The redesigned gamma-beta interface in the front legs. Actuator origin is at point A, and insertion is at point B.

This modification (figure 6.8) yielded exceptional results, not only producing a complete decoupling of the joints in question, but also increasing the range of motion of the beta joint. Testing after completion of these modifications showed the gamma joint to be capable of motion from -7 to 24°, while the beta joint was capable of motion from -32 to -73°.

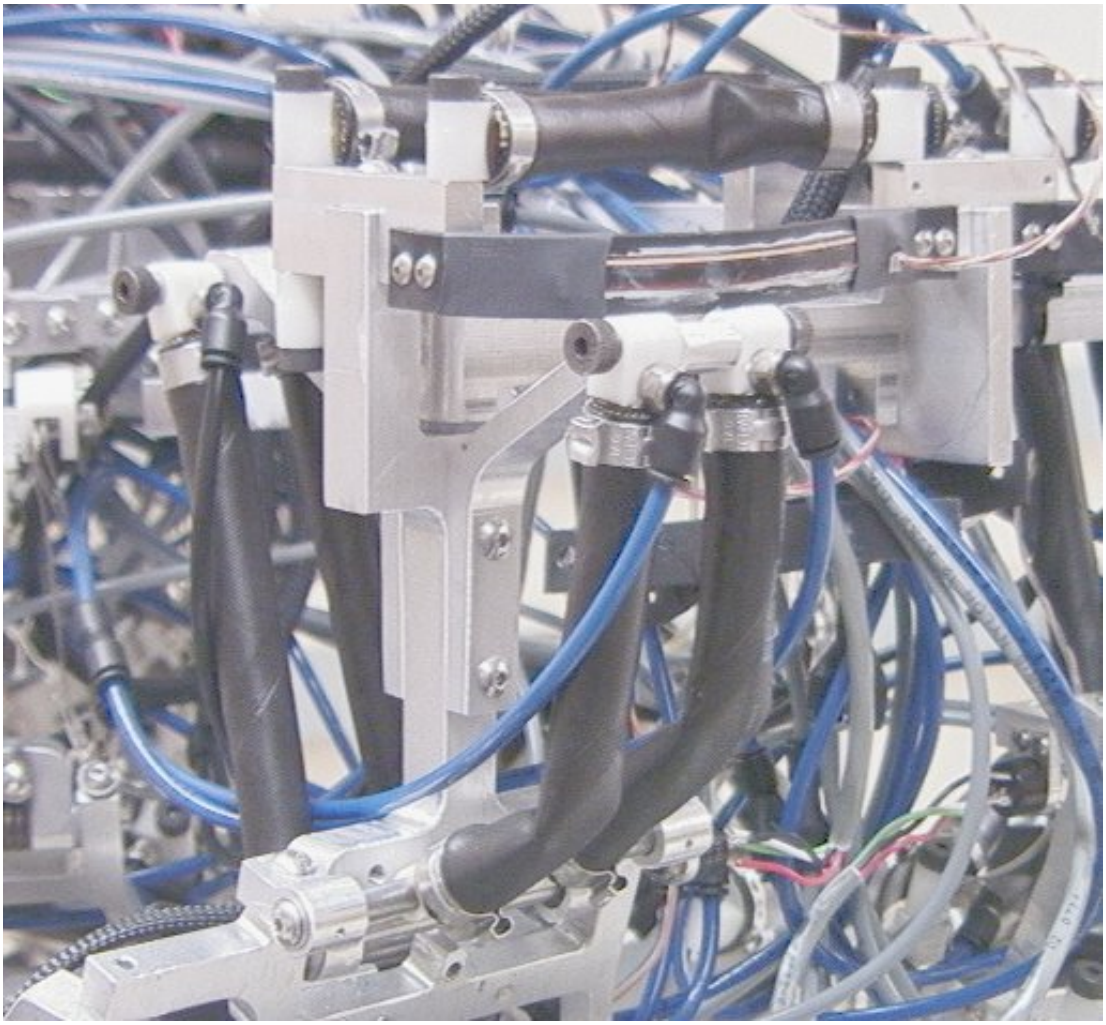


Figure 6.8: Completed front leg beta joint modification

6.10 Middle Leg Beta Modifications

In light of the improvements resulting from the removal of tendons from the front legs, and the limited range of motion of the middle leg, it was decided that the same modification would be made to the middle leg β assemblies (the rear legs actually had a suitable range of motion, and packaging of the valves and electronics precluded the possibility of changing orientation of the rear β actuators). These changes were made in much the same way as on the front legs; in fact, most of the

components were of the same design. The most significant difference was that due to the use of paired 20mm actuators for the extensors of this joint, it was necessary to add a structural member to reinforce the armatures holding these actuators (figure 6.9). As with the front legs, implementation of this modification resulted in a significant improvement in joint ROM, which was measured to be -40 to -85° upon completion of these changes.

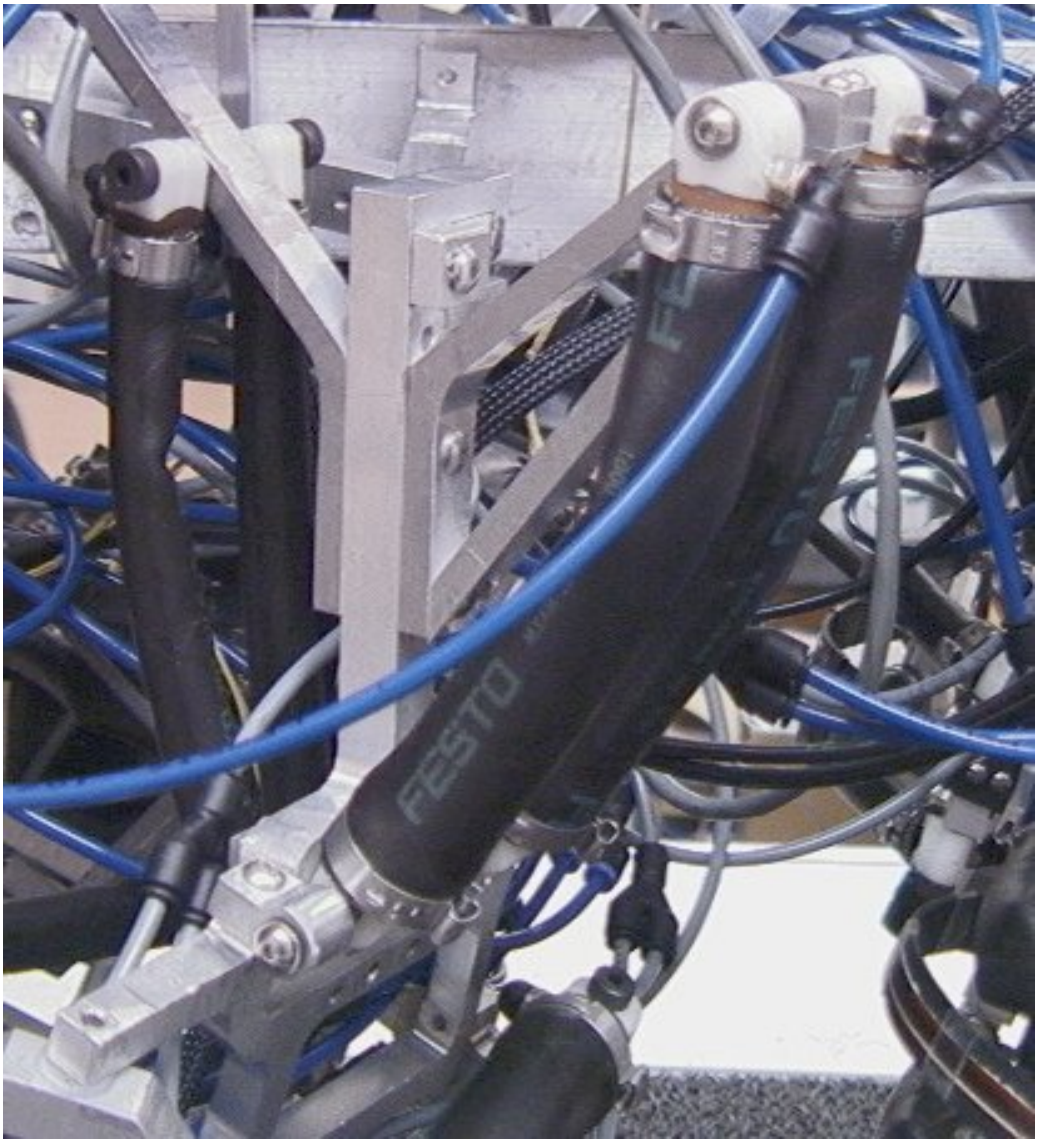


Figure 6.9: Completed middle leg beta joint modification.

6.11 Front Leg Tibia Adjustment

During gantry supported walking tests in the early stages of closed loop walking tests, it became apparent that the front legs were not able to reach as far below the body as the other legs, resulting either in the body pitching downward, or the font legs being unable to contact the ground. This was due to the fact that during the walking trials, the desired stance of the robot was higher than that of the animal; as previously discussed, this was done to prevent dragging of the coxa-femur joints on the ground. This issue was resolved by lengthening the aluminum tube elements of these leg segments from 3.25 to 5 inches.

Chapter VII: Addition of Tendon Elements to a BPA Driven System

7.1 A Limitation of the RV Design Philosophy

One of the earliest design decisions that was made during the development of Robot V was, whenever possible, to avoid the use of tendons to transmit power and motion from the actuators to the joints. Although the reasoning that motivated this decision is presented in more depth in Chapter II, to put it quite simply, reduction of actuator length to allow for tendons reduced the available stroke of the actuator, resulting in a loss of either range of motion, available moment, or some combination of the two. Because these design envelopes were so limited, inclusion of a tendon element was deemed too restrictive to the performance of the design. This same methodology, while possibly not put into words, was visible on previous robots. With the shift from in-house BPAs to the Festo BPAs however, a problem quickly became apparent. Most braided pneumatic actuators are made to have a certain amount of stretch—the actuator is not extended to its full length while at rest—which provides a level of compliance. Muscle fibers can generally be stretched beyond their rest position (Zajac 1989); this is in fact one of the many desirable muscle-like properties of the BPA. Festo BPAs however, are manufactured to be fully extended at their rest length (a result of the angle of the weave when bonded to the rubber) and thus exhibit only a minimal amount of extension when loaded beyond their unloaded length. The practical result of this is that, because Festo BPAs must be made to their maximum desired length, when an opposing pair of actuators are installed on a joint one or both sets of actuators will buckle (figure 7.1). Not only is this potentially damaging to the actuators, but it presents problems for a control system as the

transition from buckled to unbuckled length presents a discontinuity in the apparent actuator length. In fact, as buckled actuators were inflated they at times initially caused motion in the opposite direction as desired, as they extended to full length before beginning to expand radially.

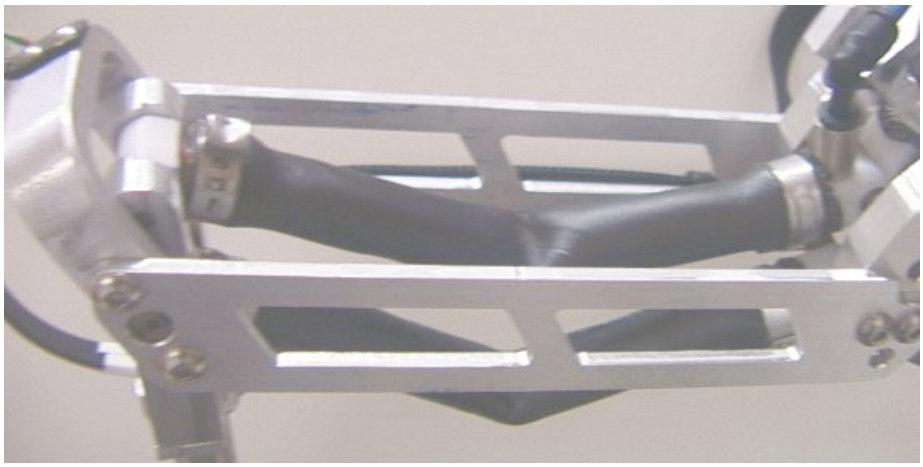


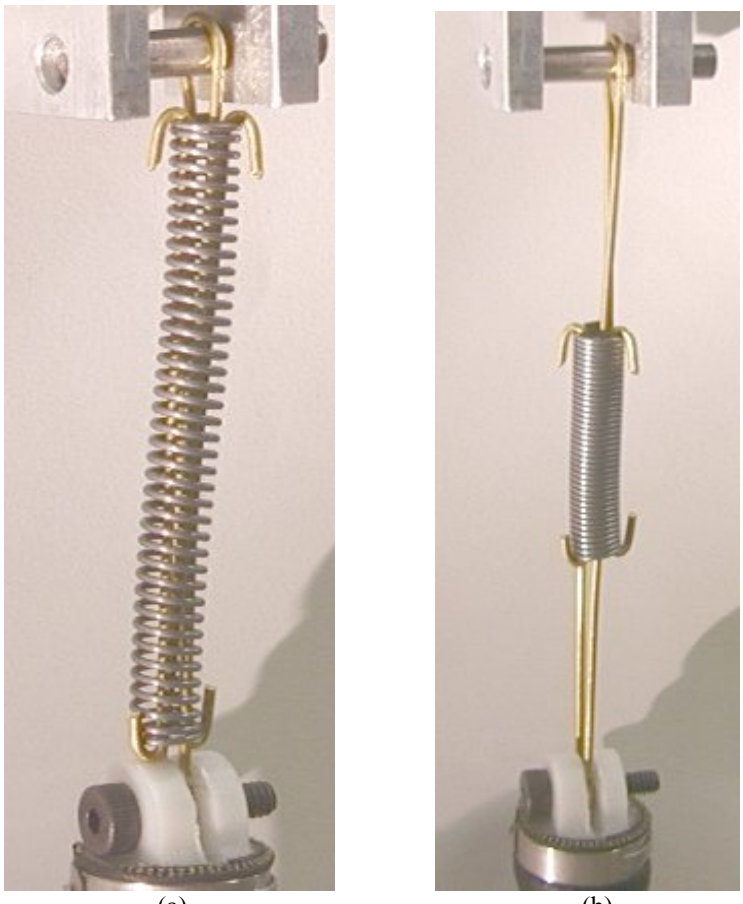
Figure 7.1: Buckling of actuators in the front right tibia.

7.2 Theoretical BPA Tendon Design

The most viable solution to the above problem is to add a "tendon" element into the joint mechanics. In reality, this tendon will function in just the opposite way that its biological counterpart does. Biological tendon normally has a very high stiffness with respect to a relatively compliant muscle; in contrast, muscle can often be modeled as two separate elements: a contractile element, which causes the muscle to shorten, and a series elastic element, which allows the muscle to be stretched. This mechanical system utilizes actuators with a very high stiffness, and a "tendon" with a low stiffness is needed to allow both actuators to maintain an unbuckled state when they are not pressurized, effectively acting as the series elastic element of the muscle. Many of the above buckling issues could quite easily be solved by removing

one actuator and replacing it with a spring; however, it would then not be possible to control joint stiffness.

A simple device capable of meeting all of the above requirements is a compression spring held together by two wire forms. These wires grip one end of the spring, then pass through the spring itself and form a loop on the opposite end that can be used for mounting. By placing such a wire form on each end of the spring, a mechanism is created that can be extended a limited amount before the spring reaches full compression, at which point the mechanism acts as a rigid member (figure 7.2).



(a) (b)

Figure 7.2: (a) An unloaded (short) tendon device and (b) a loaded (extended) tendon device.

What becomes a critical issue in the implementation of this modification then is the length of the respective elements; too short a tendon and the buckling problem will not be completely eliminated, too long a tendon and range of motion or available forces may become restricted. A few diagrams of a simplified case will help to determine the proper element lengths. Consider a simple joint driven by actuators attached to equal moment arms, and with a range of motion centered between the two actuators (at the midpoint of motion the moment arm is perpendicular to the actuators) (figure 7.3). For such a device, it should be apparent that were no tendon used, both actuators would be the same length. For the sake of simplicity, the left actuator will be referred to as actuator A, and the right actuator shall be likewise referred to as B. If a single tendon is added to one of the actuators, in this case actuator B, and the system were unloaded, then the optimum situation would for the tendon to just reach its minimum length while both actuators maintained their fully extended length (figure 7.3 a). Because the actuators are inextensible, this position is the extreme in the direction of actuator B. If the actuator A were to be fully pressurized, then the tendon would need to extend the full amount necessary to allow for the full contraction of the actuator, because actuator B is inextensible (figure 7.3 b).

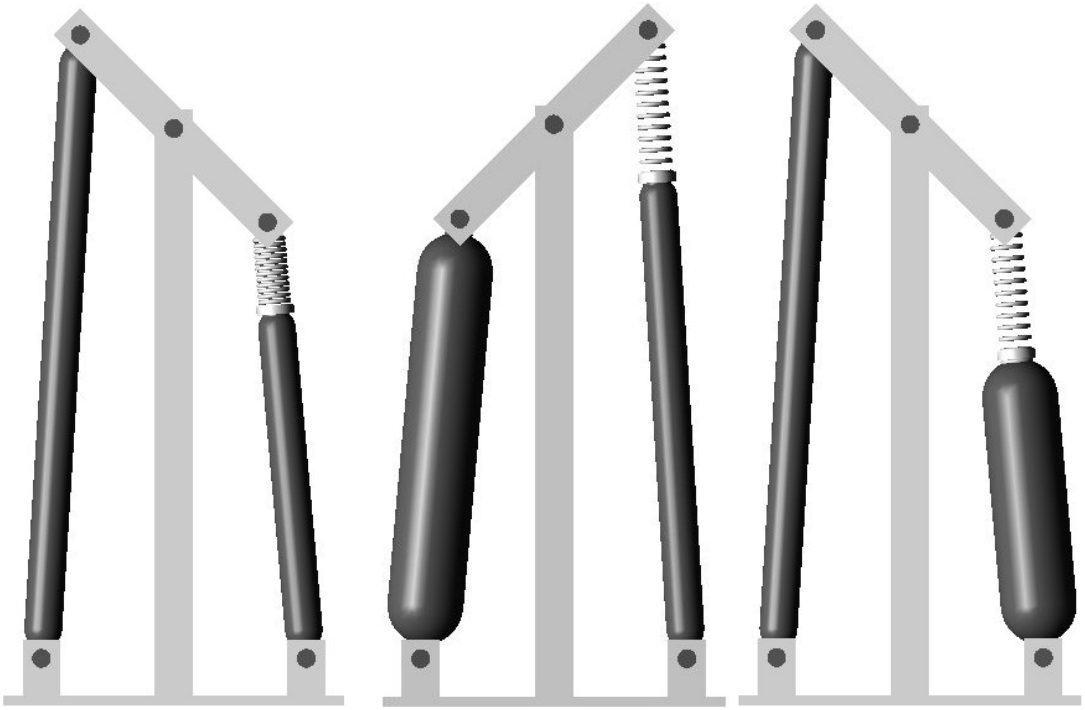


Figure 7.3: Three desirable states of a simple BPA driven armature with a tendon element.

For this simple case, it is possible to begin to determine the desired length ratios of actuator A (length A), actuator B (length B) and the tendon (length C) given the changes of length ΔA, ΔB, and ΔC. It should be obvious that:

ΔA = ΔC.

The available stroke for a Festo BPA is approximately 20%, thus,

ΔA = 0.2A

ΔB = 0.2B.

Furthermore, given that the tendon device is built around a compression spring, a reasonable relationship between the tendon length and stroke is:

ΔC = 0.5C,

and,

ΔC = 0.2A.

Lastly, because of the earlier stipulations on moment arm length and range of motion:

$$\mathbf{A = B + C + \Delta C.}$$

Making some substitutions from above,

$$\mathbf{C = 0.4\ A}$$

$$\mathbf{B = 0.4\ A.}$$

These relationships can easily be expanded to include devices with unequal moment arms (and correspondingly unequal length actuators) by noting the similarity of the motions involved. If actuator A is mounted with a moment arm of X and the B-C combination has a moment arm of Y, then:

$$\mathbf{\Delta A = (X/Y)\ \Delta C}$$

And following the same procedure as above,

$$\mathbf{B = 0.4\ (Y/X)\ A}$$

$$\mathbf{C = 0.4\ (Y/X)\ A}$$

Although these relationships become more complex for a joint that is not centered between the actuators, because the stroke of the actuators is relatively short compared to their overall length the above relationships will produce a close approximation; more accurate results could be determined on a case by case basis. However, in such a situation, there is most likely one actuator that is longer than the other, and thus does not use the full range of its stroke. In this case, the longer actuator is the obvious choice to have a tendon attached to it; the tendon should keep the same length ratio relative to actuator A as described above, and the remaining length of B should be determined from that.

It is desirable to use a fairly soft spring in the tendon for two reasons. First, while the tendon is not fully extended, it is not possible to completely control joint stiffness, as the state of the spring will also affect this. A soft spring will be compressed more quickly, as less force will be required from the actuators to accomplish this task. Secondly, it must be remembered that the force produced by the actuators is nonlinear, and in fact decreases as the actuator shortens; at the same time, spring force will increase. This means that the stiffer the spring, the less the actuator will be able to contract before becoming unable to produce the necessary force for further motion.

It is important to point out a few of the strengths and weaknesses of this design modification. First, and perhaps most importantly, inclusion of a tendon does not reduce the range of motion. This can easily be seen in figure 7.3; the actuator without the tendon maintains its full stroke, allowing the joint to move between the same extremes as before the tendon was added. The ability to co-contract is, for the most part, maintained. As figure 7.3 C demonstrates, even when actuator A is fully contracted, it is still possible to contract actuator B. However, until the tendon reaches its maximum extension, stiffness will not be controllable; in fact, because the stiffness of the spring is much lower than that of the actuator, the system will effectively retain a stiffness equivalent to the tendon in the direction of the tendon. The result is that a window of possible positions will exist in which it will not be possible to control joint stiffness in one direction; motion opposing the actuator without a tendon will still have to overcome the full stiffness of the actuator. Just as the addition of the tendon affects stiffness, it will also affect the range of possible

forces that the actuators can produce. Until the tendon is fully extended, the actuator it is attached to will not be able to deliver a force to the joint greater than that necessary to cause the current deflection of the spring. This is extremely important in determining where tendons are placed; they should not be placed on actuators responsible for maintaining stance, as they will not be able to provide the necessary forces for standing in some circumstances.

7.3 Initial Implementation of a Tendon on Robot V

It was of course desirable to test a prototype version of the tendon design on a single joint of the robot before replacing all joints with a tendon-actuator set. The femur-tibia joint of the right middle leg was chosen for this modification for a number of reasons (figure 7.4). First, it was driven by single opposing actuators, whereas many joints on the robot use a pair of actuators to produce motion in each direction. This means the joint selected needed only one tendon, while others would need two. Second, the actuators for this joint were relatively long, resulting in significant buckling and yet making the longer required tendon easier to make. Lastly, the actuators in the femur are the most distal on the robot, and thus some of the easiest to access and modify.

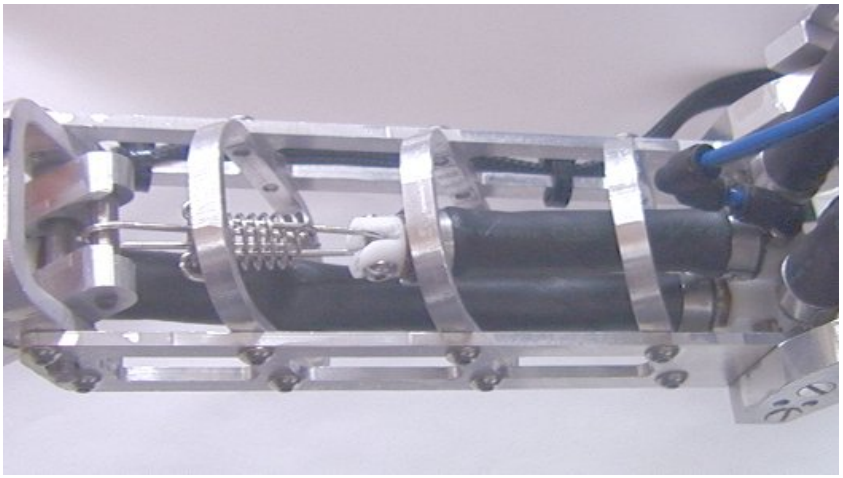


Figure 7.4: A tendon element added to the right middle leg femur-tibia joint.

7.4 Another Option

Another approach to solving the slack problem in the actuators was to attempt to maintain a small pressure in the actuators. This would cause a small contraction, thus reducing the actuators to a length where no slack was present. This process of course has the advantage of not requiring tendons, but would require careful monitoring and adjustment of pressures; at extreme joint positions it was not desirable to maintain any pressure in one actuator, as at this point it should be fully extended.

Although computationally more demanding, this method was ultimately selected for a number of reasons. First, and most importantly, the tendon driven joint was found to be rather unstable, due to the range of positions in which the tendon spring was not bottomed out, which resulted in the attached actuator being unable to control position or stiffness. The joint itself was not unstable, but the actuator-tendon assembly was; the controller produced small extensions and contractions of the actuator, which resulted in vibration of the spring over a very small displacement, although no vibration of the limb was noticeable. This deficiency was most

noticeable when the joint was perturbed so as to stretch the tendon, as when the joint was released, the controller could not adequately deal with the sudden and uncontrollable changing length and stiffness of the tendon spring. In addition to this limitation, in operation, the tendon proved to be somewhat lacking in robustness. In this experiment, a tendon of 40% total allowable actuator length (2.75 inches total tendon length, 1.75 inch unloaded compression spring) with a stiffness of 5 lbs/in was used. Although both of these problems could be overcome in time, the comparable success of the "tendon reflex" developed in software by Brandon Rutter eliminated the need for this. It may be worth further research on future robots, however.

Chapter VIII: Integration of Sensors

8.1 Feedback Requirements

Possibly one of the most important demonstrations of Robot V's capabilities was its ability to locomote, albeit roughly, without the use of any sensors. As previously discussed, this clearly illustrated the benefits gained by developing a mechanical system with animal-like preflexes. This walking was not, however, the cockroach-like locomotion that was the ultimate goal of the project. To attain anything close to the smooth motions seen in nature, it was necessary to integrate some level of feedback into the system.

The earliest robots produced in the Case Biorobotics lab had only rudimentary feedback mechanisms. Robot II, for example, had an angle sensor at each joint and a simple load sensor in each tibia that was used to determine if contact with the ground had been achieved or not (Espenschied et al. 1996). Robot III, which was much more complex, initially used only joint angle sensors (its load sensors were never used in its control system). In effect, this provided information only on the robot's kinematics, information on the system dynamics was not available. Although position could be controlled, the forces produced by the robot's pneumatic cylinders remained unknown, and as a result, the ground reaction forces produced by each leg were unknown. Locomotion in this state was not pretty, to say the least, but most likely could have been vastly improved by implementing a valve system capable of trapping air. It was believed that at least two types of feedback—position and force—would be needed onboard Robot V. Position sensors would of course be necessary to determine joint position and thus proper placement of feet and posture;

some sort of independent force feedback would be necessary to determine actuator forces, and thus joint stiffness.

8.2 Position Feedback Devices

The position feedback mechanism used by previous robots was a standard rotary potentiometer. This is simply a resistor that changes resistance as an attached shaft is rotated. By reading the voltage across the device, an angle can be calculated. Although these are robust and well understood mechanisms, they do carry with them some disadvantages. Because the rotary shaft must be directly linked to the joint's axis of rotation, the potentiometers must often be placed in positions that could result in damage during operation. Furthermore, the rotary shaft often works loose from it's mounting, resulting in inaccurate signals and a frequent need for recalibration.

Although a number of other position sensing devices are present on the market, one that offers significant promise is the Flex Sensor patented and manufactured by Spectra Symbol. These are simply long, thin, variable resistors that change resistance as they are bent (Figure 8.1). By attaching these to a flexible mount, a flexion sensor with a small restoring force can be produced. These can be rigidly attached to elements on either side of a joint, so that a unique resistance is produced for a given joint angle.

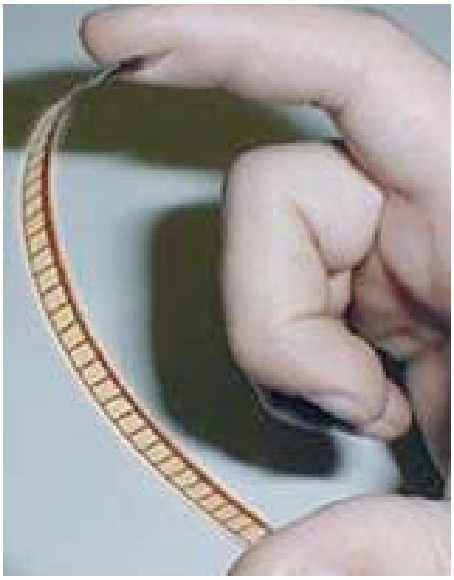


Figure 8.1: Spectra Symbol flex sensor.

The resulting sensors have a number of advantages over conventional potentiometers. First, as previously mentioned, the slippage problems that result from shaft loosening in potentiometers are alleviated. It was expected that this would greatly reduce the need for recalibration of onboard sensors. Secondly, the flex sensors are much lighter than their predecessors, and in addition they are also less costly. Finally, because they only need to be mounted to rigid elements on either side of the joint—unlike potentiometers which must either be mounted directly to the axis of rotation or connected to it via a linkage—more mounting configurations are available. This means that sensors can be moved to more protected positions to avoid damage during operation.

A sample sensor was fabricated using 1095 spring steel as the base material, using a standard two-part epoxy as the bonding agent between the steel and the flex sensor. Although some adhesives, such as cyanoacryllics, provide greater strength, the epoxy was chosen for its high ductility. Because the nature of the sensor demands

that it experience significant strains from bending, the use of high strength but brittle bonding agents was ruled out. The sensor was attached to the body-coxa β joint of the middle leg. This joint had a small range of motion compared to many other joints, so it presented a worst case scenario for the signal output. If the resistance change for this joint was sufficiently large for a control signal, then sensors placed at other joints with greater ranges of motion should also produce usable signals. To ensure that the epoxy provided a strong enough bond, a simple program was made to cycle the joint through one thousand cycles. Careful inspection of the sensor after this test showed no apparent damage.

The major difference between a potentiometer and a flex sensor in terms of their use as a control device is that the flex sensor does not have an integrated signal output, although a potentiometer does. A potentiometer circuit (figure 8.2) consists of a constant voltage V_i placed across a constant resistance to the circuit ground V_g. The sensor output travels along the resistor, so that as its position changes, so does the voltage V_s. As a result of this configuration, as long as no other high resistance elements are placed in series with the sensor, the output signal can take a value in the full range V_g–V_i.

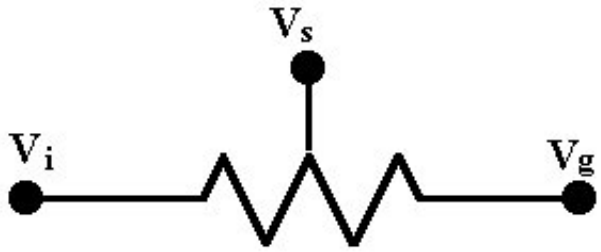


Figure 8.2: Schematic of a potentiometer circuit.

Flex sensors, on the other hand, do not have a signal output; instead they can be thought of simply as resistors with variable resistance. To produce a usable signal a second, constant value, resistor must be placed in series with the flex sensor (figure 8.3). As the resistance of the flex sensor changes, so will the voltage drop across the constant value resistor (and across the flex sensor, for that matter) and this value can be used for a control signal. One disadvantage of this arrangement is that because the flex sensor has a high resistance even when unbent, unless order-of-magnitude resistance changes can be achieved by the flex sensor (which is not the case), the signal voltages will not vary through the full range of V_g–V_i. This can be compensated to some extent by driving the circuit at a higher voltage than that which is read by the controller. For example, V_i could be 12 Volts, and the circuit could be calibrated to read sensor values V_s of 0-5 Volts.

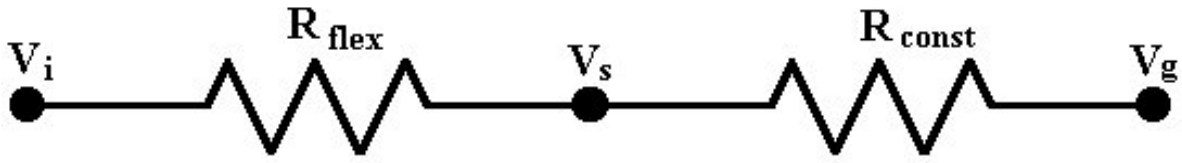


Figure 8.3: Schematic of a flex sensor circuit.

The more common technique for producing a viable output signal is to include the sensor in a Whetstone Bridge (figure 8.4). Because these resistors have a fairly small change in resistance compared to their baseline resistance, and because different sensors have different resistance ranges, the bridge configuration offers a greater level of tunability than the previously mentioned method. A Whetstone Bridge consists of four resistors arrayed in two parallel lines, one consisting of resistors R_1 and R_2 and the other of resistors R_3 and R_4. The voltage difference

between the nodes of the pairs of resistors, V_{sen}, is measured, and can be used for a control signal. This configuration is especially advantageous because proper selection of resistors R_1, R_2 and R_3 can be used to produce a controlled range of outputs for a given output range of variable resistor R_4 (Schwarz, Oldham, 1993).

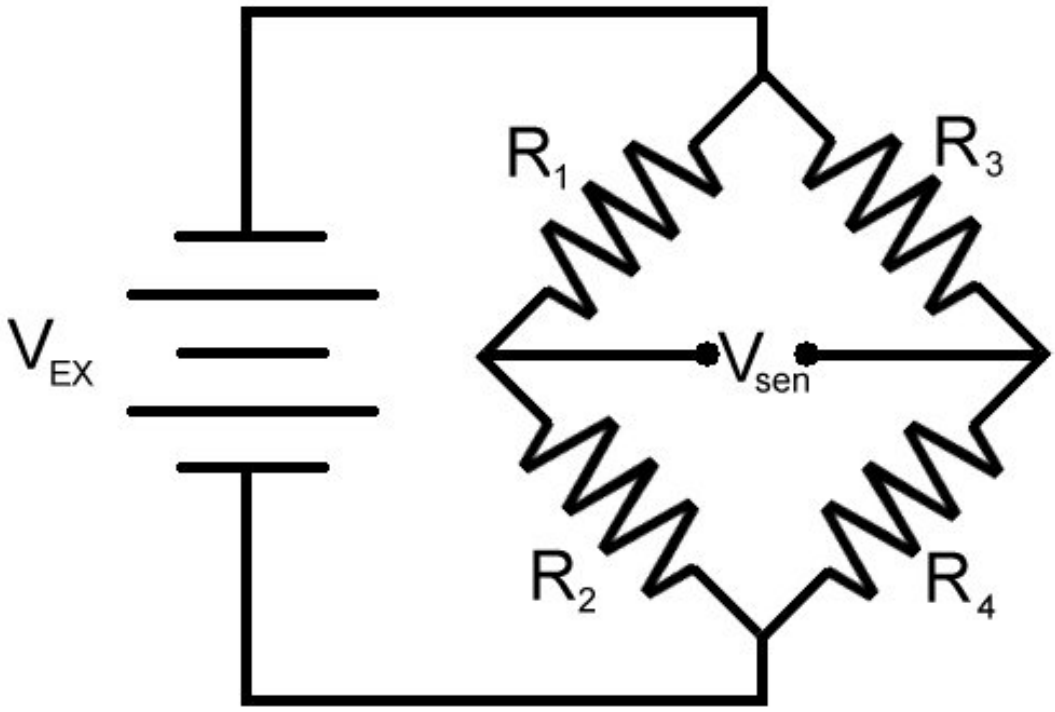


Figure 8.4: A Whetstone Bridge circuit.

Analysis of this circuit shows that:

$$V_{sen} = \frac{V_{EX}\,(R_1\,R_3 - R_2\,R_4)}{(R_1 + R_2\,)(R_3 + R_4\,)}$$

and, knowing the resistance range of the flex sensor R_4 it is possible to balance the bridge such that V_{sen} remains within a range usable by the control system. It should be noted though that this cannot act as a voltage magnifier; V_{sen} cannot exceed V_{EX}. Because of the great variability of resistors onboard the robot, an excitation voltage of

24 volts was used, and the bridges for each sensor were balanced to produce a voltage range of 0-1.25 V. Due to this large scaling down of voltage, some bridges were not perfectly balanced and would produce saturated signals near the extreme joint positions. This was seen as a worthwhile trade off though, as it allowed some other sensors that otherwise would have had too small of a range to produce usable signals.

Brandon Rutter wrote a simple closed-loop controller to test the viability of these sensors as control devices. This attempted to use the sensor mounted on the right middle leg body-coxa β joint to move that joint through sinusoidal motion. After sufficient calibration of the sensor data and tuning of the controller, reasonable motions were produced. The two most significant sources of error were slack lengths in the actuators and the inability of this controller to account for pressurized air already in the actuator. Further development of the controller, including the addition of force feedback mechanisms was expected to alleviate these problems.

A critical problem became apparent in the implementation of these devices when a rear leg femur-tibia sensor mount failed during testing. Although the sensor was not damaged, the spring steel mount fatigued and broke after a small number of cycles; it was believed that other mounts would fail in the same way. This issue was resolved by replacing the spring steel mount with one made of elastomer filled neoprene rubber with a hardness of 70 on the Shore A scale. This was chosen because this application did not require high strength, but large deformations were necessary, and the 300% elongation of the neoprene was expected to allow a great number of cycles. The adhesive backing on the sensors did not bond well to the neoprene, so a urethane adhesive was used to bond the two surfaces. Although this

did not provide as strong a bond as the sensor-to-steel bond, it was deemed suitable, especially when a layer of tape was wrapped around the ends of the sensor and mount to prevent a shear failure from starting. This process was later improved further by using a wire brush to prepare the neoprene surface for bonding and placing a layer of heat-shrink around the sensor and its backing. The heat-shrink was also used to control stiffness of the mount ends that did not hold the sensor. Because these ends had a lower stiffness than the material that also carried the sensor, they had a tendency to bend more, reducing the amount of bend—and thus the signal produced—of the sensor. The addition of heat-shrink over the sensor ends not only aided in reducing shear failure of the sensor-to-mount bond, but also decreased the amount of deflection of the mount regions that did not contain sensor material (figure 8.5).

Figure 8.5: The finalized sensor design.

Before assembling a sensor in this fashion, a neoprene mount was attached to the middle leg femur-tibia joint (this joint was chosen because it experiences the greatest range of motion on the robot) and cycled through one thousand flexion-extension cycles and inspected for damage. None was found, so a fully functional sensor was assembled on the rear leg femur-tibia joint. This joint was then cycled

through two thousand flexion-extension cycles, and both the mount and sensor were inspected for damage. Again, no problems were apparent, and this sensor configuration was deemed acceptable for future use.

8. 3 Implementation of Flex Sensors

Flex sensors were mounted on neoprene backings at all joints on the robot. Initial tests were performed to determine how clean a signal they produced. Brandon Rutter wrote a simple program to move the right middle coxa-femur joint through its range of motion, holding positions at ten degree intervals. This initial output (figure 8.6) was deemed suitable for control of the robot. Although there were some control stability issues, especially in the zero to negative ten degree range, it should be noted that this was an artifact of the controller itself, and that the data produced by the sensor produced a very clean signal; the joint was in fact vibrating at this time. The instability in this range was a result of both actuators being near their rest position, and thus slack; this problem is further addressed in Chapter VII. Furthermore, the position spikes seen early in the motion are a result of the slack actuators being pressurized and actually extending before contracting as the slack is removed.

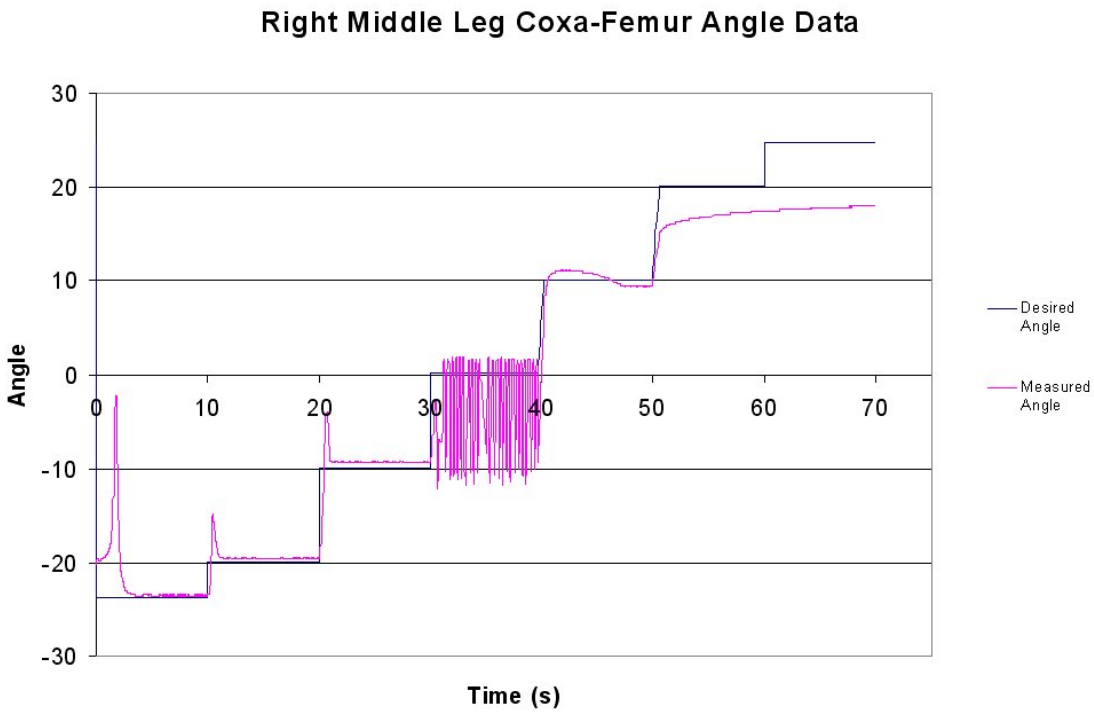


Figure 8.6: Angle sensor output from the right middle coxa-femur joint.

8.4 Force Sensing Devices

It is virtually impossible to directly measure a force; instead, a number of mechanisms have been created to measure the effect of a force, such as a displacement, and thusly determine the force necessary to cause such an effect. One such device, which has been used on previous CWRU robots, is the strain gauge. This consists of a very thin wire folded back on itself many times. Applying a force to either end of the strain gauge will cause a deformation in the direction of the force—strain—that results in a change in the resistance of the wire. These sensors can be mounted on rigid elements of a structure to determine the strain present on, and thus the loads applied to, those elements (Bachmann 2000).

Although they are quite well understood and manufactured in a wide variety of forms, strain gauges present many problems. They are extremely delicate, and require precision in their mounting to accurately determine the forces being applied to them. They must also be placed in the path of the force being measured; for example, to measure the ground reaction force of a robot's leg, strain gauges should be placed on the most distal limb segment, as far from the foot as possible so as to maximize bending strains. Likewise, to measure the force produced by an actuator, strain gauges should be placed on the actuator's mounting element. This has many undesirable results. Large actuator mounts to allow room for a strain gauge reduce available actuator stroke, and thus moment, while at the same time increasing the mass and inertia properties of the limb. Delicate sensors placed at distal points of a limb have a tendency to suffer damage, resulting in poor or unusable signals. Two additional problems encountered in the use of strain gauges are their notoriously weak signals, which are highly prone to noise, and their inability to measure the state of an individual actuator in a pair of opposed actuators. This effect can easily be illustrated using the diagram below (figure 8.7). If actuator 1 is pressurized, it will pull on the lower link, which will in turn produce a tension in actuator 2. Strain gauges at the attachment points of both actuators will consequently experience a force, even though only one actuator is activated. Although the addition of joint angle data does somewhat alleviate this cross-talk problem, it would be more desirable to have sensor data dependent on the state of only one actuator (Kingsley 2001).

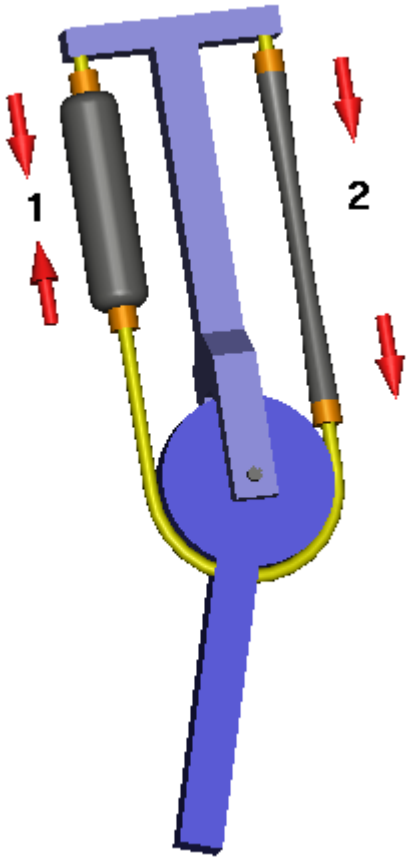


Figure 8.7: Pressurizing actuator 1 produces a force on actuator 2.

Pneumatic actuators readily offer another solution to this problem, as the pressure in an actuator can be used to determine its force output. In the simple case of a pneumatic cylinder, if one knows the area of the piston and the pressure in the cylinder, the force can easily be determined by multiplying the two together. BPAs have a much more complex relationship between force, pressure, and length, but as length data are already available from the joint angle sensors, it is possible to determine actuator force through only pressure and length data. Pressure sensors also provide solutions to many of the problems inherent to strain gauges. They can be mounted away from the actuator itself, allowing for longer actuators and reducing

limb weight, while also reducing the chances of sensor damage. In addition, pressure sensors are much easier to replace when damaged than strain gauges, especially since they do not require the precise recalibration that strain gauges do. Pressure sensors are also manufactured to provide much larger signals than strain gauges, so amplifying circuits are not required, nor are the sensors as susceptible to noise.

Previous work has shown that the Motorola MPX 5700 pressure sensor (figure 8.8) provides an excellent signal, while being small enough to mount 48 sensors on the robot. Furthermore, tests demonstrated that because pneumatic line losses were relatively small, the sensors did not need to be placed close to the actuator to achieve a reliable signal (Kingsley 2001). This allowed for banks of eight sensors to be placed immediately adjacent to each pair of valves (figure 8.9) where they would be protected from damage.

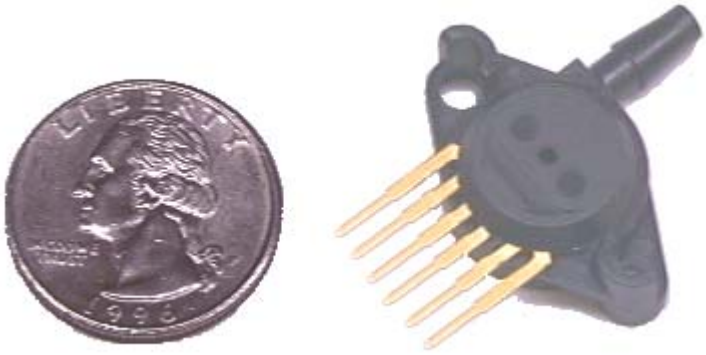


Figure 8.8: Motorola MPX 5700 pressure transducer.

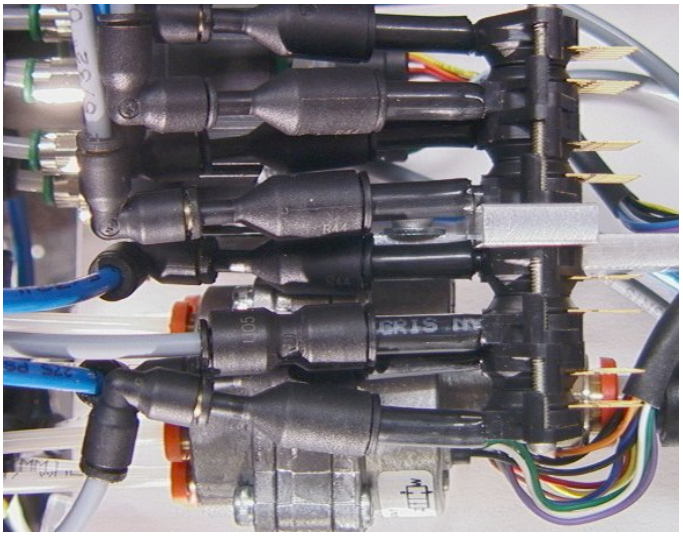


Figure 8.9: A pressure sensor array.

8.5 Implementation of Pressure Transducers

One significant advantage that the pressure transducers hold over the joint angle sensors is that they are constructed to produce a usable output signal, much like a potentiometer (figure 8.2). This meant that they could be provided with an input voltage of five volts and produce a clean signal that was already within a usable range, without the need for any conditioning circuits. The controller written by Brandon Rutter was modified to output the signals from the pressure sensors on the right middle coxa-femur joint as it was moved through ten degree increments (figure 8.10).

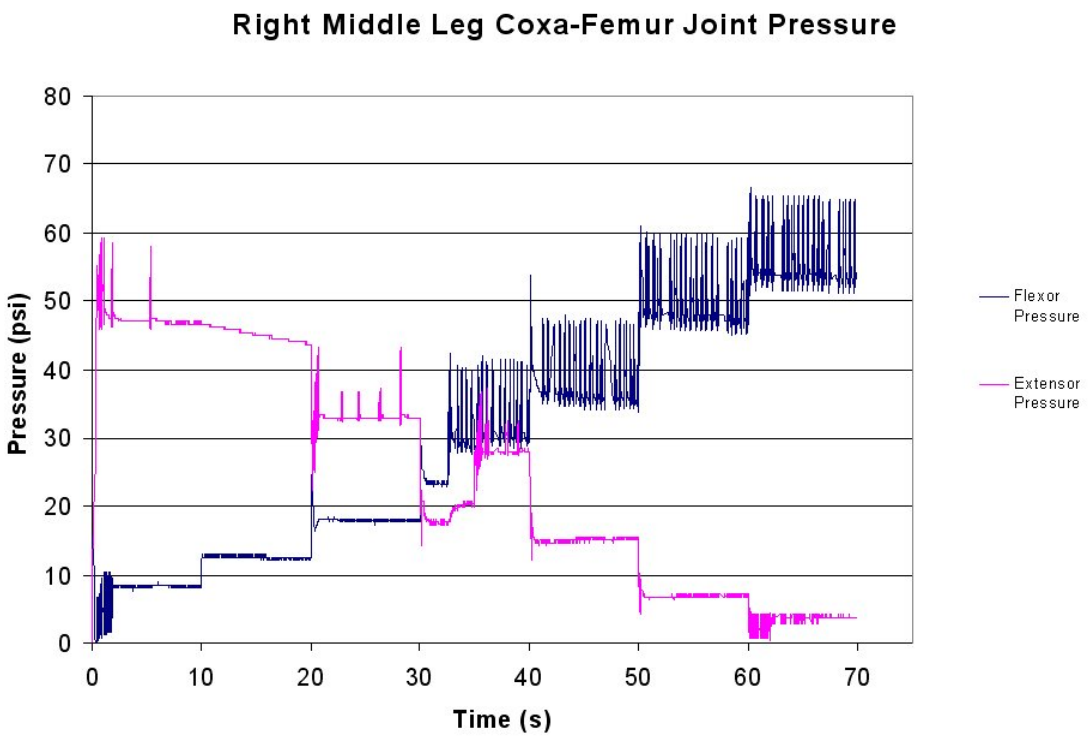


Figure 8.10: Pressure sensor data from the right middle coxa-femur joint.

This output was, obviously, unsuitably noisy; however, this was not a failing of the sensors themselves, but of the control scheme. As deviation from the desired position was sensed by the joint angle sensors, airflow in to and out of the actuators was controlled using pulse width modulation (PWM) to flutter the valves controlling the joint open and closed. The result was that bursts of pressurized air were released by the inlet and exhaust valves. The noise in the figure, especially as the flexor was pressurized, was a direct result of these bursts of pressurized air. There were a number of ways this signal could be improved. In software, the signal could be averaged over a small time step to reduce the effect of the individual pulses; however, this would add more computations to a controller with a number of real time demands already placed on it. A number of filtering circuits, such as a resistor and capacitor in

parallel or an inductor in series with the sensor, could be implemented, but these would delay the signal. The selected option, then, was to implement a mechanical filter in the system.

The pressure sensors were manufactured with a barbed fitting that attached to a ¼" OD nylon tube. This short length of tube was in turn attached to a reducer that fed into the network of 5/32 OD nylon air line that served as the pneumatic system of the robot. A number of filtering devices were placed in the length of ¼" tube attached to the flexor and extensor pressure sensors of the right middle coxa-femur joint in the hope of muffling the pulses of air while retaining approximately the same average pressure. The most effective material found for this (at the suggestion of Brandon Rutter's visiting father) was a soft polymer earplug. One of these was slightly trimmed, compressed, and inserted into the air line of the right middle coxa-femur flexor, where it was then allowed to re-expand. Operation of the joint showed a dramatic improvement in the signal (figure 8.11) with minimal loss of data. This test was deemed highly successful, and the filters were implemented on all pressure sensors.

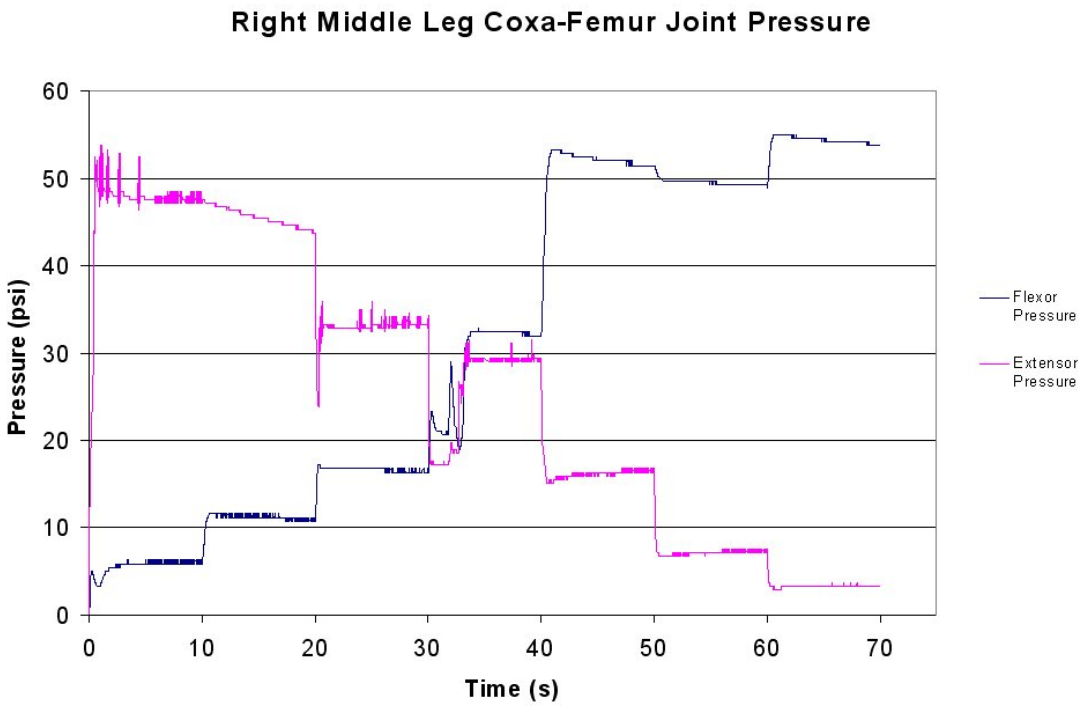


Figure 8.11: Right middle coxa-femur joint pressure sensor output with a filter on the flexor signal.

8.6 Onboard Electronics

Ajax was never intended to be an autonomous robot; it was expected to require both offboard power (compressed air and electricity) and control. This meant that the vast majority of the electronics necessary for control, such as the processors required for valve control and data collection, remained separate from the robot itself. Even though this was the case, it was still desirable to perform some limited signal processing onboard the robot, as well as condensing the power and signal lines for all 72 sensors into one region. To accomplish this, a small circuit board was constructed and placed on the robot's back, between the middle and rear legs. One of the primary tasks for this component was to house the Whetstone Bridge circuits that were necessary for operation of the flex sensors.

The circuit board (figure 8.12) accepts two power inputs, one for the flex sensors with a 24 volt differential, and another for the pressure sensors, which was maintained at 5 volts. LEDs were connected to each of these circuits to ensure proper polarity. Pressure sensor power was delivered to pin connections which in turn connected to individual clusters of pressure sensors. Flex sensor power was distributed through three strips, with one central high voltage strip in the middle of the board, and low voltage strips on the left and right side. Pin connections allowed the flex sensors to be plugged into their respective bridge circuit. After potentiometers were added to some joints (Chapter XI) additional connections were added for these sensors on the board, and their output overrode that of the flex sensors for the same joints.

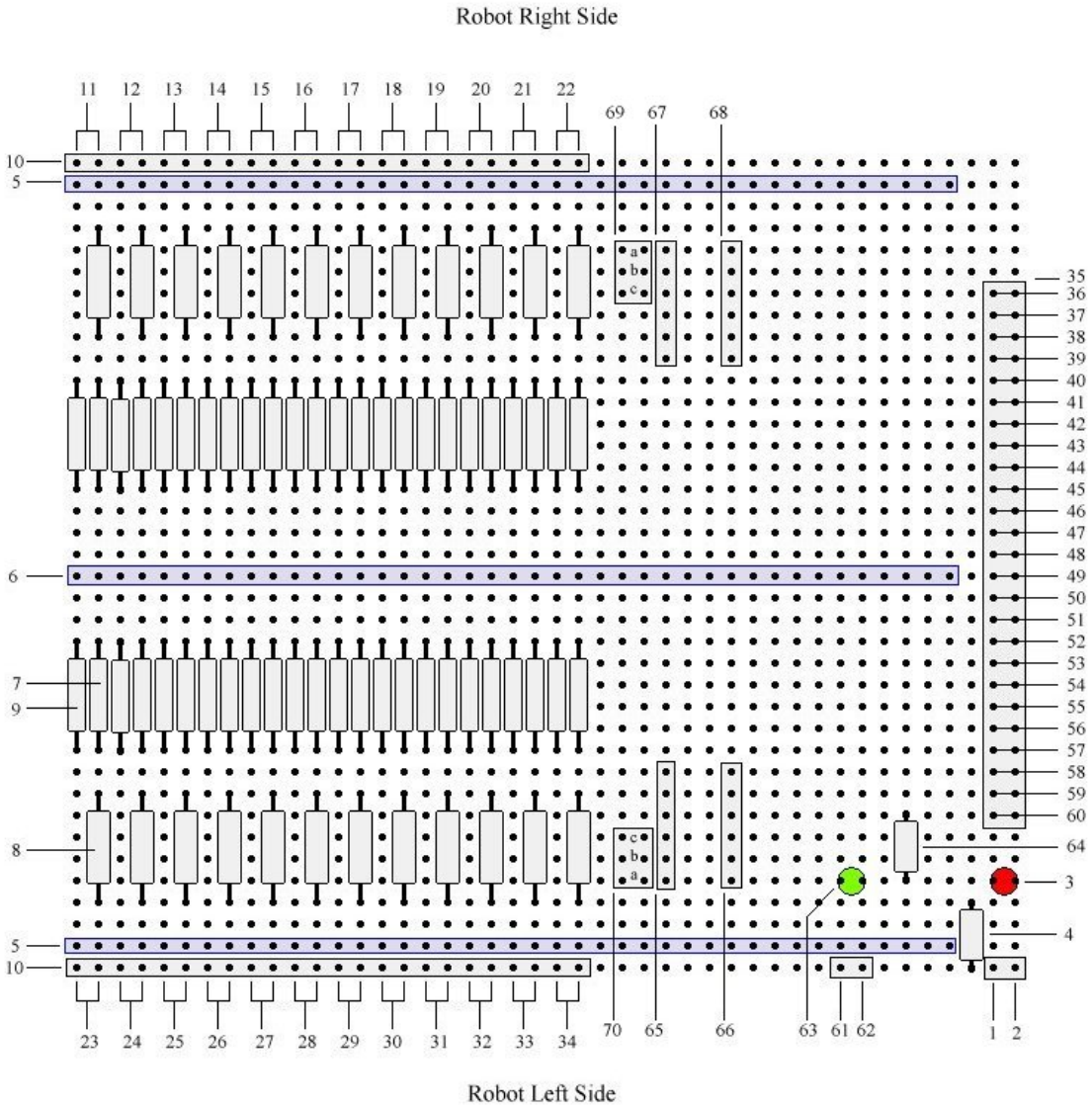


Figure 8.12: The circuit board carried onboard the robot. Components are labeled as follows:

1	Flex sensor high voltage pin	15	Right front gamma angle sensor pins
2	Flex sensor low voltage pin	16	Right middle tibia angle sensor pins
3	Flex sensor LED (red)	17	Right middle femur angle sensor pins
4	Flex sensor LED resistor	18	Right middle alpha angle sensor pins
5	Flex sensor low voltage power rail	19	Right middle beta angle sensor pins
6	Flex sensor high voltage power rail	20	Right rear tibia angle sensor pins
7	Bridge circuit resistor R_1	21	Right rear femur angle sensor pins
8	Bridge circuit resistor R_2	22	Right rear beta angle sensor pins
9	Bridge circuit resistor R_3	23	Left front tibia angle sensor pins
10	Flex sensor connection pins	24	Left front femur angle sensor pins
11	Right front tibia angle sensor pins	25	Left front alpha angle sensor pins
12	Right front femur angle sensor pins	26	Left front beta angle sensor pins
13	Right front alpha angle sensor pins	27	Left front gamma angle sensor pins
14	Right front beta angle sensor pins	28	Left middle tibia angle sensor pins
29	Left middle femur angle sensor pins	31	Left middle beta angle sensor pins
30	Left middle alpha angle sensor pins	32	Left rear tibia angle sensor pins

33 Left rear femur angle sensor pins
34 Left rear beta angle sensor pins
35 Flex sensor output connecting pins
36 Flex sensor low voltage pins
37 Right front tibia sensor output
38 Right front femur sensor output
39 Right front alpha sensor output
40 Right front beta sensor output
41 Right front gamma sensor output
42 Right middle tibia sensor output
43 Right middle femur sensor output
44 Right middle alpha sensor output
45 Right middle beta sensor output
46 Right rear tibia sensor output
47 Right rear femur sensor output
48 Right beta tibia sensor output
49 Left rear beta sensor output
50 Left rear femur sensor output
51 Left rear tibia sensor output
52 Left middle beta sensor output
53 Left middle alpha sensor output
54 Left middle femur sensor output
55 Left middle tibia sensor output
56 Left front gamma sensor output
57 Left front beta sensor output
58 Left front alpha sensor output
59 Left front femur sensor output
60 Left front tibia sensor output
61 Pressure sensor high voltage pin
62 Pressure sensor low voltage pin
63 Pressure sensor LED (green)
64 Pressure sensor LED resistor
65 Pressure sensor cluster power (high)
66 Pressure sensor cluster power (low)
67 Potentiometer cluster power (high)
68 Potentiometer cluster power (low)
69 Potentiometer signal right
   a. Right front beta
   b. Right middle beta
   c. Right rear beta
70 Potentiometer signal left
   a. Left front beta
   b. Left middle beta
   c. Left rear beta

Chapter IX: Leg Coordination Mechanisms: The Cruse Controller

9.1 Controller Hierarchy Theory

A biological control system can be broken into a number of layers. At the lower levels, muscles properties and local reflexes provide quick movements, which are well suited to respond to sudden perturbations. At higher levels, reside task planners, posture controllers, and adaptive controllers that generate the large scale behavior of such a system (Loeb at al. 1999). In the same vein, the control system of Robot V can be broken down into distinct layers. The bottommost of these is the mechanical system itself. The passive properties of the BPAs, as well as the inclusion of torsion springs at some joints build into the system a level of passive "reflexes" or "preflexes" that Robot III could not exhibit. Above this, in the middle level are the interleg, intraleg and joint controllers also found in Robot III. The intraleg controller performs the inverse kinematics to properly coordinate the joints of a leg and position feet. The interleg circuit coordinates leg motions—the main focus of this chapter. On the highest level of the locomotion control system, which has not been developed for Robot V but will be ported from Robot III, will be the posture controller, responsible for controlling body motions.

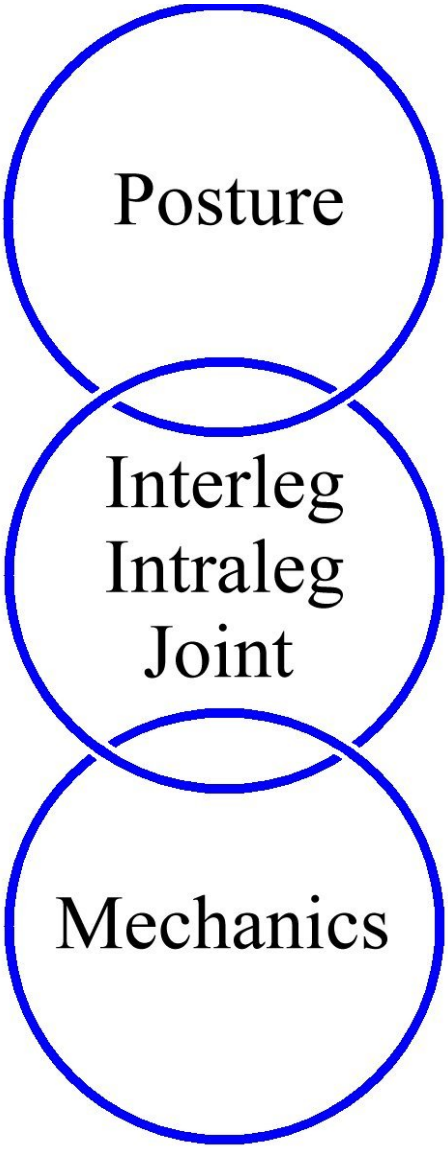


Figure 9.1: Hierarchical distribution of Robot V's controller

9.2 Biological Motivation

Insects regularly demonstrate the ability to navigate over a wide array of surfaces, and are capable of coordinating their legs in a continuous range of statically stable gaits (Wilson 1966). At slow speeds, insects demonstrate a wave gait, wherein a single leg is in swing at a time. As speed increases, the number of legs

simultaneously in swing increases until a tripod gait is achieved, where the front and rear legs on one side of the body and middle leg on the opposite side are all in swing at the same time (Watson et al. 1997).

As with the mechanical design of the robot, it was desired to develop a biologically inspired controller that could reproduce at least some of the robust walking demonstrated by the insect. As previously mentioned, this controller would reside on three levels: at the lowest level, the mechanical system itself was designed to assist in walking; above that, local controllers were needed to both control individual leg motions as well as coordinate motions between legs; finally, at the highest level , the overall posture of the robot needed to be controlled. Many of the desired biological features are derived from this second level, and specifically from the inter-leg coordination mechanisms, which in the animal are controlled by the thoracic circuits.

9.3 The Cruse Model

A robust biological models of insect leg coordination was proposed by Holk Cruse, and is based in the stick insect *Carausius morosus* (Cruse 1990). Although this is of course not expected to be the same mechanism as that found in the cockroach, it is fairly well understood and has been shown to be robust enough to control previous robots built in the Case Biorobotics Lab as well as the TUM robot (Weidemann et al. 1994) and others including xx (Frik et al. 1999). The network described here is based on that reviewed by Cruse (1990), but Durr et al. (2004) have made additions to this network and have integrated it into Walknet, which includes intra-leg and posture control.

The network (figure 9.2) consists of a node for each leg, which sends and receives signals with all adjacent legs. Each leg has a pattern generator that causes it to carry out a basic step-and-swing motion between an anterior extreme point (AEP) and posterior extreme point (PEP). The swing speeds of all the legs are the same under all circumstances. The stance speeds, however, while the same between all legs, can be varied to produce different walking speeds.

Cruse (1990) proposed six different types of signal that could be relayed between legs, but previous work has shown that only three of these (labeled 1, 2 and 5) are needed to produce stable walking on flat terrain (Espenschied et al. 1993). Of the three unused mechanisms, one assists in navigation over more complex terrain by targeting footfalls to land where more anterior limbs have found stable substrate, and the other two seem to provide a means to more equally distribute loads among the limbs contacting the ground. Although potentially useful for navigating complex, sparse terrains, these mechanisms were not included in the current incarnation of the controller as for the time being walking on a smooth surface presented a great enough challenge.

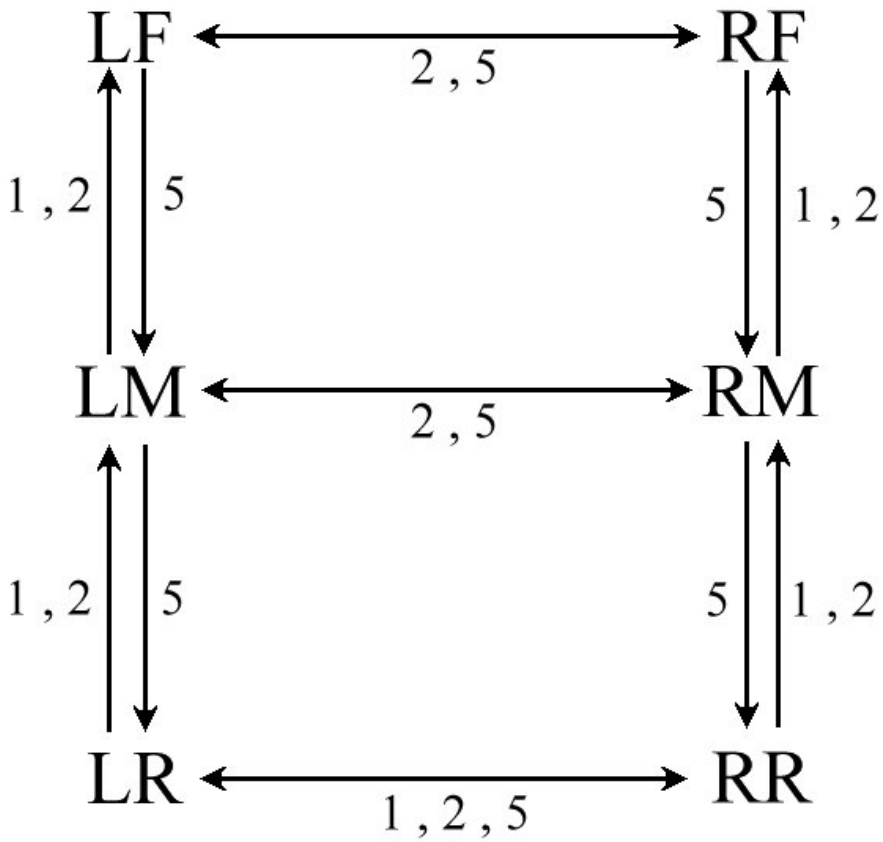


Figure 9.2: A version of the Cruse controller utilizing only three mechanisms.

During walking, each leg begins its stance phase at a preset, and constant, AEP, and begins to move toward the preset PEP as stance continues. Signals received from adjacent legs act to adjust this target PEP, and once a leg has reached what is currently defined as its target PEP, it will enter a swing phase and return to the AEP. Mechanism one acts as an inhibitor to prevent nearby legs from entering swing while the sending leg is currently in swing. It outputs a constant negative value (AEP is defined as a more positive position than the PEP) while a limb is in its return stroke. This has the effect of extending the stance phase of legs receiving this signal. Mechanism two is a positive impulse which is sent when a leg completes its return

stroke and enters stance. This motivates nearby legs to enter their swing by reducing the distance between the AEP and PEP. In implementation, this mechanism was made to last for the first ten percent of the unmodified PEP. Mechanism five is a positive ramp function, which begins when a leg enters stance. This also has the effect of reducing the distance between the AEP and PEP of neighboring legs, but is different from mechanism two in that it increases as the leg moves through stance; thus, the further a leg is in stance, the more motivation there is for an adjacent leg to enter swing (figure 9.3) (Cruse 1990).

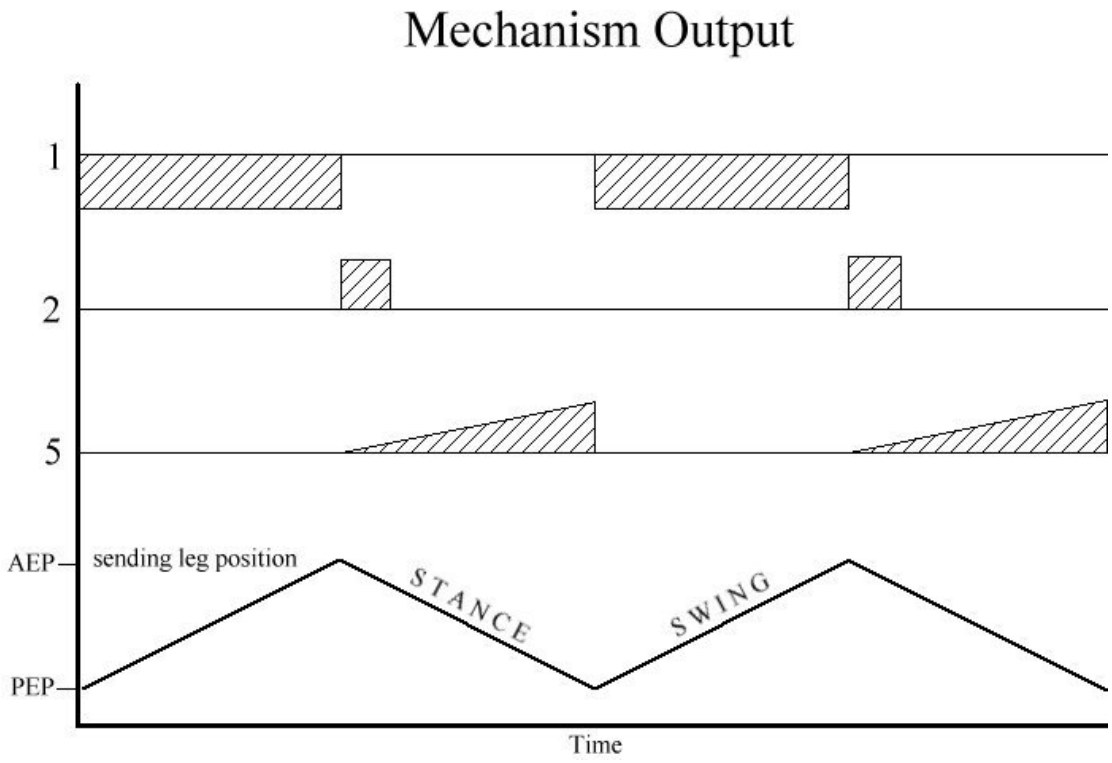


Figure 9.3: The three mechanisms utilized by the simplified Cruse controller.

9.4 Cruse Controller Implementation

The controller was implemented in C code for use on the robot. The general algorithm is explained here, and the actual code is contained in Appendix A. A preset

swing speed, speed factor (speed of stance with respect to swing) AEP, unmodified PEP and network scaling constants are set. The controller then begins looping through the network for discrete time steps and adjusting mechanism, position, and state values as necessary. The algorithm below describes this process:

```
Set constants and initial network values
Loop for small time step
        For each leg:
                Update foot position
                If necessary, change stance/swing state
                Set network values
        Calculate updated PEP values using new network values
```

Although this algorithm is simple enough to implement, it is not guaranteed to produce a stable controller. It is necessary to determine scaling factors for the network mechanisms, and to differentiate between ipsilateral (same side of body) and contralateral (opposite sides of the body) connections (contralateral connections should be weaker than their ipsilateral counterparts (Espenschied et al. 1996)), and the scaling factors for these signals can be quite different. Furthermore, different walking speeds will affect whether a given set of scaling factors will or will not produce stable walking. Through trial and error, the following values were found, in simulation, to produce stable locomotion over a full range of walking speeds:

Table 9.1: Network constants for stable walking.

	Ipsilateral	Contralateral
Mechanism1	1.0325	0.3
Mechanism2	2.0	0.1
Mechanism5	7.25	3.08

9.5 Cruse Controller Simulated Results

Using the above network scaling factors, walking simulations were produced over a wide range of speeds. Graphs of foot positions show not only that the controller produced statically stable gaits, but that a range of gaits, from tripod to a single leg wave gait, were possible. As expected, at high speeds, where the stance speed was close to the swing speed, tripod and near tripod gaits were produced by the controller. The maximum speed attainable while maintaining smooth, stable walking, appeared to occur with a stance speed of 71% of the swing speed. This sort of limitation was expected to occur, as at high stance speeds the legs are unable to return to their AEP in swing before adjacent legs complete their stroke. Although the factor of stance-to-swing speeds of 0.71 does seem low (theoretically, it should be possible to attain a 1:1 ratio) it may be possible to attain higher ratios as discussed later. A graph of the gait at a stance-to-swing speed ratio of 0.60 is shown below (figure 9.4).

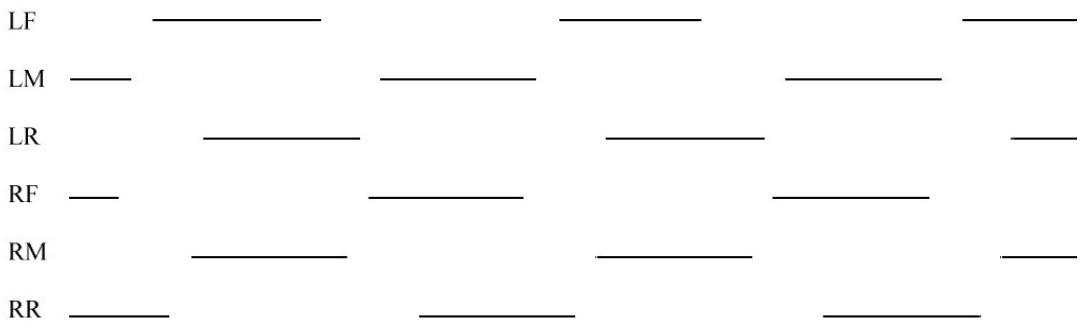


Figure 9.4: A near-tripod gait produced by the Cruse controller with a speed ratio of 0.6. Black lines indicate a foot is in swing, empty spaces indicate stance. The plot covers four seconds of locomotion.

As the speed ratio was reduced, thus slowing the stance speed and the overall rate of locomotion, the gait smoothly shifted toward a wave gait. This happened because the individual tripods (left and right) began to spread apart as the legs shared less common time in swing. Below one can see what, discounting the middle legs, could be considered a quadruped gait, where the front and rear legs on opposite sides of the body are simultaneously in swing (figure 9.5).

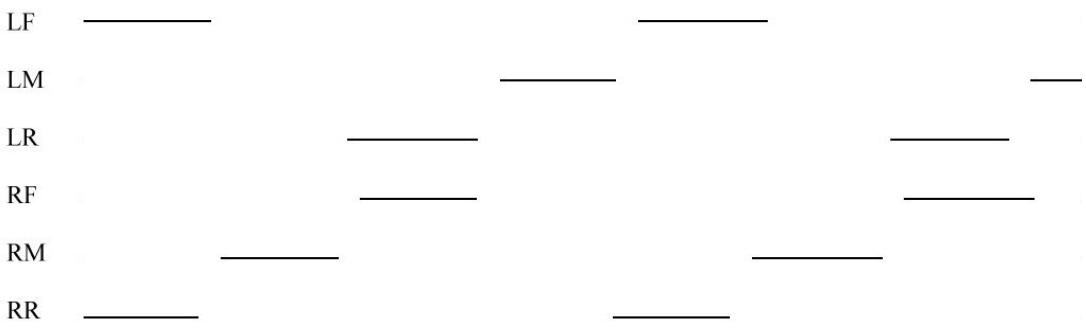


Figure 9.5: A quadruped-like gait is produced at a speed ratio of 0.28.

At still slower speeds, the controller shifted into a wave gait, where one leg at a time entered swing (figure 9.6). Note that this maintains the metachronal wave, with the posterior legs entering swing before adjacent anterior legs. This gait was achieved at a speed factor of approximately 0.166, although at slightly higher speeds the wave gait was discernable, with some overlap of swing periods between front and rear legs on opposite sides of the body.

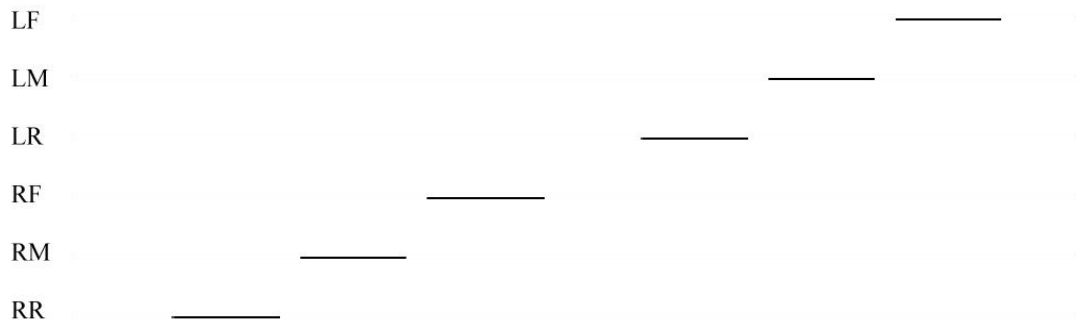


Figure 9.6: At a speed factor of 0.12, the controller produces a wave gait.

9.6 Breaking the Controller

Even though the controller was able to produce a continuous range of stable gaits, a concerted effort was made to find situations in which it was not able to produce usable results. There were a few conditions identified under which it could be made to function improperly; solutions to these problems were identified.

One of the easiest failures to generate in the system is a result of the initial conditions. Consider starting the controller when contralateral feet have the same starting position, i.e. both front legs are a distance x from their AEP, both middle legs are a distance y from their AEP and both rear legs are a distance z from their AEP. Because contralateral legs in this situation will be receiving the same signals, they will enter stance and swing simultaneously, resulting in a “swimming” motion (figure 9.7). One can consider this akin to a human standing with both heels aligned in the coronal plane; if walking was begun with leg motions driven only by their position, the human would then begin hopping.

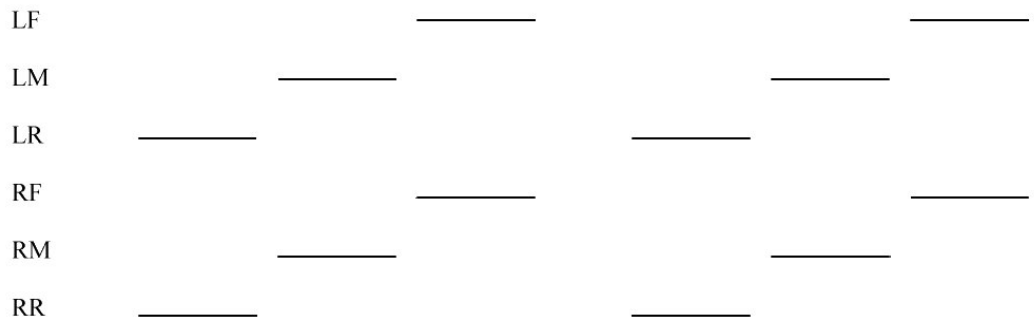


Figure 9.7: The controller demonstrates “swimming”, moving contralateral legs simultaneously. In this case, the speed factor was 0.28 and all legs started halfway between the AEP and PEP.

The solution to this problem was conceptually quite simple; a bias was introduced to the contralateral network connections so that contralateral legs no longer received the same signals. In the controller, this was simply represented by multiplying all contralateral signals sent by a right leg to a left leg by a biasing factor. This approach was found to produce stable gaits (figure 9.8), but the necessary bias to achieve this was dependent on both the initial leg configuration and the speed factor. Below are the minimum biases for three speed factors and two different initial configurations. In configuration A, all feet started halfway between the AEP and PEP. In configuration B, the front legs started one third of the distance from the AEP to the PEP, the middle legs started halfway between the AEP and PEP, and the rear legs started two thirds of the distance from the AEP to the PEP.

Table 9.2: Contralateral biasing minimum values.

Speed Factor	**A**	**B**
0.60	1.0502	1.0
0.28	1.008	1.004
0.12	1.0	1.18

Using slightly higher biasing factors was not found to be detrimental; nor did the biasing have any apparent effect on normal operation. Using a contralateral bias of 1.25, all three speeds were found to achieve a stable gait within approximately fifteen seconds—three steps were needed for stability at a speed factor of 0.60, four steps at 0.28, and five at 0.12. However, before reaching this static stability, the controller often entered periods of great instability. These transient issues are addressed along with the next controller failure.

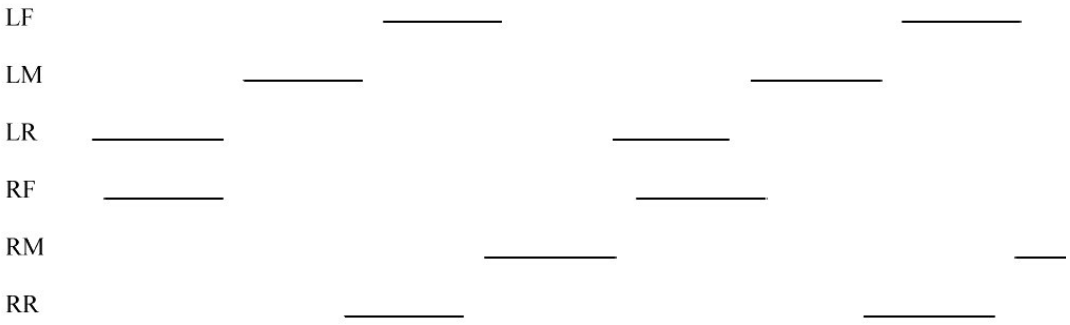


Figure 9.8: A contralateral bias eliminates the "swimming" effect. In this case, the same initial conditions as figure 9.7 were used, along with a bias of 1.1.

Although stable gaits were produced for a full range of speeds, depending on the initial conditions of the system, transients could be produced in the early stages of locomotion, before the steady state gait was achieved. In some cases, these were not noticeable, or resolved before the first steps had been taken; however, in other circumstances excessive instability occurred (figure 9.9). This would consist of two or more adjacent legs entering swing simultaneously, and sometimes up to four legs being in swing for a short period. The initial positions of the legs seemed to greatly aggravate this problem, as did the initial state (stance or swing).

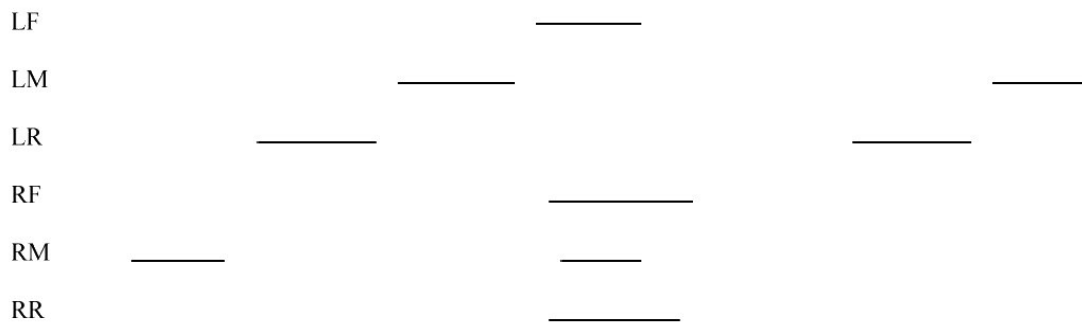


Figure 9.9: The controller produces unstable transients at a speed ratio of 0.24. Time shown is zero to four seconds; the controller did eventually become stable. Note that for a short time all three right legs as well as a left leg are simultaneously in swing.

A solution to this problem was found by noting that when the legs begin in positions close to what would occur in the desired gait, transient problems do not occur; effectively the system begins in some already desired configuration and does not need to be adjusted to achieve the desired gait. Conversely, if the initial leg positions are significantly different from what would occur in the desired gait, their positions relative to each other must be changed to attain a valid set of positions, leading to instability in the process. Unfortunately, there is no initial leg configuration that would be valid for any randomly given speed. One solution was to implement a "start-up" phase, in which the legs are first moved to a stable tripod position, with one tripod near the AEP and the other near the PEP, and begin walking at a speed that would result in a tripod gait. From this initial state, the speed was slowly ramped down until the desired speed was achieved. Using 0.05 second time steps in the controller simulation, a ramp with a one percent decrease per time step was found to result in a stable system that achieved a wave gait (thus transferring from fastest gait to slowest gait) in approximately six seconds, or four steps (figure 9.10).

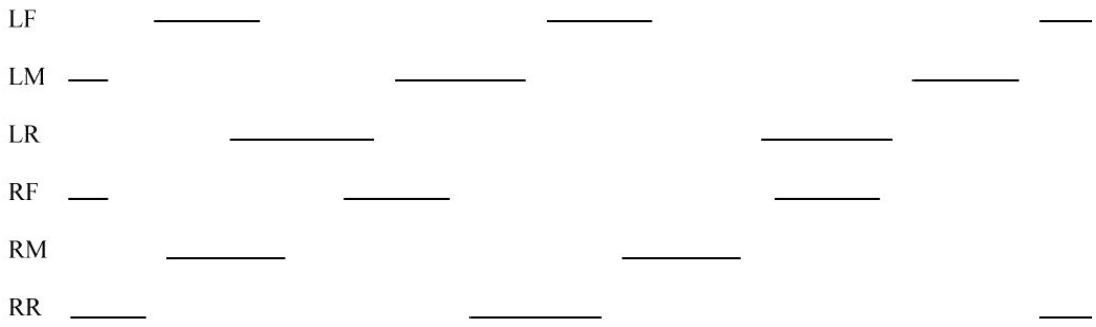


Figure 9.10: The controller transitions from a speed ratio of 0.60 to 0.24 during startup. The graph shows the time frame of 1.15 to 4.15 seconds.

The same concept was implemented in another algorithm, this time starting from a slow speed and ramping up to the desired gait. This had the advantage of erring on the side of caution—the controller started by moving only one leg at a time—at the cost of a slight loss of speed, as in this case it took approximately seven seconds to transfer from a wave gait to a tripod gait, albeit four steps were still needed to complete this change in speed. Although this also required a slightly less intuitive initial leg configuration, a sufficient one was found, and ultimately this ramping up of speed was chosen as the desired start-up method because of the greater initial stability it offered by moving only one leg at a time.

Implementation of this algorithm had the dual advantage of not only addressing the problem of instability during transients, but because it adjusted the system to a set of initial conditions known to produce stable walking, the first problem—which resulted in the "swimming" motions—was also addressed. Quite simply, both problems were a result of the initial conditions of the system. The robot itself should be expected to operate with any given initial conditions; that is, after

being set on a surface, it would be expected to stand and begin walking without careful adjustment of leg positions by an operator. Implementation of the "start-up phase" would then be required in hardware as well; after standing, the robot would be expected to adjust its legs to positions conducive for walking before the onset of locomotion.

Although control of the initial conditions solved the contralateral coordination problem, the contralateral bias was retained in the system. As was previously noted, it did not have any apparent detrimental effect on gaits; however it was believed that situations may arise during operation in which this would be beneficial. For example, a leg could be perturbed, delaying its motion so that its position was close to that of its contralateral mate. In this case, although the other legs would continue in acceptable positions, the contralateral bias would be instrumental in returning the system to normal operation.

9.7 Controller Stability

One critical issue in the implementation of any controller is the stability of the system. If small perturbations in controller values result in a lack of stability, it may not be possible to move the controller from simulation to reality. Furthermore, it may be possible to further improve controller performance by analyzing and adjusting the scaling factors of the control functions. The Cruse controller presents a great challenge in the later case, as the six network scalars produce twenty-six signals that can each have an effect on the value of other signals. Because of this complexity, deriving control equations for the controller was unrealistic; instead, a stability analysis was performed wherein the network scaling factors were multiplied by

constants, and the resulting gaits were studied to determine if they were stable and reasonable. In this way, the effect of variations of the scaling factors on overall system stability could be seen.

Before evaluating stability, one important question had to be answered: what makes a gait bad, or, for that matter, good? In some cases, the answer is obvious. If two adjacent legs are simultaneously in swing, static stability has probably been lost (this is not necessarily true, consider the case of both middle legs in swing while the other four are in stance). At other times, the controller demanded "stutters" where it attempted to move feet through very brief periods of swing (figure 9.11). While this may or may not have resulted in an otherwise stable posture, it was unrealistic to expect the hardware to be capable of entering and exiting swing in periods of one tenth of a second. Other gaits may simply look dangerously close to instability, with one limb entering swing at exactly the same time that an adjacent one completes swing. For the sake of simplicity, only two criteria were used to determine controller stability: did it remain statically stable and were periods of stance and swing reasonable for implementation in hardware? If these were met, the controller was considered stable.

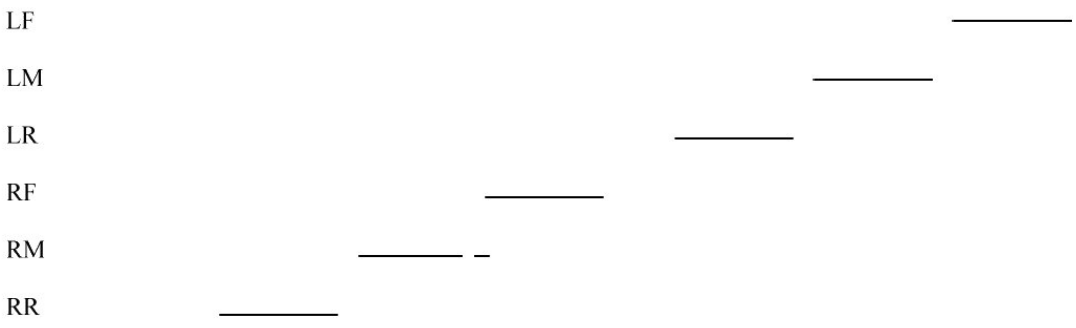


Figure 9.11: The right middle leg makes a "stutter" step.

The stability of the controller was examined at three speeds (figures 9.4-9.6) which represented three different gaits: at a speed factor of 0.60, a near tripod gait; at 0.28, a transitional gait; at 0.12 a wave gait. Each network constant was multiplied by a scaling factor to determine both the minimum and maximum values over which the controller remained stable. These tests were performed using the ramping up of initial speed and contralateral bias of 1.25 previously described. As with other examples, a time step of 0.05 seconds was used, and the simulated gaits were studied over a period of sixty seconds. In the tables, I represents an ipsilateral connection, C represents a contralateral connection. It should be noted that similar stability studies (Espenschied et al. 1993) have been performed, but did not address the effect of varying speed on stability.

Table 9.3: Stability ranges for Cruse network constants.

Maximum Values

Speed	**1 I**	**1 C**	**2 I**	**2 C**	**5 I**	**5 C**
0.60	2.30	3.96	∞^*	9.80	1.23	1.44
0.28	2.60	3.20	∞	27.7	2.10	3.00
0.12	2.70	3.34	1.49	23.2	1.39	2.60

* Values up to 1000 were tested, at which point stability was maintained

Minimum Values

Speed	**1 I**	**1 C**	**2 I**	**2 C**	**5 I**	**5 C**
0.60	0.14	0	0.04	0	0	0
0.28	0.28	0	0.63	0	0	0
0.12	0.56	0	0.83	0	0	0

As can be seen from above, the ipsilateral connections play a much more important role than the contralateral; in fact, the system maintains stability with the removal of the contralateral connections. In addition, it should be noted that while the second mechanism was necessary, at high speeds the maximum value tended toward infinity; only at slow speeds did large values cause instability.

Because of the complexity of this system, it was also important to look at the higher order effects of perturbations on stability as well; that is, how simultaneous changes in multiple mechanisms affected stability. To study all possible combinations of effects would be beyond the scope of this work, and, while academically interesting, would have little effect on the implementation of this controller. Instead, to provide a rudimentary understanding of the interplay of these mechanisms, combinations of two mechanism scalars were multiplied simultaneously by the same factor and the stability of the simulation was again analyzed at the three speeds previously studied.

Table 9.4: Second order stability ranges for Cruse network constants.

Speed: 0.6

	Maximum							**Minimum**		
1 C	2.9	xxx	xxx	xxx	xxx	0	xxx	xxx	xxx	xxx
2 I	∞	4.28	xxx	xxx	xxx	0.03	0.3	xxx	xxx	xxx
2 C	2.64	4.34	15.00	xxx	xxx	0	0	0.06	xxx	xxx
5 I	1.13	1.14	1.20	1.20	xxx	0.22	0.01	0.36	0	xxx
5 C	1.25	1.40	1.40	1.40	1.10	0.02	0	0.22	0	0.08
	1 I	**1 C**	**2 I**	**2 C**	**5 I**	**1 I**	**1 C**	**2 I**	**2 C**	**5 I**

Speed 0.28

	Maximum							**Minimum**		
1 C	3.18	xxx	xxx	xxx	xxx	0	xxx	xxx	xxx	xxx
2 I	∞	3.90	xxx	xxx	xxx	0.69	0.74	xxx	xxx	xxx
2 C	2.56	3.90	28	xxx	xxx	0	0	0.69	xxx	xxx
5 I	2.01	2.19	2.73	1.98	xxx	0.25	0	0.32	0.13	xxx
5 C	2.93	3.53	4.89	5.04	1.84	0.17	0	0.70	0	0
	1 I	**1 C**	**2 I**	**2 C**	**5 I**	**1 I**	**1 C**	**2 I**	**2 C**	**5 I**

Speed 0.12

	Maximum							**Minimum**		
1 C	2.89	xxx	xxx	xxx	xxx	0	xxx	xxx	xxx	xxx
2 I	1.34	1.29	xxx	xxx	xxx	0.81	0.82	xxx	xxx	xxx
2 C	2.47	3.41	1.49	xxx	xxx	0.46	0	0.82	xxx	xxx
5 I	1.40	1.38	1.30	1.31	xxx	0.36	0	0	0	xxx
5 C	3.02	3.24	1.44	4.00	1.27	0.41	0.12	0.79	0	0
	1 I	**1 C**	**2 I**	**2 C**	**5 I**	**1 I**	**1 C**	**2 I**	**2 C**	**5 I**

While the data presented in these tables are too complex to describe the effects of every combination at every speed, some important trends can be pointed out. As a general rule, stability ranges were reduced when two variables were

simultaneously varied, however, in many cases, such as the 1I-2I combination, one mechanism clearly dominated the other. This data is best viewed as a tool for understanding such connections, and using that information to more finely tune such a controller.

9.8 Wait A Second….

The observant reader will have no doubt noticed that certain mechanisms seem to be unnecessary; specifically, the contralateral mechanisms can individually be reduced to zero, and the only case where two contralateral mechanisms cannot simultaneously be reduced to zero without instability occurring is the slow speed mechanism 1-5 combination. This suggests that these mechanisms are largely unnecessary and could possibly**,** with the exception of mechanism 5, be removed from the system. While this is technically true, these mechanisms do serve a useful, if not vital, purpose.

While a variety of gaits could be considered "good", it is certainly justifiable to categorize some as being better than others. For example, while one gait may produce stable walking, it may be possible to produce another at the same speed that reduces the number of legs simultaneously in swing, or extends the average time each leg spends in stance. These are properties of a gait that are much harder to quantify, and should be implemented in hardware to truly evaluate their performance.

Regardless of how these gait properties are evaluated, the contralateral mechanisms play an important role in effecting interleg timing. A clear example can be seen in the following gait patterns, generated at a speed factor of 0.12 with varying values of the mechanism five contralateral scaling constant (figure 9.12, figure 9.13),

which can be compared to figure 9.6, which uses the base scaling factors at the same speed.

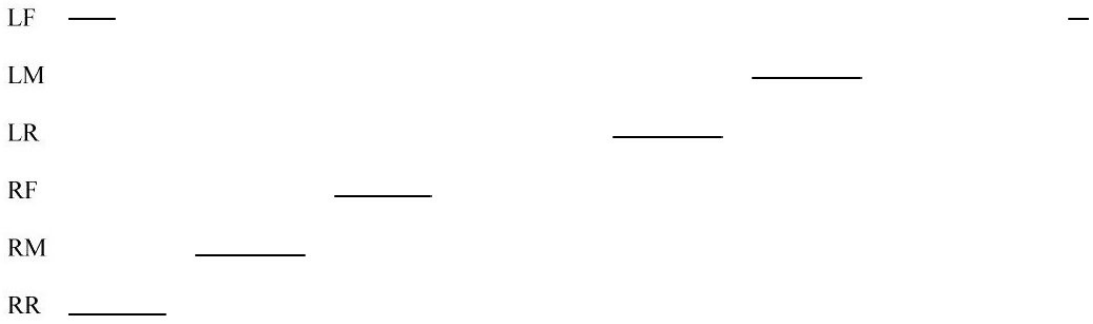


Figure 9.12: Simulated gait at a speed factor of 0.12 with no contralateral mechanism 5.

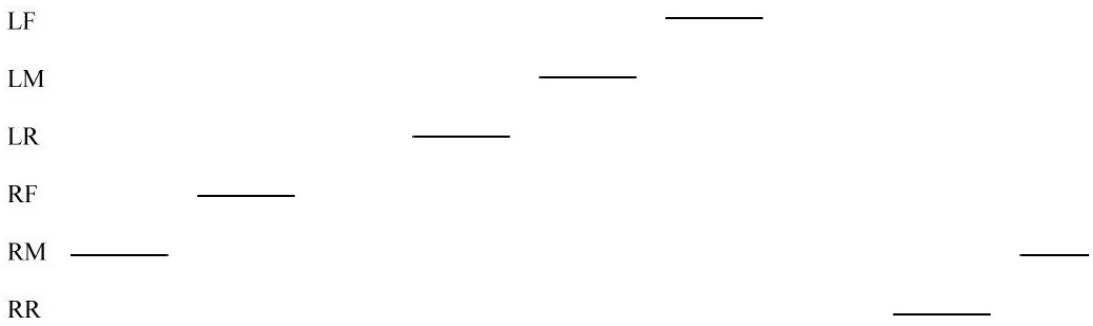


Figure 9.13: Simulated gait at a speed factor of 0.12 and a contralateral mechanism 5 scalar of 2.5.

Although both of the above gaits meet the criteria for stable walking, removal of contralateral mechanism 5 produces some undesirable characteristics: overlap of left front and right rear swing phases and inconsistent timing between initiation of swing between adjacent legs. This demonstrates that, while some mechanisms may not be strictly necessary, they are useful in producing better gaits.

9.9 Some Additional Issues

It was mentioned earlier that it was expected that the system should be able to reach, or come very close to, speed factors of 1.0 without losing stability. Testing of the controller has shown that this is in fact the case. This ability is directly related to the speed of swing and the size of the discrete time steps used by the controller. Lowering the swing speed or decreasing the time step size does in fact allow higher speed ratios, with values approaching unity. This effect occurs simply as a result of accuracy—smaller values allow the system to more closely determine exactly when the switch between stance and swing should occur, thus leading to more overall stability in the system, resulting in the possibility of stability at higher speed ratios.

Another important point to note is that in many cases, specific values of the network constants were not important; as long as the scalars were within a given range, system performance remained the same. For example, over the entire stability range of ipsilateral mechanism 2 there was no apparent variation in gait (some small variation might become apparent at higher resolutions) but values outside this range, even slightly outside this range, produced catastrophically unstable gaits. What this demonstrates is that while acceptable and unacceptable gaits can be produced, there are no specific “best” network constants. Instead, viable regimes—with ranges that in turn effect the ranges of other regimes (as shown in the stability analysis)—exist, and the system, while fairly robust to perturbations, must be tuned with these properties in mind.

9.10 An Aside

While capable of producing a continuous range of hexapod gaits, the Cruse controller is in no way restricted to use on six legged robots. As an example of its versatility, the controller was modified from a hexapod to a quadruped, using only the front and rear connections described above. Simulation showed the system to be just as effective in producing a continuous range of gaits as the hexapod version; two such gaits are shown below (figure 9.14 and figure 9.15). A quadruped gait was present at speed factors of 0.6 and above, while a wave gait occurred at speed factors of 0.26 and below. This was achieved with the following network constants:

Table 9.5: Network constants for a quadruped Cruse controller.

	Ipsilateral	Contralateral
Mechanism1	1.0325	0.3
Mechanism2	2.0	0.1
Mechanism5	3.625	1.54

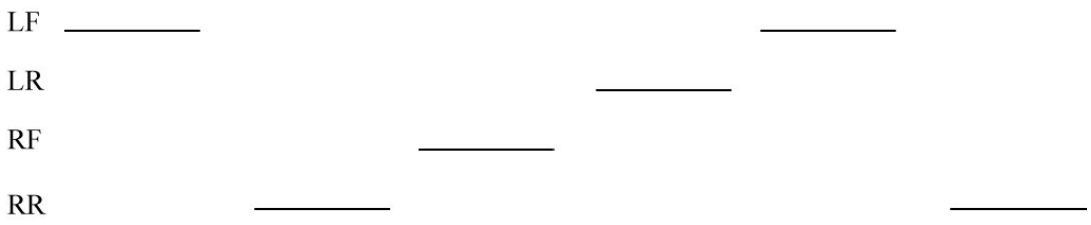


Figure 9.14: A wave gait with a quadruped controller at a speed factor or 0.26.

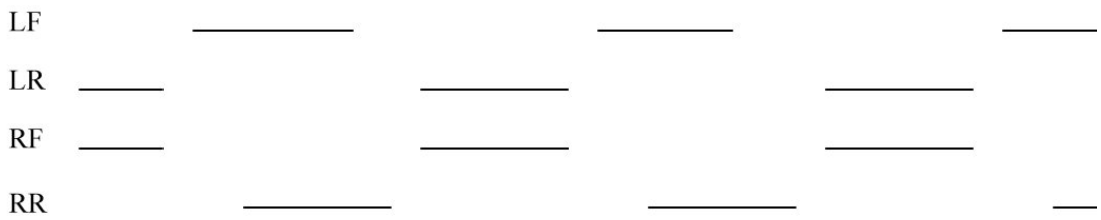


Figure 9.15: A quadruped gait at a speed factor of 0.6.

It is of course reasonable to expect that this controller could be modified to function for any given number of pairs of legs, making it a very useful and effective tool for gait coordination.

Chapter X: Modifications to the Cruse Controller for Omnidirectional Movements

10.1 Controller Limitations

The Cruse Controller described in the previous chapter presented a number of limitations, most notably that it only generated forward locomotion. It is of course desirable to have a robot that is capable of turning, and possibly lateral movement. It is also however, desirable to have a single controller capable of producing all of these tasks, as ultimately they all result from the motions of the feet.

At its most basic level, the Cruse Controller generates footpaths from the set AEP to a variable PEP, as well as a simple state definition of whether or not a foot is in contact with the ground. As a result, the controller needs only to specify position in one dimension and a contact state. Realistic operation of a robot of course demands specification of foot positions in three dimensions as well as contact state. As previously described, three coordinate axes were assigned to the robot (figure 10.1) with the X-axis pointed forward, the Y-axis to the left, and the Z-axis pointed up. Using this spatial convention, the controller was expanded to control foot trajectories in all three dimensions.

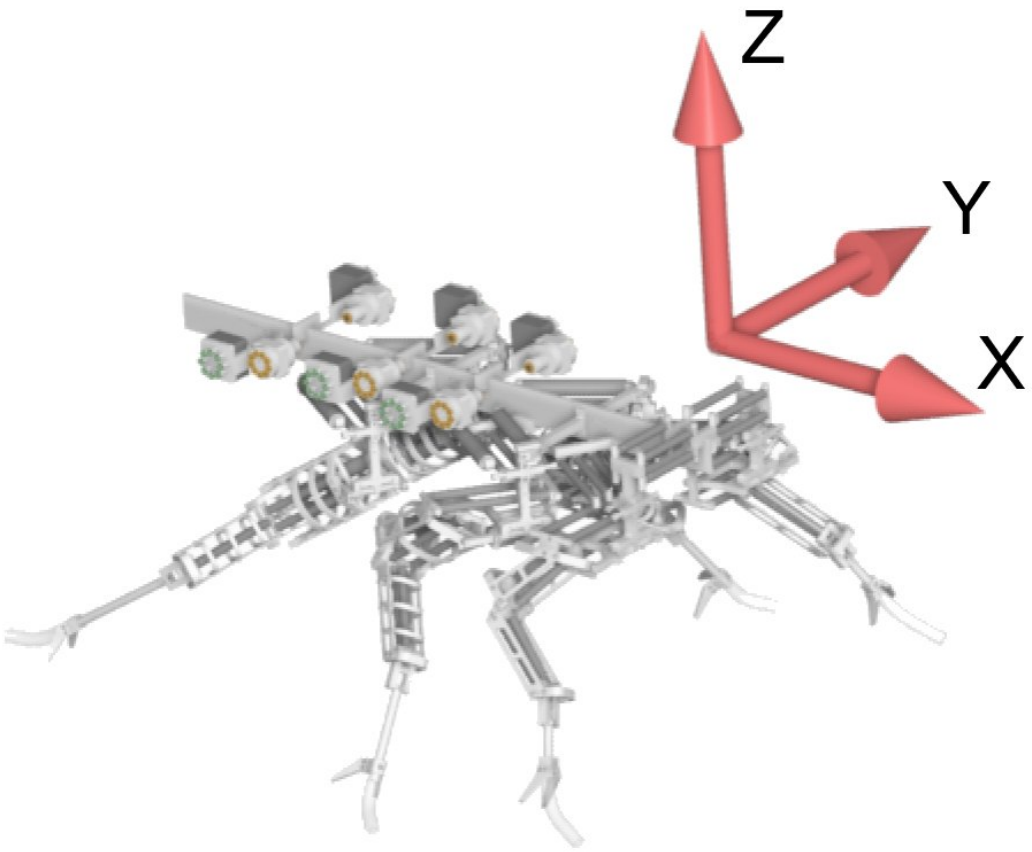


Figure 10.1: Robot V with its coordinate axes.

10.2 Foot Elevation Control

Based on an inverse kinematic analysis of the robot performed by Jong Choi (based on that for Robot III by Nelson (2002)), reasonable leg motions were found to occur when mounting points of the legs were eight inches above the ground level. Furthermore, reasonable maximum elevations during walking for the feet were found to be three inches above the ground for the front and two and a half inches for the middle and rear feet.

Although a variety of different swing paths could be considered practical including the nominal cockroach foot motions, it was decided that a parabolic motion

from the end point of stance, through the maximum point, and back to the AEP would be simple enough to implement while at the same time providing very good ground clearance. The primary difficulty in implementing this scheme is that because the end point of stance (hereafter referred to for simplicity as the PEP although it is in reality a modified PEP) is variable, a generic equation to describe the parabola of motion was needed. This would take the basic form of:

$$C\,(X - PEP)(AEP - X) + ground = Z$$

where X is the current X position of the foot, Z is the vertical position, ground is the elevation below the body axis of the foot while in stance and C is a value that will ensure that the foot reaches its peak value midway between the AEP and PEP. At the midpoint:

$$C\left(\frac{AEP + PEP}{2} - PEP\right)\left(AEP - \frac{AEP + PEP}{2}\right) + ground = Z_{max}$$

thus:

$$C = \frac{4(Z_{max} - ground)}{(AEP - PEP)^2}$$

And so:

$$Z = \frac{4(Z_{max} - ground)}{(AEP - PEP)^2}(X - PEP)(AEP - X) + ground$$

For operation of the controller, a desired Z position variable for each foot was introduced along with the X position variable. When a leg was in stance, this was set to a constant value equal to the desired elevation of the body above the ground (in the case of Robot V, 8 inches) and while a leg was in swing, the foot elevation was given by the equation described above.

10.3 A New Geometry

The work describe in this section is based on the omnidirectional controller for Robot II as described in Espenschied (1996). Before the controller was adapted to control position in the X-Y plane, it was necessary to define the geometry of the work space. In the first incarnation of the Cruse Controller, motion in the X direction was simply defined by a constant AEP value and a step length which led to the PEP value. This was, of course, lacking in any definition of Y position, but also for reasons related to the geometry and required trigonometry, it was desirable to redefine the work space of each foot as a circular area with a predefined center and radius. Inside this circular area, it would be possible to rotate the AEP and PEP with respect to the body axis by some angle θ, resulting in the leg moving at an angle with respect to the body. The new AEP and PEP would then be referred to as AEP' and PEP' (figure 10.2).

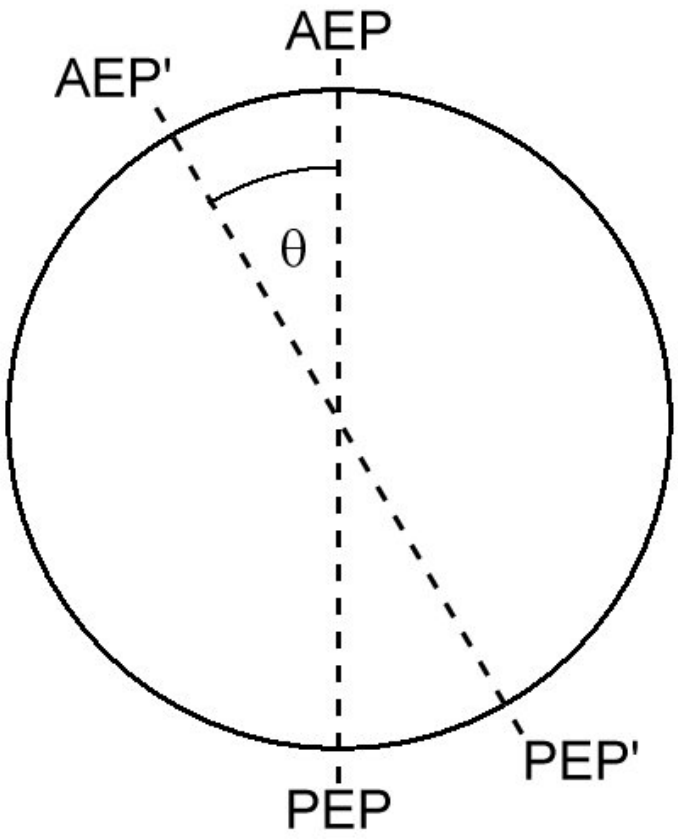


Figure 10.2: Adjustment of AEP and PEP with respect to the body.

Using data provided by Jong Choi's inverse kinematic model of the robot, the following approximations of foot workspaces were developed (figure 10.3-10.5). Based on these maps, it was believed that a nominal six inch step length (6 inches from AEP to PEP) would be practical. In these graphs, the mounting point of the limb is at the point (0,0), positive X values are in the anterior direction, and negative Y values are distal direction. The areas outlined in red are the workspaces chosen for operation of each leg.

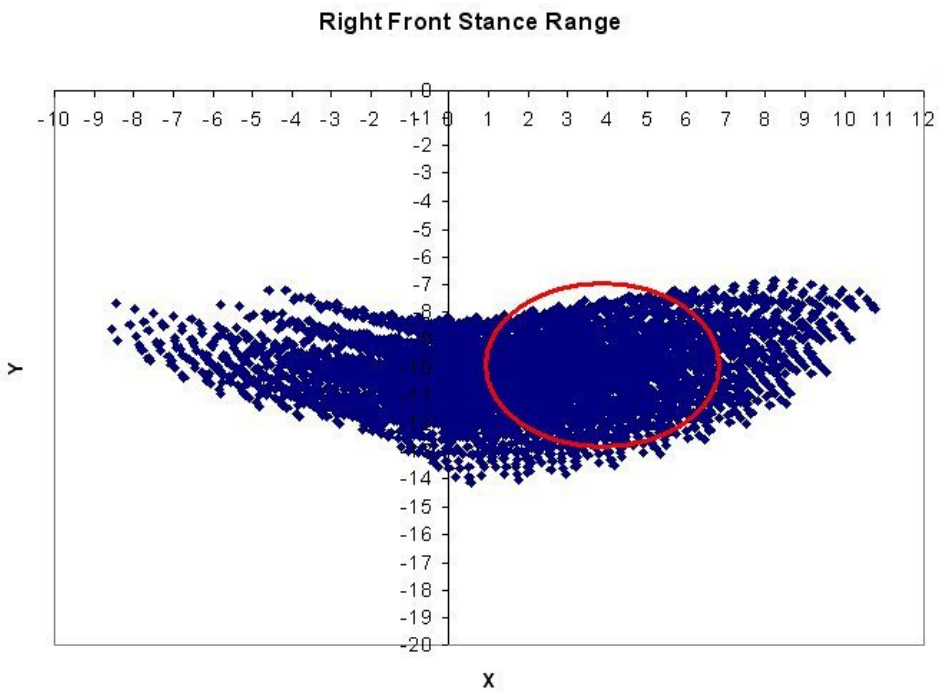


Figure 10.3: Workspace of the front right leg (from Jong-ung Choi).

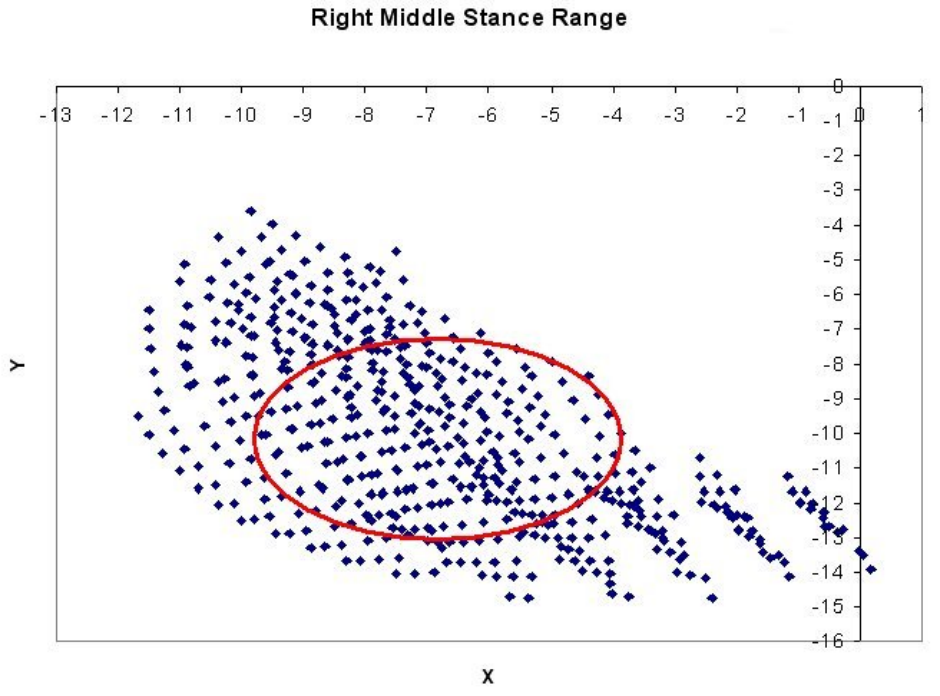


Figure 10.4: Workspace of the middle right leg (from Jong-ung Choi).

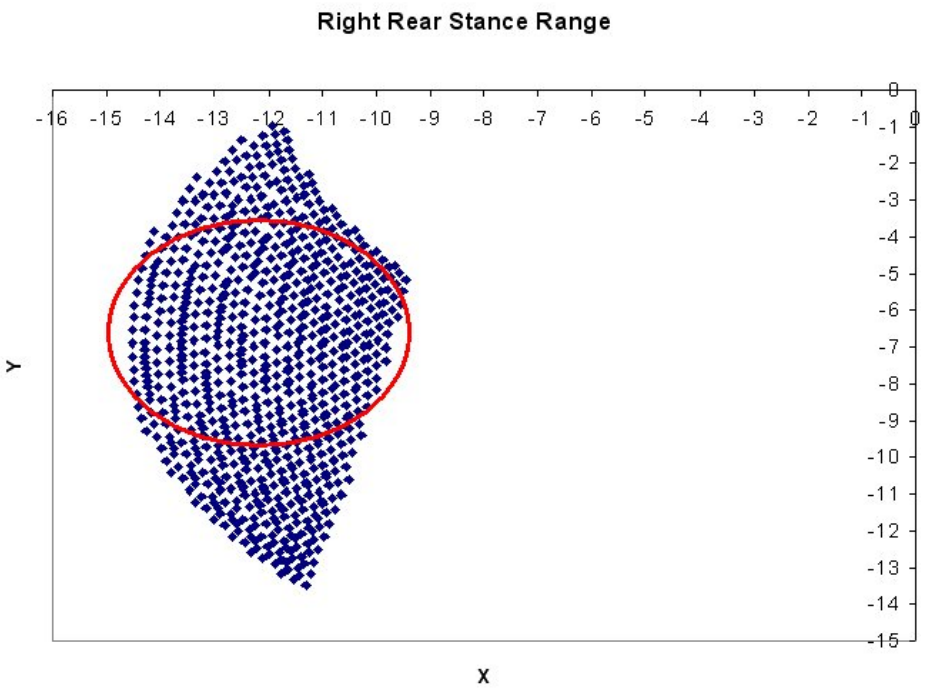


Figure 10.5: Workspace of the front right leg (from Jong-ung Choi).

Motions of the feet still followed the same pattern described in the previous chapter, beginning stance at the AEP and moving to the PEP before initiating swing; however although the AEP maintained a set distance from the center of the workspace, it was possible for the direction of motion to be offset by some angle from the body axis of the robot. It should be noted that this angular offset in no way effects the operation of the Cruse controller, the same network is capable of coordinating leg motions regardless of foot trajectory. Nor is this motion in any way based on behavior of the animal; however, it is a potentially useful feature that can easily be derived from the controller, and that serves as a useful basis for development of a turning algorithm.

10.4 Turning

Straight line motion is of course of only limited use for a legged robot; any sort of robust, adaptable walking will also require the ability to turn, and it was desired to integrate this feature into the Cruse controller. This was accomplished by extending the previously described algorithm that produced angular translation to each foot; that is, each foot would be allowed to move at an individually assigned angle. As long as these angles were selected so as to produce rotation about the same center, the robot could be made to turn. For simplicity, it was decided that the center of rotation would lie at the intersection of the center of two rear leg's workspace and the body axis (figure 10.6), as in some cases of cockroach turning, this appears to be the center of rotation (Mu et al 2003). From this position, the direction of travel for each foot to produce rotation could be approximated by a line perpendicular to a line connecting the workspace centers of the middle feet and the center or rotation. Although this angle would not be the exact angle throughout the entire motion, it provided a reasonably close and computationally efficient estimate. Because the feet were not all at the same distance from the body, their speeds also needed to be adjusted relative to each other to produce a proper turning motion; this was simply done by multiplying by the ratio of the radius to the foot versus the radius to the rear (closest to center of rotation) foot.

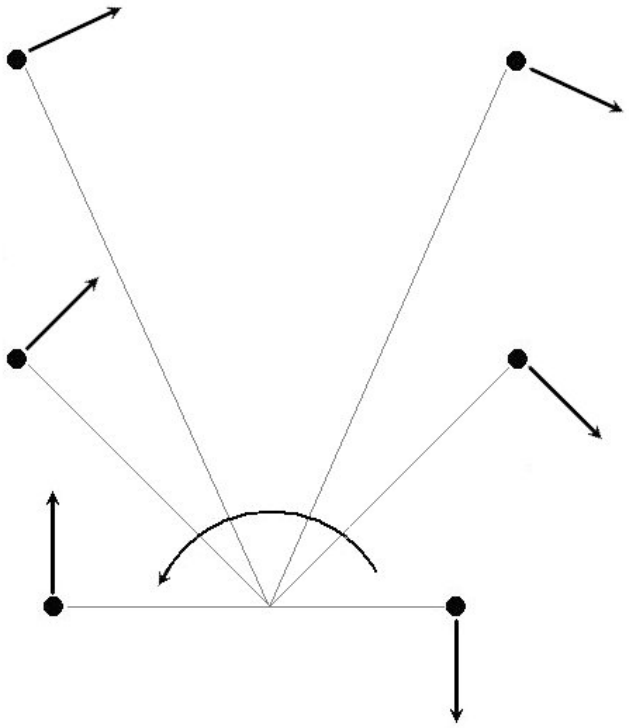


Figure 10.6: Direction of foot motion required to produce turning.

A second speed ratio was introduced to the network representing the desired speed of turning. This functioned like the previously described translational speed ratio; however, it produced motion at the specified angles for turning. This variable could be adjusted from negative one to one, with positive values causing counter-clockwise rotation. Also like the translational motion speed ratio, a ramp-up phase was introduced for turning to ensure that a stable gait was maintained. Using these features, the Cruse controller could be made to generate stable gaits as in translational motion. Although there were discernable patterns over a range of speeds, these were not the regular wave and tripod type gaits generated in translational motion. In addition, the range of speeds was less than that of translational motion, with stability being lost at a speed factor of approximately 0.1 when using a time step of 0.05

seconds and a swing speed of 10 inches per second. This was a result of the feet moving at very different speeds during rotation; the rear legs—which were the slowest—were set to the base speed ratio, and as a result the front and middle moved at higher speeds. The front feet specifically were moving too fast at higher speed ratios to maintain a stable gait, as they were required to move at three times the speed of the rear legs to maintain ground contact without slipping.

In another, albeit less biologically correct, incarnation of the turning controller, the center of rotation was placed on the centerline of the middle legs. Because the radii from the center of rotation to the feet were closer in value, this version was able to attain much higher turning speeds: up to a speed factor of 0.6 using a time step of 0.05 seconds and a swing speed of 10 inches per second. In addition, in this situation the regular translation gaits, such as wave and tripod, manifested themselves. The slower maximum speed than translational motion was still attributed to the disparity in foot speed between legs.

10.5 Planar Motion

To achieve true planar motion—translation in any direction as well as rotation—the two above algorithms needed to be combined. This was done by first calculating the desired speed and direction for both the translational and rotational motions. These vectors were then broken into component vectors in the X and Y directions, the corresponding components of which were added, allowing for the actual desired vector (speed and direction) of each foot to be calculated. The end result was that by specifying translational speed, translational angle, and rotational

speed, each leg was given an independent speed and direction which, combined with the other legs, would produce the desired translation and rotation of the body.

It would of course not be possible to produce stable gaits with both the translational and rotational speed factors at high values, as the resultant speeds would undoubtedly be greater than the swing speed. To determine the stable ranges of the controller, the simulation was run with varying translational speed factors and the rotational speed factor was increased until stability was lost. This produced some unexpected but ultimately understandable results (figure 10.7). Most interesting in this figure is of course the fact that stable rotational speed factors quickly decrease, then begin to rise again. The cause of this is the disparity in foot speeds, as the front feet in specific are at a much greater distance from the center of rotation than the rear legs. As the translational speed increases, the relative speed difference between the feet decreases when the component speed vectors are summed. The result is that higher translational speeds actually allow for higher rotational speeds.

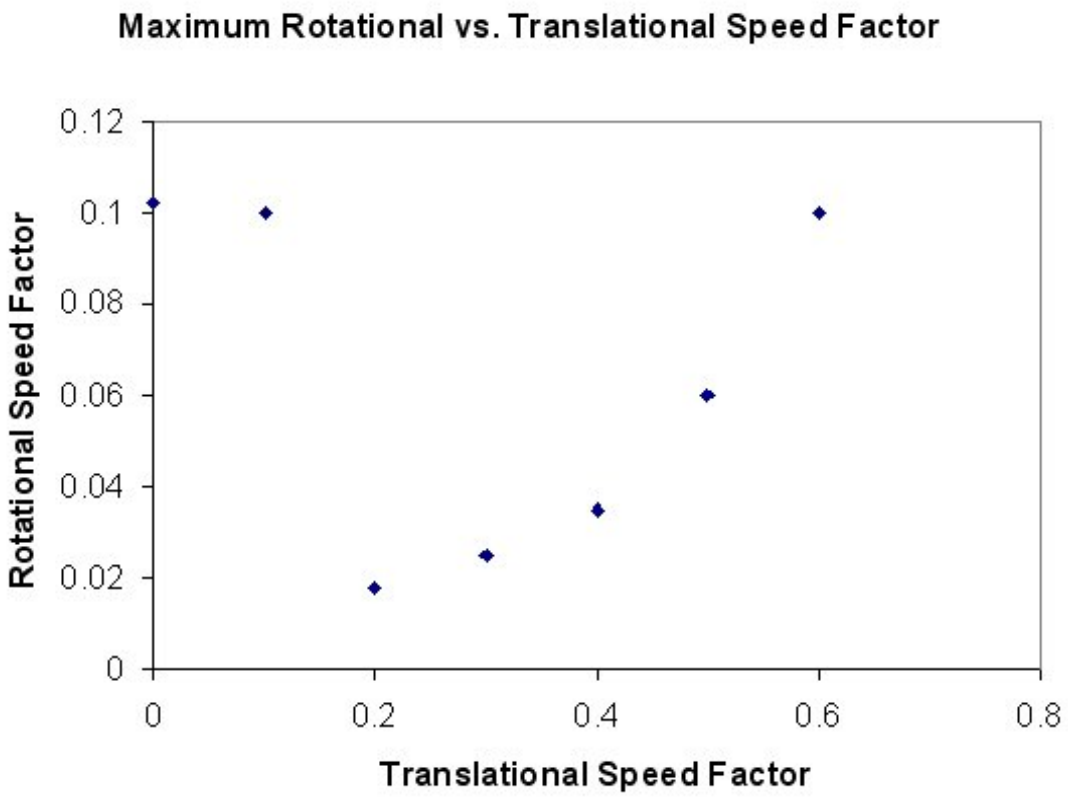


Figure 10.7: Maximum speed factor relationship for a stable gait.

Chapter XI: Final Results

11.1 Divide and Conquer

Before the operational results of Robot V can be detailed, it is important to note that this was a collaborative effort, and could not have been accomplished without significant effort from two other individuals: Jong-ung Choi and Brandon Rutter. As the robot was made to walk, each foot moved through a series of desired points, as defined by the Cruse controller. An inverse kinematics model was then used to determine the necessary joint angles to achieve these positions; in the cases of the front and middle legs this was by no means a simple task, as these legs were kinematically redundant, meaning that an infinite number of solutions existed to achieve most foot positions. This problem was overcome with the addition of a cost function that attempted to minimize joint deviation from the center of its range of motion. For each time step, it was necessary to measure actual current joint position and determine valve duty cycles to either inlet or exhaust air from actuators depending on deviation from the desired position. This reading of sensors and determination of valve duty cycles was performed by the low level controller. Jong-ung Choi created the inverse kinematics solution for Robot V based upon the solution by Nelson (2002) for Robot III while Brandon Rutter was responsible for software development of the low level controller (based on the system developed by Nelson for Robot III) and user interface. The following results may seem incomplete, and in some instances are, but due to the separation of tasks relating to the control system of the robot, much of the work required to actually produce closed loop walking was not

done by the author of this dissertation. Further results will be forthcoming in Choi's dissertation.

11.2 Sensor Calibration

To produce a usable signal, each angle had to be calibrated to determine the angle or pressure associated with a given output. Pressure sensor calibration was fairly simple, as conditions for each sensor, and thus the output of each sensor, were almost exactly the same. Prior to their installation, a number of pressure sensors had been tested, and found to have a highly repeatable output-to-pressure relationship. After the sensors had been installed on the robot, the valves were opened, and supply pressure was varied manually using a regulator, and confirmed with a gauge attached to the air line. By collecting multiple pressure/voltage data sets, it was possible to determine a relationship between the pressure and output signal of the sensors. The calibration was then confirmed by pressurizing actuators to known levels, and comparing this to the sensor output. The sensors were found to perform remarkably well, and suffered from very little noise.

Calibration of the angle sensors was a more complex task, because the range of motion of each joint and output of individual sensors varied greatly. To accomplish this, an auxiliary angle measuring device, consisting of two bars meeting at a single joint containing a calibrated potentiometer (figure 11.1) was used. This device was attached to one member of a joint, and the joint was manually moved through its range of motion. Simultaneously, the angle sensing device was also moved, with care taken to ensure that its two links remained parallel to the components of the joint, thus moving through the same angle as the joint. Using this

process, a large number of datum points were collected (ranging from 10,000 to 30,000 points) and a linear relationship between sensor output and joint angle was determined using a MatLab script written by Brandon Rutter.

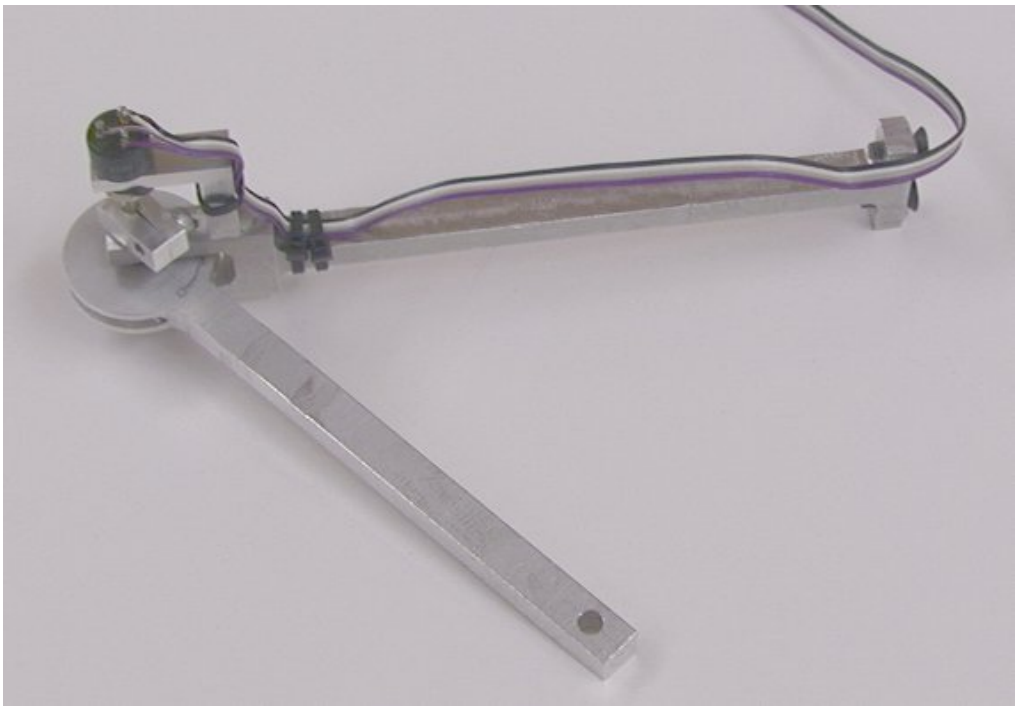


Figure 11.1: Angle sensor calibration device.

Calibrations made in this manner were not always of the greatest quality, with errors due to human error in not keeping the links of the measuring device parallel with joint components, hysteresis and creep of the sensors, and the attempt to use a linear equation to define the relationship between joint angle and sensor output. This last source of error could have been avoided by using a higher order polynomial; however, a linear relation was used because in most cases the data suggested that the relationship was nearly linear, and use of lower order equations would greatly reduce the computational requirements placed on the controller. Although linear relationships were not an exact fit, any fit with a RMS error of less than five degrees

(the accuracy necessary to keep the foot within a two inch radius of its desired position) was assumed to be acceptable, and all sensors but the beta joint sensors performed to this level.

Sensor output was measured by an A/D card over a range of 0 to 255, and most angle sensors were deemed suitable, with RMS errors calculated as being less than 5 units. Significant problems were found in the rear and front beta sensors however, which suffered not only saturation problems, but errors on the order of twenty degrees—over half the range of motion of these joints. Although multiple attempts were made to rectify this by modifying the sensor mounts, all were unsuccessful. To solve this problem, potentiometers were attached to these joints and calibrated in the same manner as the flex sensors. The potentiometers demonstrated the expected linearity and repeatability of such devices. Later, the same modification was made to the middle legs.

11.3 Walking On Air

With the sensors calibrated to within reasonable limits, the next goal was to demonstrate the capability of generating smooth, coordinated, feedback driven motions of the legs. Although this situation was very much like walking, it had the notable difference, and advantage, of no ground reaction forces. This in effect removed the possibility of sudden perturbations in required actuator loads as legs came into or out of contact with the ground. These tests were very encouraging (figure 11.2), and although the desired motions were not met in all cases, observations of the robot's behavior served as the impetus for many of the modifications detailed in Chapter VI. In brief, many of the necessary changes were small adjustments to

actuator lengths to remove slack or adjust ranges of motion. Some larger changes, such as the re-design of the front leg beta assemblies, were also motivated by the performance of the robot in this stage of testing.

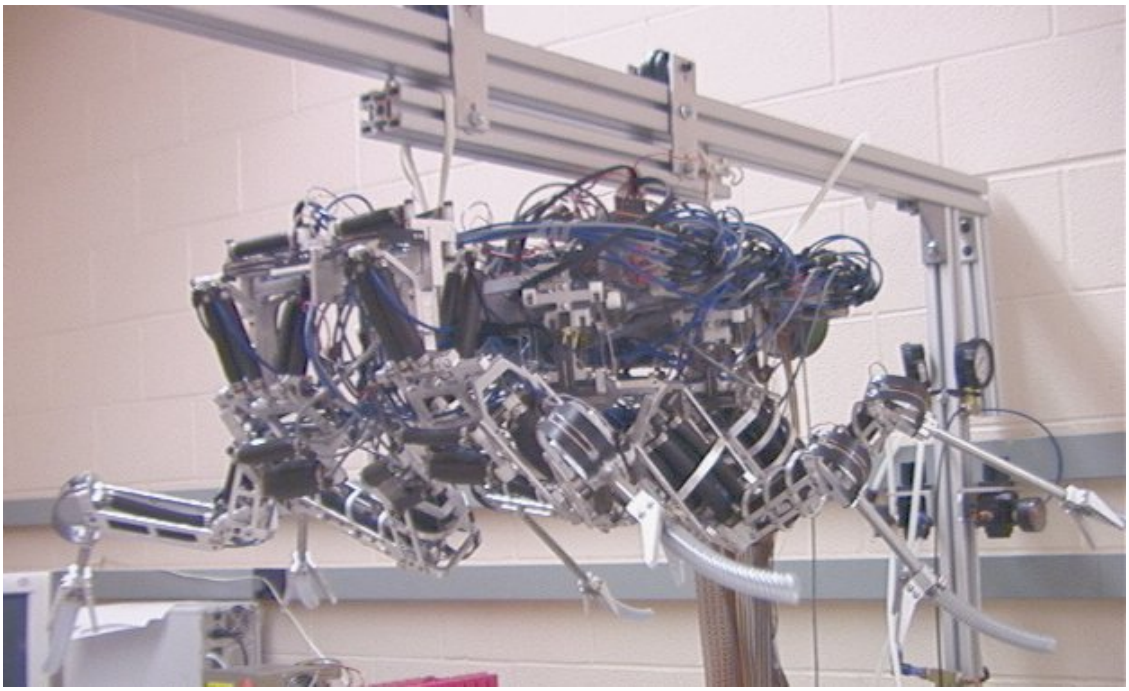


Figure 11.2: Robot V during one of the "air walking" tests.

11.4 Initial Walking Attempts

Once the air walking had advanced to a stage where the controls and capabilities of the robot were thought to be sufficient, Ajax was lowered onto a treadmill for its first unsupported, closed loop attempts at walking. While the overall results of this first try were by no means spectacular, they were quite encouraging. The robot was able to stand and maintain a proper posture prior to walking. The locomotion itself was less than stellar, but the robot did move forward, and was able to shift between tripods without falling, and for a first attempt, this was very encouraging. The greatest problem with this attempt at walking seemed to be not so

much mechanical, but control based; specifically that the feet did not rise high enough during swing to effectively clear the ground. The result was that the feet dragged, and the robot seemed to shuffle. Still, forward motion was achieved, a number of improvements were identified that would hopefully produce better locomotion.

A second attempt at walking, complete with a number of modifications to the robot and the footpaths, showed little improvement. During the attempted locomotion, there were obvious problems with the coxa-femur joints of the middle legs dragging on the ground, and inhibiting locomotion. It was believed that this was potentially the result of a number of errors. First, it was possible that the middle legs were not able to sustain the necessary loads to maintain a tripod stance in their given position; note however that using an open loop controller the robot demonstrated the ability to maintain a tripod. Secondly, the inverse kinematic model used to form the foot paths did not account for coxa-femur joint position, so nothing had been done to ensure that this collision did not occur. Finally, although a Cruse Controller was used to generate the foot paths, nothing was done to ensure that the starting position of the feet were correct. Just as initial conditions were critically important to ensuring stability at the beginning of Cruse Controller operation, so they should also be important to ensuring stable walking on the actual robot. This lead to the belief that a separate algorithm would be needed to achieve a proper standing posture before walking was attempted. In addition to these issues, it appeared as though the claws on the middle feet were inhibiting locomotion, so they were temporarily removed.

11.5 Isolating Problems

Because a number of problems had presented themselves during the initial attempts at walking, it was decided to take a step back and try to isolate them. To

accomplish this, the robot was again placed on the gantry, but lowered so that the feet were in contact with the ground. In effect, this would allow walking algorithms to be tested without the need for the robot to support its full weight (figure 11.3).

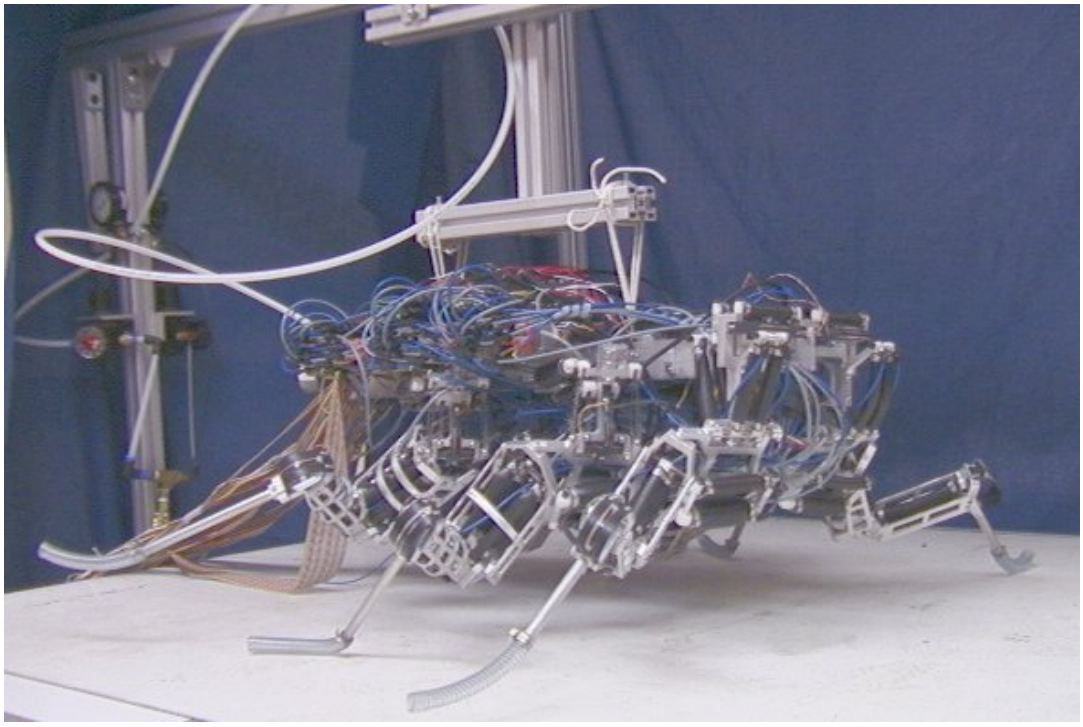


Figure 11.3: Robot V walking while partially supported by a gantry.

The initial attempt at walking in this configuration was fairly successful, and after some adjustment of footpaths (specifically the middle legs), removal of the claws from all of the legs, and a lengthening of the front leg tibias, the robot demonstrated movements similar to a cockroach. This was further improved by replacement of the rear claws, as well as one of the two claws on each front foot. The reasoning behind the removal and replacement of the claws was that while they did add a beneficial offset to the robot's height and increased friction between the feet and the treadmill, they often interfered with the swing motions, especially on the

middle legs. The current configuration seemed best after trials with a number of different configurations.

11.6 Standing

Further attempts at walking without supports were still unsuccessful, although somewhat improved. At this point, the root problem seemed to be the initial conditions of the robot—that is, it was not starting in a proper stance, and the following motions were thus nowhere near what were desired. Specifically bad were the middle legs, which began with their beta joints far too flexed, and were never able to achieve a desired position. All of this suggested that the starting conditions must be achieved with some level of accuracy before walking can begin; this is indeed the same lesson learned in the Cruse Controller simulations. The practical solution to this is to create a standing algorithm for the robot before attempting walking, in which the robot first raises itself to a maximum possible body height, then places its feet one by one in the necessary positions to initiate walking. This was done with Robot II with good results (Espenschied, 1995). While this process has no root in animal behavior, it must be remembered that the Cruse controller only functions well in steady-state conditions.

11.7 Additional Walking Attempts

Additional attempts were made at walking using a standing algorithm before initiation of walking. Although the results showed some improvement, there was obviously much work required before the locomotion could be considered satisfactory. Most problematic was that the feet remained in the same plane parallel to the body axis throughout their stance and swing, instead of extending the tibia

during swing, which would greatly increase ground clearance. As a result, the feet had difficulty clearing the ground during swing. This was further compounded by the fact that the kinematic model did not take the claws and semi-rigid member into account. This resulted in the controller not lifting the feet high enough. In total, many of the problems seemed to result from path planning of the feet and joint angles. As the robot has demonstrated the ability to stand, support its weigh in a tripod stance, and produce cockroach-like air walking motions, it is believed that improved walking will be possible as the controller becomes more advanced.

Chapter XII: Power Plant Development

12.1 Compressed Air Requirements

For Robot V to be fully autonomous, one of the most critical elements of the design would be the air compressor that powers the braided pneumatic actuators. Although a number of other issues also need to be solved, specifically advancements to the controller and electronics this did not preclude taking the first steps to create an onboard compressor for a robot of this size, as such devices are not currently available.

To understand the requirements placed on Robot V's power plant, a rough estimate of the air consumed by the robot while walking was made. The approximate volume of all actuators unpressurized is 88.5 cubic inches, and the approximate volume of all actuators fully inflated is 265 cubic inches. Given these physical properties, and on the assumption that the actuators are cycled from zero to one hundred psi-gauge every two seconds (this frequency is a reflection of the desired walking speed) the robot can be expected to consume roughly 132.5 cubic inches of 100 psi air per second. If this air is assumed to be compressed adiabatically, then using the simple gas equation:

$$P_1V_1^{\gamma} = P_2V_2^{\gamma}$$

where P and V are the absolute pressure and volume at a given state, and γ is the gas constant of air, equal to 1.4, then 575 cubic inches of atmospheric air are consumed every second. It should be noted, however, that this is a very conservative over estimate; the swing actuators see very little load, and thus do not require very high pressures during normal operation, and in addition, actuators can often be kept

partially inflated during operation. Also, during walking or running, the stance actuators should not be completely deflated during each cycle so they can store energy during the first half of stance and release it in the second half (Alexander 1988). Therefore, the actual air flow requirements are expected to be much lower.

These requirements represent one of the worst possible cases for air compression: high pressure coupled with a relatively high flow rate. Rotary air compressors, such as turbines, are capable of very high flow rates, but provide little compression, normally on the order of 30 psi. Positive displacement compressors, such as pistons, do just the opposite; they are capable of high compression ratios but low flow rates.

12.2 Power Plant Design

The most critical property of the air produced by the compressor is the pressure. At lower pressures, the actuators are not capable of producing sufficient force for stance, let alone locomotion, and the robot effectively becomes useless. Flow rate, on other hand, only effects the speed at which the actuators may be cycled; lower flow rates mean the robot must move slower so that its compressor can keep up with the demands of the actuators. Although this slower movement is not desirable, it at least results in a viable robot. Therefore, the logical choice of compressor is one of the positive displacement type.

Air compressors can be found that are driven by either electric (AC or DC) motors or internal combustion engines. Most commercially available compressors are far to heavy for use onboard such a small robot, with even the lightest weighing

approximately twenty pounds, not including batteries or fuel. Therefore, it was deemed necessary to design a new compressor for this application.

An internal combustion engine driven compressor was chosen for the simple reason of weight—not only do IC engines weigh significantly less than electric motors of comparable power, but the energy density of the liquid fuel is superior to that of batteries. The goal of this part of the project was to show that a small, lightweight engine could be converted to an engine-compressor and deliver high pressure air. Therefore, an existing four cylinder engine was chosen and modified. It was not expected to provide the air flow needed, but it would be a valuable learning experience.

The engine chosen for this application was the O.S. FF-320 "Pegasus" (Figure 12.1). This four-stroke four horsepower engine, which is intended for use onboard large RC aircraft, was designed with the weight restrictions of flight in mind; with a total weight of 4.8 pounds, it is within the range anticipated for Robot V's power system. In addition, the "flat four" layout allows two opposing cylinders to be converted to compressors, resulting in well balanced, smooth operation. Like most RC airplane engines, this one runs on glow fuel. The fuel is ignited by glow plugs, which are heated initially by batteries, and maintain their temperature as a result of the combustion in the cylinder. The fuel itself contains a non-combusting lubricant, and as a result, the entire engine is lubricated by the combustion residue produced during operation.

O.S. FF-320 SPECIFICATIONS

Displacement: 0.809 cu in (13.3 cc) x 4

Bore: 1.091 in (28 mm)

Stroke: 0.866 in (22 mm)

RPM: 1,800-8,500

Output: 4.1 bhp @ 8,000 rpm

Weight: 77.3 oz (2,190 g)

Figure 12.1: OS FF-320 “Pegasus”.

Theoretically, this modified engine is capable of producing air far in excess of the desired pressure. With a maximum cylinder volume of 0.92 in^3 and a clearance between the piston and head of 0.1 inches at top dead center (tdc, or position of piston at the furthest point in its stroke) and assuming adiabatic compression of the air, an air pressure of approximately 360 psi can be expected. Although this pressure would not be fully realized due to factors such as this not being a truly adiabatic process and leakage throughout the system, it does suggest that the goal of 100 psi should be readily attainable.

Although this power plant was expected to produce air at the pressures desired, the flow rate was anticipated to fall short of the target value. From the

specifications previously listed, each stroke of a piston is expected to move 0.82 cubic inches of atmospheric air per stroke (this is greater than the listed displacement due to changes in the clearance between the piston and head as well as some incursion of the original head into the cylinder); with two pistons operating at 100 hertz (6000 rpm) a total flow rate of 164 cubic inches of atmospheric air could theoretically be produced. Although this is less than half the target value, it must be recalled that the target value was intentionally an overestimate of air use. Furthermore, the addition of an air storage tank will allow the robot to function at high frequencies for limited periods of time, followed by a "rest" period where air pressure is recovered.

12.3 The First Iteration

The original heads were removed from the two cylinders closest to the output shaft (where the propeller would be mounted if this where being used on an airplane). These cylinders were chosen for use as compressors for easier cooling, and thus more efficient compression once a fan was attached to the shaft, as well as keeping the two remaining power cylinders closer to the carburetor. The fuel lines to the compressors were removed and blocked. Although the camshaft opened the valves in the original heads at the desired positions for inlet and exhaust of air—inlet at minimum excursion and exhaust at maximum excursion—the engine is a four stroke engine, which means that, amongst other things, the valves open only on two of the four piston strokes. As a result, if the original camshaft and valves were used, half the time the air in the cylinders would be compressed and re-expanded without being exhausted. This inefficiency was unsuitable for use onboard a robot.

A new pair of heads (Figure 12.2) were designed and manufactured from 6061 T-6 Aluminum with an Emco 55 CNC milling machine. Inserted into each head were two steel-balled check valves made by Deltrol. These valves were mounted in opposite directions so that one would serve as an inlet and other as an exhaust. To keep the valves flush with the inner surface of the head, Aluminum spacers were manufactured and placed between the flat surface of the valve used for tightening and the outer surface of the head (Figure 12.3). To ensure a good seal, an O-ring was placed between the spacer and the surface of the head.

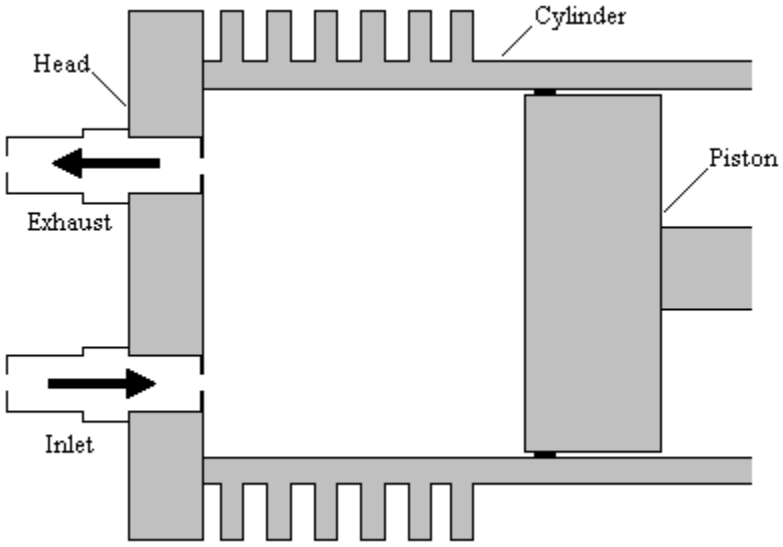


Figure 12.2: First cylinder head design.

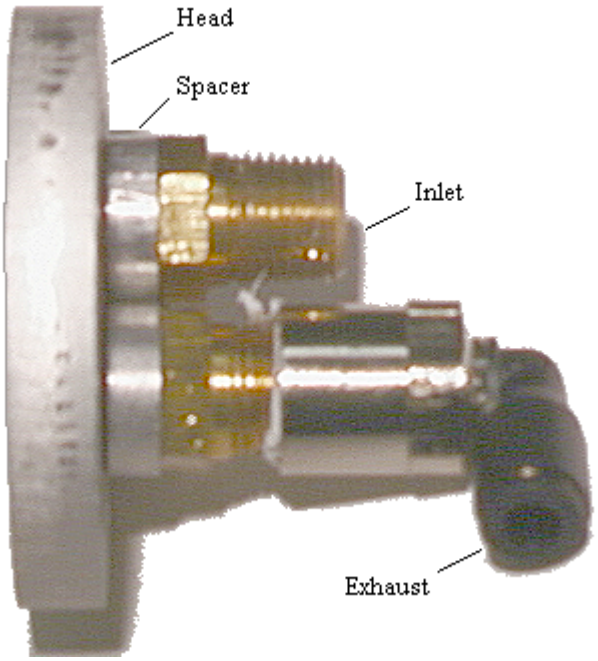


Figure 12.3: Cylinder head components.

One of the greatest initial concerns with this design was the reaction time of the check valves. To ensure that they would be able to open and close at the desired rates and pressures (of primary concern on the inlet, which at most could experience a difference of atmospheric pressure), a check valve was placed in-line with a solenoid valve operating at frequencies ranging from 1 to 100 hertz with a 50 percent duty cycle; pressure was varied from 0 to 100 psi in 20 psi increments as well as an additional test at 10 psi. A flow meter was used to measure the amount of fluid passing through the valve. Although there was significant drop-off at higher frequencies, the flow rate stayed above 82 in^3/s (just below the theoretical output of each cylinder, equivalent to 2.85 cfm). This flow rate was observed for the entire pressure range, except for pressures of 10 and 20 psi, which were insufficient to open

the valve completely except at very low frequencies, thus restricting flow (Figure 12.4).

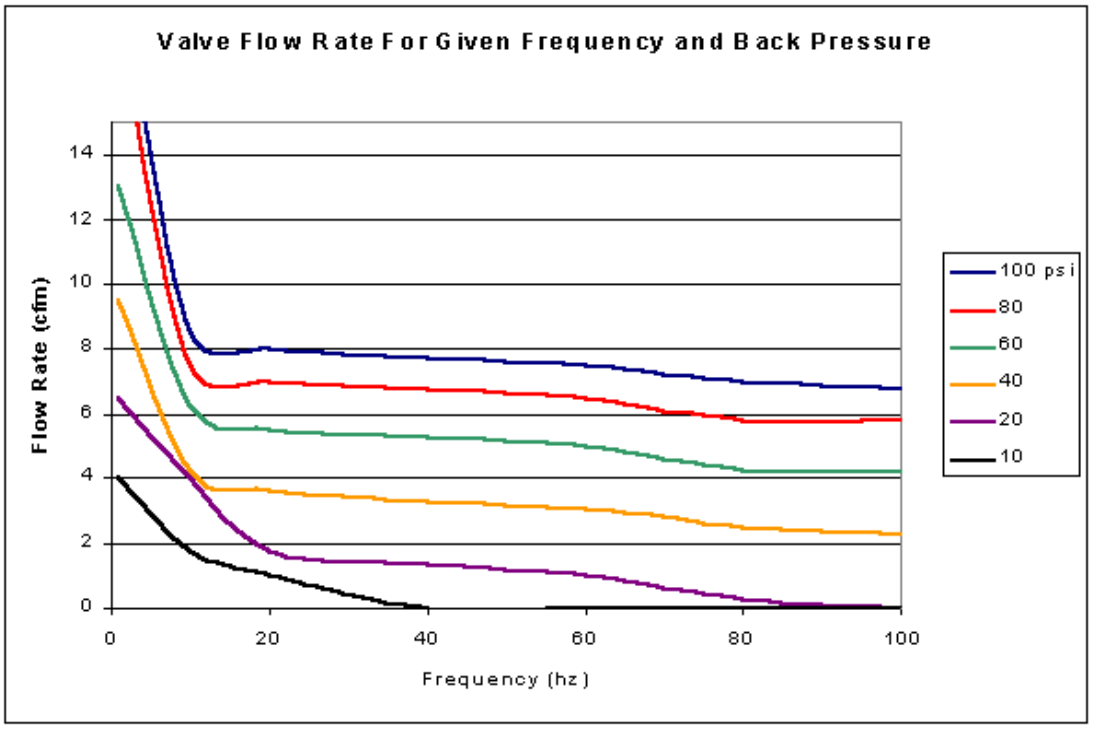


Figure 12.4: Flow rates of Deltrol check valves.

Of course, the back pressure of the system and the frequency of the solenoid valve also affect the measured flow rate. To more accurately display the effect of the check valves, the above data can be normalized by dividing by flow rate of the system without the check valve; i.e. just the pressure reservoir, solenoid valve, and flow meter (Figure 12.5).

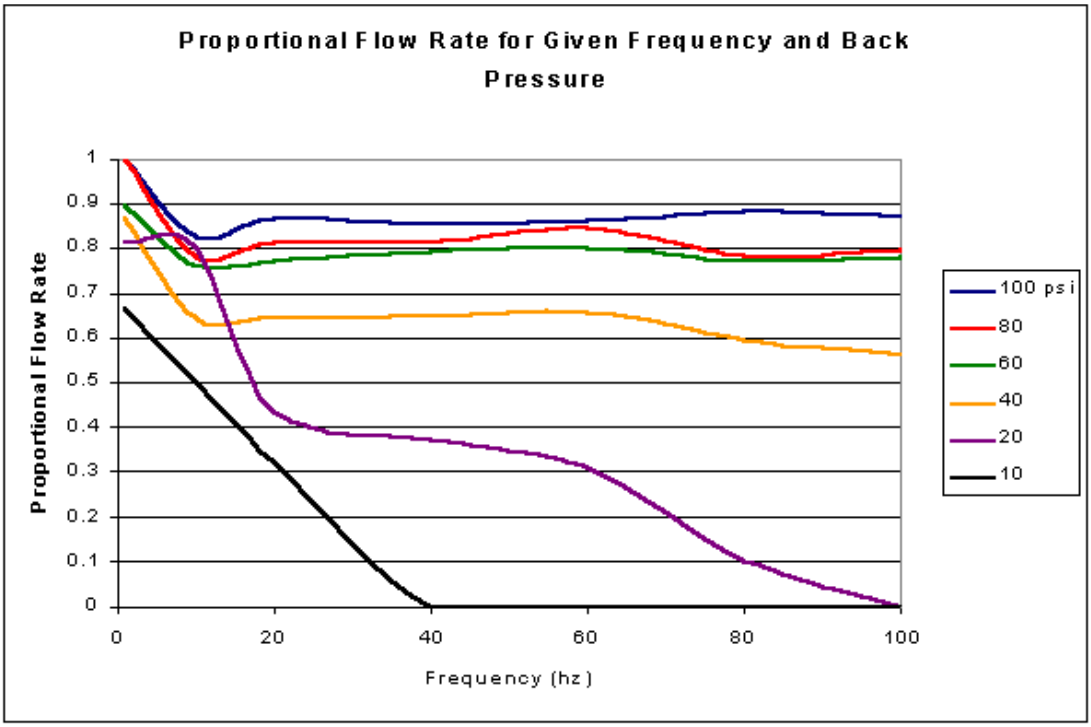


Figure 12.5: Normalized flow rates of Deltrol check valves.

From the above data, it was clear that this design would be limited by the rate at which air could be inlet to the compressors. Because the inlet valve could experience at most a pressure differential equal to atmospheric pressure (atmospheric outside and total vacuum inside) and the engine was expected to run at approximately 100 hertz, it was quite possible that the entire system would become starved for air.

This design exhibited two problems during testing. First, and most importantly, it was not able to attain pressures above approximately 60 psi, and this pressure was only achieved when the compressor was running at low speeds; at high speed, it produced approximately 30 psi. In both cases the flow rate was significantly lower than expected. In addition, they cylinders became very hot, with temperatures rising high enough to melt the nylon tube that carried air away from the exhaust.

Both of these problems can be attributed to a single design error. Because there was a gap of approximately 0.075 inches between the head and piston at top dead center, a portion of the compressed air was not exhausted from the cylinder. As the piston began its return stroke, this air re-expanded within the cylinder, keeping the pressure above atmospheric for much of the cycle; this in turn kept the inlet valve from opening until the very end of the return stroke. The result was that very little air was brought into the system on every cycle, and thus very little air was exhausted. The compressor was able to achieve higher pressures at low speeds because there was more time available for air to fill the cylinder. Because this volume of air was continuously compressed and re-expanded, it quickly began to heat up, resulting in the melting of the nylon air tube.

Of additional concern was the lubrication of the compressor cylinders. Because fuel was no longer being burned in these cylinders, it was possible that oil would not be sufficiently distributed to them, resulting in increased friction, a temperature rise, and possibly eventual seizure. Immediately after operation, the compressor heads were removed and the cylinders were inspected for lubricant. Fortunately, it was found that they had remained sufficiently lubricated by the oil produced in the combustion process in the remaining two cylinders.

12.4 The Second Iteration

Due to the failings of the first compressor head, a second one was designed. This head was recessed further into the cylinder, allowing a clearance of only 0.005 inches between the head and the piston. Not only did this reduce the amount of air that remained in the cylinder on the return stroke, it also increased the maximum

pressure the compressor was capable of achieving (note that this did not reduce the amount of air moved, which is dependent strictly on the displacement of the piston.) To further ensure that sufficient air was provided to the system, the inlet valve was removed from the cylinder head. Due to the design of the valves, the inlet valve provided a small volume in which air could remain during operation, which in turn contributed to the volume of air that remained trapped in the cylinder. The inlet valve also did not have sufficient bandwidth to completely fill the cylinder with air, as is suggested by figure 12.5. Inlet of air was instead provided by slitting the cylinder just above the piston at its minimum stroke; not only did this remove the need for a large pressure differential to open the inlet valve, but it also provided a larger area through which air could fill the cylinder, albeit less time to do so (Figure 12.6). This is the mechanism that many two-stroke engines use for the inlet of air.

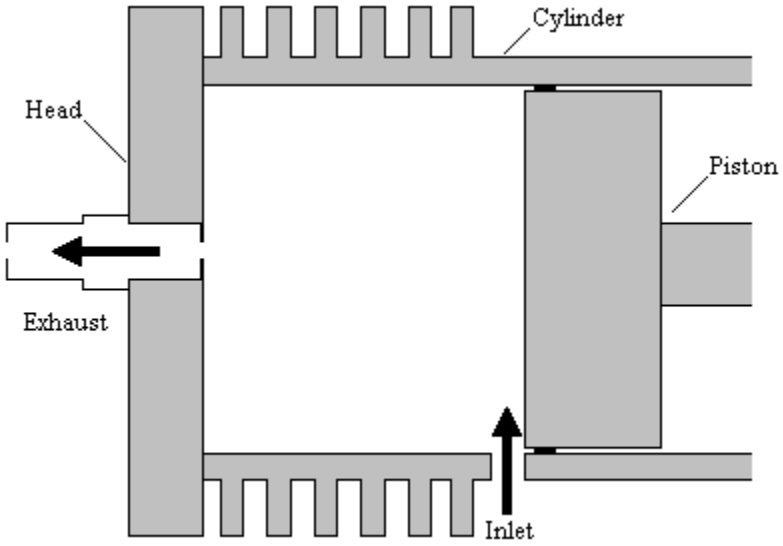


Figure 12.6: Second version of the cylinder head.

In addition to these modifications, a yoke of brass tubing was used to connect the two exhausts; this yoke also had a single quick disconnect port to attach the nylon air line that would be used throughout the rest of the robot. This was done simply to move the nylon tube further from the cylinder heads to prevent it from melting.

This design exhibited one fundamental flaw: it was difficult to start. The reduced volume of air remaining after compression quickly expanded on the return stroke of the piston, resulting in a near vacuum in the compressor. The load of this vacuum was so great that the engine would stall while starting.

12.5 Two Wrongs Make A Right

The design failures of the first two compressor heads were in many ways diametrically opposed; one did not draw enough vacuum to cause sufficient inlet of air, the other drew too much vacuum and was difficult to start. By combining these two designs, it was believed the engine could meet the design goals. The single ported heads were once again mounted on the compressors, along with the slotted cylinders. The air line immediately above the cylinder hear was split with a “T”, one line leading to a check valve, the other to a manually operated ball valve. This ball valve allowed the compressors to be open to the environment during engine start-up, and then closed for operation once the engine was running at its operational speed (figure 12.7). Although a two ported head could have also been used to produce this configuration, space limitations due to the sizes of the valves made this design unfeasible.

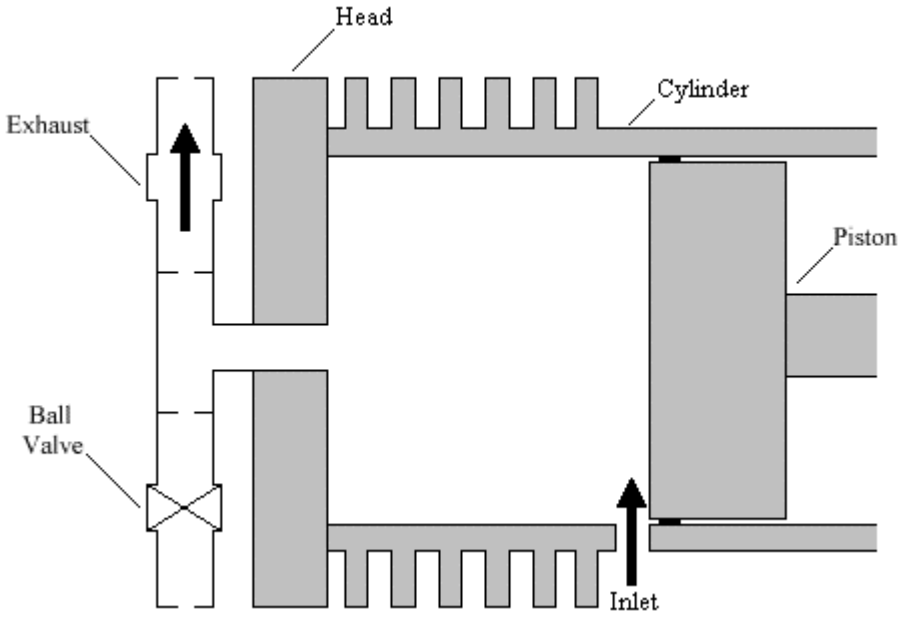


Figure 12.7: Finalized cylinder head design.

In operation, this configuration was shown to produce pressures in excess of the desired 100 psi (a relief valve set to open at 100 psi was used to confirm pressure output, meaning that maximum pressure was not recorded) and a flow rate of 300 cubic inches per second (Theobold 2003). Although this is below the estimated 575 cubic inches of air per second needed, it is within the same order of magnitude, especially when one keeps in mind that the required volume was an intentional over-estimate. At the very least, this demonstrates that this sort of converted engine/compressor is viable for use onboard a robot, and the technology could reasonably expanded using existing products (OS also makes a radial five cylinder engine, two of which could be assembled together in a engine/compressor combination) to provide power to pneumatically actuated robots in this size range.

Chapter XIII: A Comparison With the Insect

13.1 Ranges of Motion

Upon completion of the robot and calibration of the sensors, the range of motion of the joints was again measured, this time using the sensors to determine angle; the previous measurements had been made visually, and were thus much more prone to error. These angles were compared to those of the animal while walking in a tripod gait; data were collected using high-speed video (Watson and Ritzmann 1998, Watson at al. 1999, Bachmann 2000).

Table 13.1: Robot V actual and desired ROM, animal ROM.

JOINT	ROM	DESIRED ROM	ANIMAL WALKING
Front Leg			
γ	-7 - 24°	-20 - 20°	-50 - 10°
β	-73 - -32	-80 - -50	-90 - -60
α	12 - 50	20 - 60	35 - 65
c-f	-21 - 20	-20 - 30	20 - 60
f-t	-28 - 43	-30 - 45	-50 - 20
Middle Leg			
β	-88 - -40	-80 - -50	-95 - -70
α	27 - 42	20 - 60	45 - 55
c-f	3 - 41	-10 - 40	20 - 60
f-t	-23 - 46	-30 - 45	-40 - 10
Rear Leg			
β	-46 - -25	-40 - 0	-25 - -5
c-f	-8 - 39	-10 - 40	20 - 60
f-t	-37 - 54	-30 - 45	-30 – 30

In many cases the robot's ROM met or exceeded those of the animal, but a few joints do deserve to be addressed specifically. In general, the robot was found to

function better in a higher stance than the animal; this was in large part done to prevent limb sections from dragging on the ground. As a result of this, many joints, especially the C-F joints, were adjusted so that they would extend further than on the animal. Prior to these adjustments, the robot had a tendency to bring the C-F joint into contact with the ground, and although this behavior is exhibited by the animal, it was not believed to be desirable walking for the robot. The front leg γ joints fall about ten degrees (25%) short of the animal; the structure of the robot however will allow reconfiguration of the joint with minimal difficulty should this present a problem in generating walking motions. Although the middle α joint demonstrated far less motion than desired, it should be noted that it still exceeds that of the animal, which itself demonstrated very little motion. The β joint of the rear leg provides a very different range of motion from the same joint on the animal; however in walking experiments this joint has performed well. If this difference were to eventually cause problems, it might be possible to modify the joint in the same manner as the front and middle β joints were, although this may require reconfiguration of the valve and electronic assemblies mounted above the joint. Of final concern are the middle and rear C-F joints, which also fall short of the animal's ROM. In large part, especially in the case of the rear leg this was part of an intentional trade off between power and ROM; the moment arms of the insertions of these actuators could be reduced to provide a greater range of motion, but this could be at the cost of the very important capability of these joints to provide weight support and propulsive forces.

13.2 Mass Distribution

Another interesting comparison between the animal and the robot is the distribution of mass among both limb segments and the body, both as a strict comparison of segment weights and in terms of moments of inertia of segments. Data used for cockroach masses was derived from dissected *Blaberus discoidalis*, and the moments of inertia about the center of mass were derived approximating coxae segments as triangular plates and all other segments as sleder rods; in all cases, uniform density was assumed (Kram et al. 1997).

Table 13.2: Mass distribution of the cockroach and Robot V.

	Cockroach			Robot V		
	Base Mass (kg)	Percentage Total Mass	Percentage Leg Mass	Base Mass (lbs)	Percentage Total Mass	Percentage Leg Mass
Front Leg						
Coxa	2.00E-05	0.0083	0.6494	0.557	0.0167	0.4540
Femur	7.90E-06	0.0033	0.2565	0.440	0.0132	0.3586
Tibia	2.90E-06	0.0012	0.0942	0.230	0.0069	0.1874
Middle Leg						
Coxa	3.80E-05	0.0157	0.6485	0.708	0.0213	0.4615
Femur	1.40E-05	0.0058	0.2389	0.570	0.0171	0.3716
Tibia	6.60E-06	0.0027	0.1126	0.256	0.0077	0.1669
Rear Leg						
Coxa	4.80E-05	0.0199	0.6234	0.714	0.0214	0.4565
Femur	1.80E-05	0.0074	0.2338	0.578	0.0174	0.3696
Tibia	1.10E-05	0.0046	0.1429	0.272	0.0082	0.1739
Body	2.25E-03	0.9311		28.98	0.8701	
Total	2.42E-03			33.3		

Possibly the most noticeable difference between the animal and the robot is that more mass is distributed to the legs of the robot, and especially to the distal segments of the

animal is autonomous, effectively carrying its own onboard power supply and controller, elements that are offboard on the robot; their addition would obviously increase the percentage of mass contained within the body of the robot. Secondly, the mass of the tarsi were not included in the cockroach data as they have no real analog onboard the robot; however, the claws of the robot do factor in as part of the mass of the tibia, thus increasing its overall mass. Although these differences are relatively small, they do offer the robot some advantages, most notably that the more proximal stance actuators are required to support a lesser fraction of the total mass of the entire system than their biological counterparts. However, this does come at the price of the distal leg segments requiring greater relative loads during swing on the robot.

Table 13.3: Moments of inertia of the cockroach and Robot V.

	Cockroach		**Robot V**	
	Moment of Inertia (kgm^2)	**Moment of Inertia w.r.t. Rear Tibia**	**Moment of Inertia (lbs in2)**	**Moment of Inertia w.r.t. Rear Tibia**
Front Leg				
Coxa	8.80E-11	0.5867	2.3870	0.4894
Femur	3.90E-11	0.2600	2.9540	0.6057
Tibia	4.10E-12	0.0273	0.8607	0.1765
Middle Leg				
Coxa	5.80E-11	0.3867	2.531	0.5190
Femur	1.20E-10	0.8000	4.881	1.0008
Tibia	3.40E-11	0.2267	1.887	0.3869
Rear Leg				
Coxa	8.10E-11	0.5400	5.42	1.1113
Femur	1.70E-10	1.1333	5.735	1.1759
Tibia	1.50E-10	1.0000	4.877	1.0000

As previously stated, all moments of inertia are calculated at the center of mass, for rotation about an axis parallel to the axis of segment rotation. In the case of the robot's coxae, the most distal axis was used; α for the front and middle legs, and β for the rear legs. Given the sensitivity of moments of inertia to perturbations (this value is dependent on the square of distances) the level of correlation between the values after scaling with respect to the rear tibia is remarkable. The most notable difference, occurring between the front tibia, is most likely once again due to the inclusion of the claws on the robot's tibia, and exclusion of the tarsus from the biological model. The variations of the middle and rear coxae are most likely due to their width on the robot (to accommodate a pair of 20mm actuators) and their supporting ribs.

13.3 Joint Angles During Locomotion

A final, and very telling, comparison between the robot and the animal is one of the behavior of joints during locomotion (figures 13.1-13.12); however, this must be entertained with the following two caveats: first, the robot was intended to have a slightly higher (two inch, or thirty three percent higher) stance than the animal for reasons previously discussed, and second, the robot's feet were moved through a trajectories that do not exactly model those of the animal. These desired joint angles were not driven by a model of the actual animal during locomotion, the controller only attempted to move feet to desired positions generated by the Cruse controller. Furthermore, this joint angle planner attempted to keep joints near the center of their total ROM in the case of kinematically redundant limbs. Although there are indications that animals use a similar strategy, there is no guarantee that this is in fact

the mechanism used, or that the center of the total ROM is kept as the center of excursion in such situations. It should also be noted that during the supported walking, the front legs became caught, resulting in the undesirable angle data in the graphs. This was a result of failure to adequately clear the ground during swing, resulting in the foot being dragged.

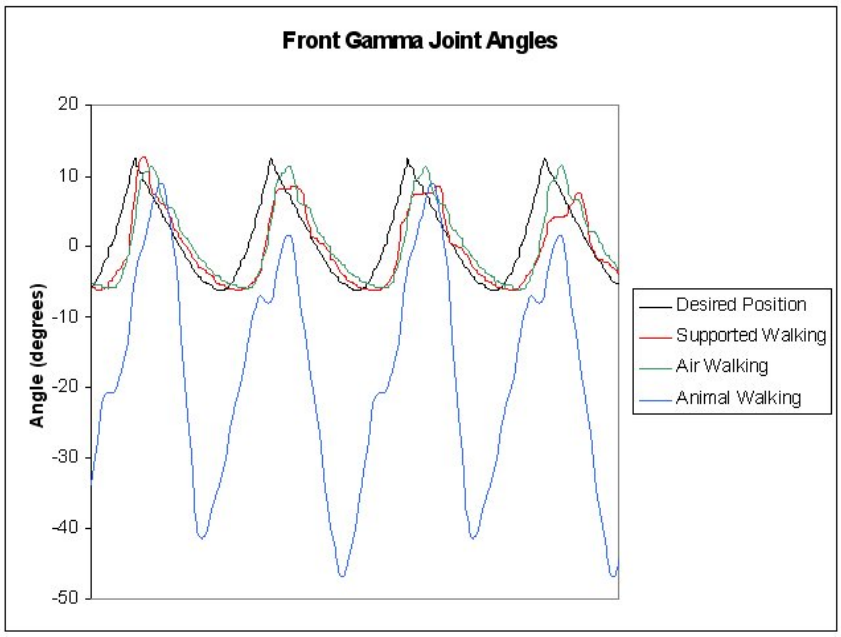


Figure 13.1: Front gamma joint angles.

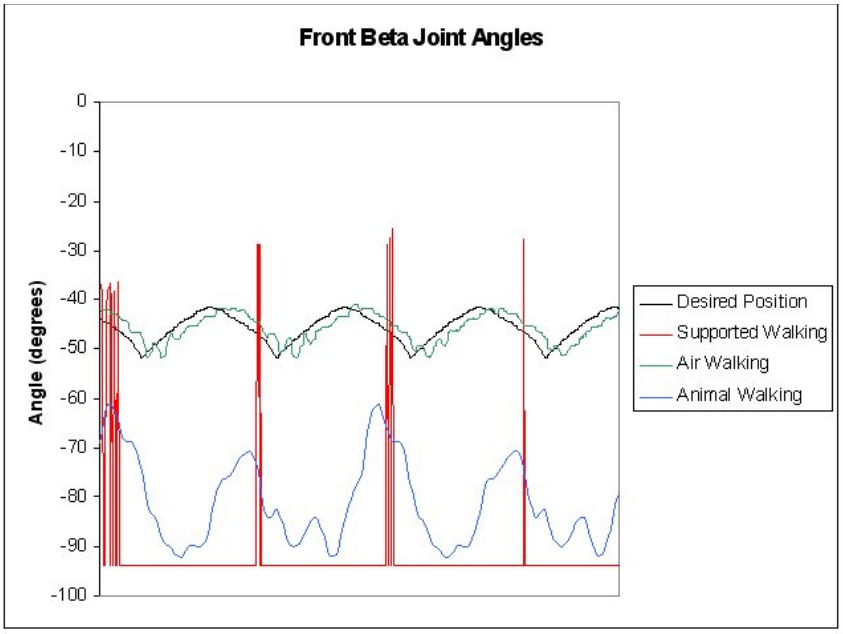


Figure 13.2: Front beta joint angles.

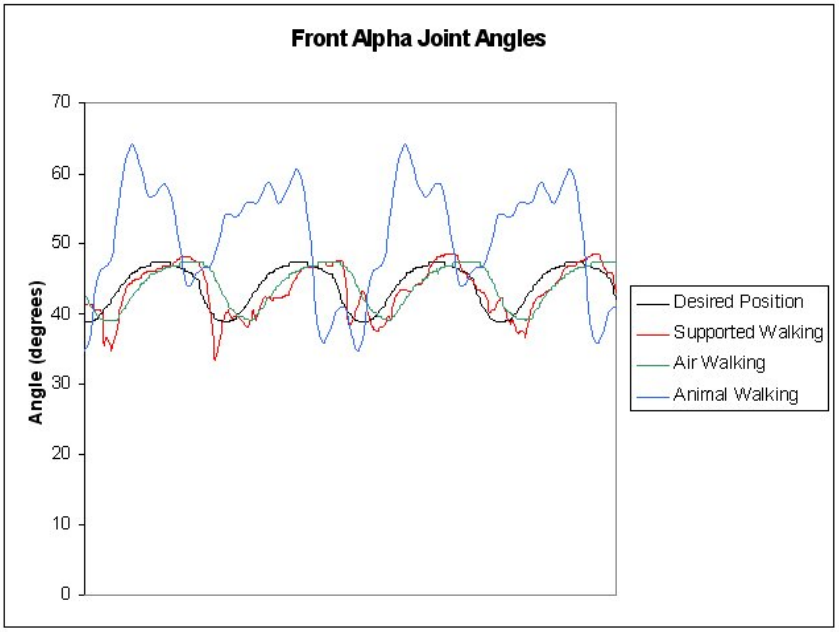


Figure 13.3: Front alpha joint angles.

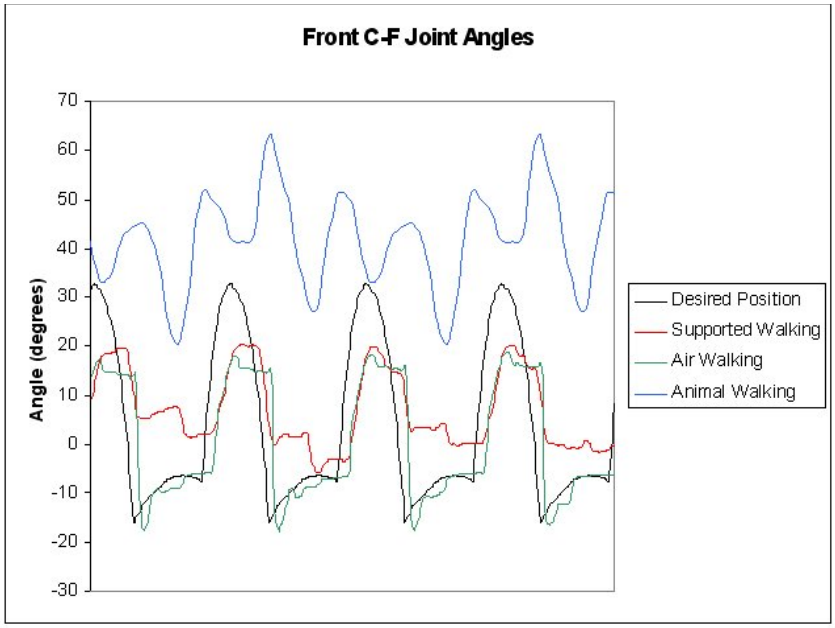


Figure 13.4: Front C-F joint angles.

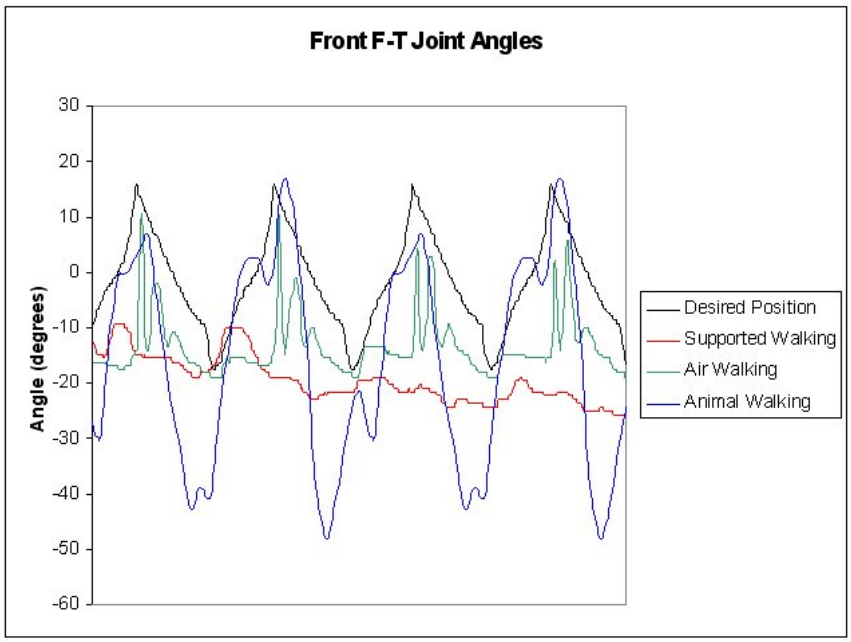


Figure 13.5: Front F-T joint angles.

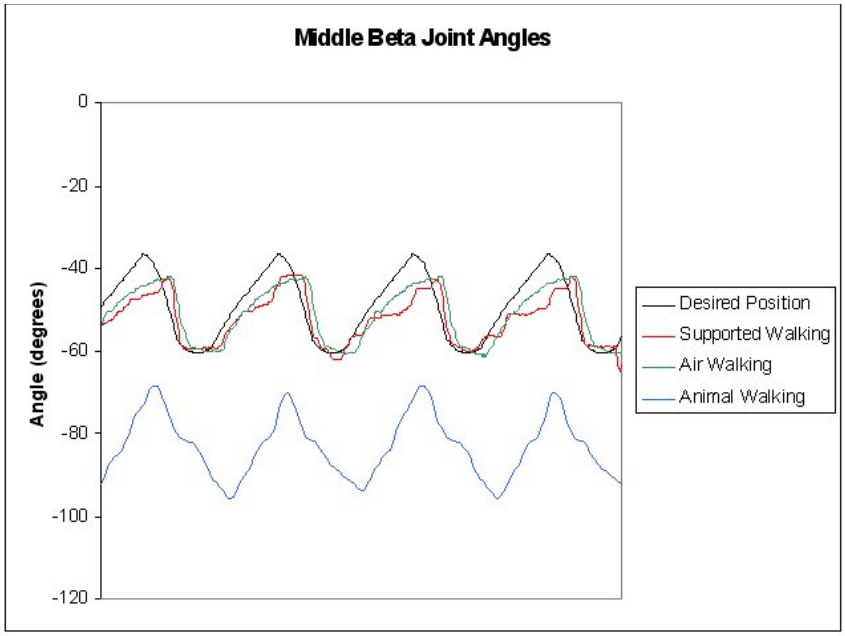


Figure 13.6: Middle beta joint angles.

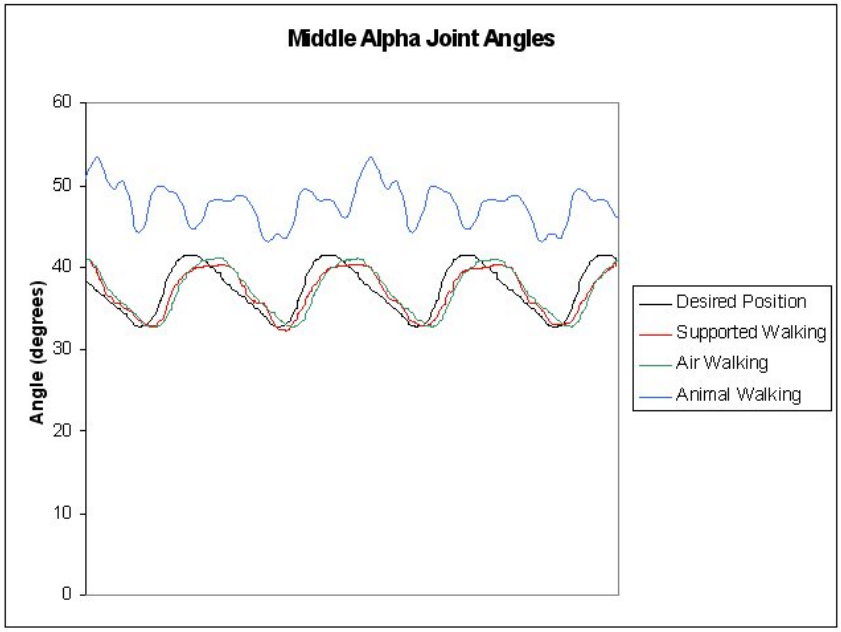


Figure 13.7: Middle alpha joint angles.

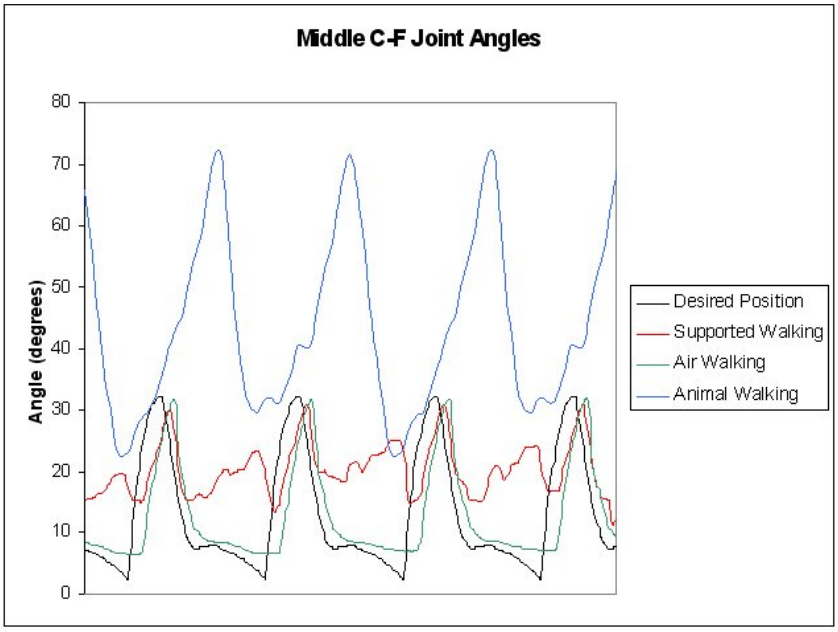


Figure 13.8: Middle C-F joint angles.

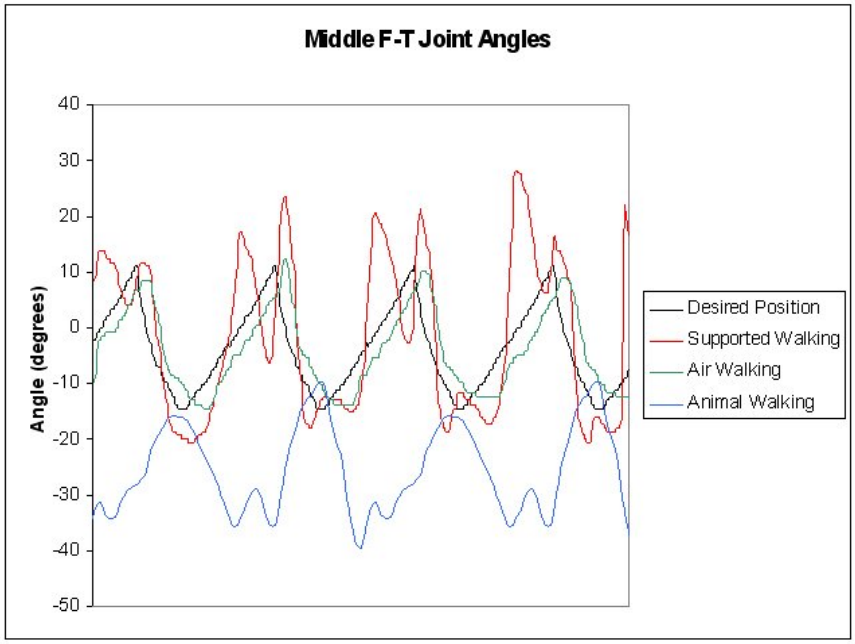


Figure 13.9: Middle F-T joint angles.

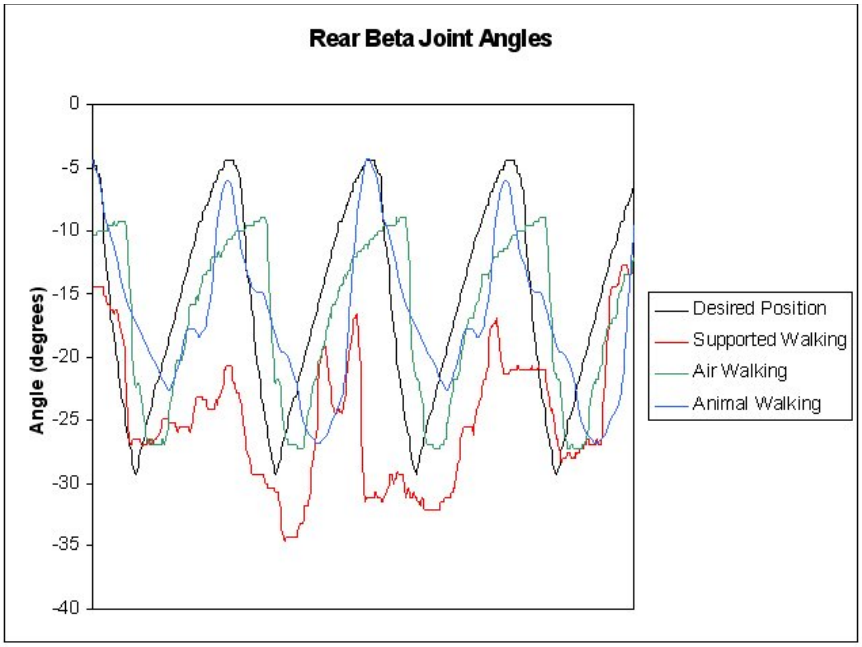


Figure 13.10: Rear beta joint angles.

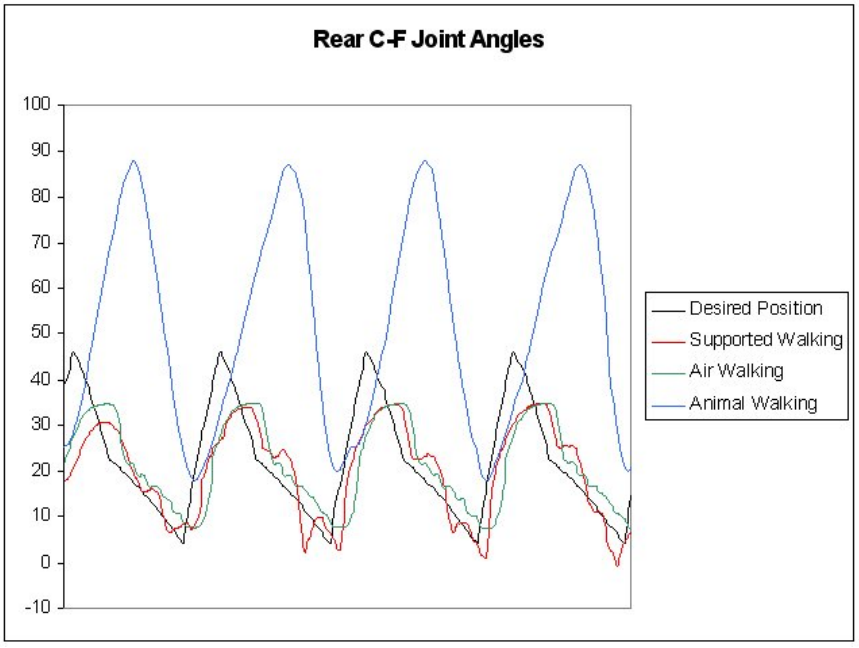


Figure 13.11: Rear C-F joint angles.

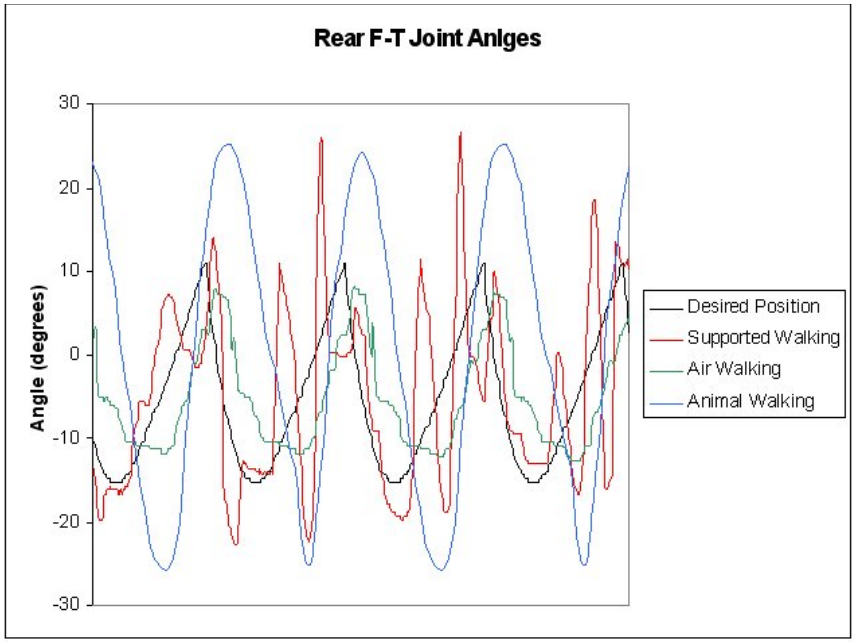


Figure 13.12: Rear F-T joint angles.

Chapter XIV: Future Work and Lessons Learned

14.1 Limitations of Robot V

Before discussing potential paths of research following Robot V, it is important to identify the strengths and weaknesses of the robot, and the lessons learned. First, and foremost, the underlying motivation of this project was to demonstrate the benefits of non-conventional actuators with muscle-like properties and a biologically inspired design (both mechanical and control). In this regard, the robot was a success; it was able to make rudimentary walking motions with the use of only an open loop controller. The robot was able to lift payloads in excess of ten pounds, and locomote carrying payloads of five pounds.

Perhaps the greatest limitation of the robot was that it was not autonomous; this limitation was a result of both power and control limitations. Although autonomy was not the goal of this research, some work was done to support future efforts, specifically the development of the power plant detailed in Chapter XII. Unfortunately, compressed air is not the only power requirement; electrical power is also required by the sensors (low power, but requiring a clean signal) and the valves (higher power, but not as restricted in clarity).

14.2 Recommendations for Future Designs

There are of course always modifications and improvements that can be made to any design, and Robot V is no exception. Although the specifics of a future design may vary greatly from what has been presented here, there are some basic design principles which should be considered in the future.

The β joints (with the possible exception of the rear β) demonstrated the least satisfactory performance; in specific, they did not have the desired range of motion. The simplest solution to this would be to lengthen the associated actuators, but it should be noted this would force an increase in spacing between the legs. In light of the complications caused by the implementation of cable tendons in the β joints, it may be reasonable to pursue an entirely new joint architecture. Please note however that the most readily available option—mounting the actuators for this joint vertically—may present immediate problems in the height of the robot.

The β joints also, as detailed in Chapter VIII, were not able to support the use of the bend sensors used elsewhere on the robot. This should not be considered a failing of the sensors themselves, but of the design. The initial development of the robot had focused on inclusion of potentiometers for joint angle sensing, and it was only after the design was nearly complete that the flex sensors were identified as a replacement. As a result, many joints that were designed specifically to integrate potentiometers were instead retrofitted with bend sensors. Because of the small range of motion of the β joints, coupled with an architecture not best suited for the flex sensors, signal output was poor. If the entire joint were to be redesigned, it would not be difficult to design it to include these sensors; the quality of the C-F and F-T joint angle sensors should be considered testimony to the effectiveness of these sensors.

The Festo actuators, which were in so many ways key to this research, performed remarkably well. Although there were issues with them kinking, these were addressed as detailed in Chapter VII. One additional design improvement that should be strongly considered in the future however is the inclusion of some

mechanism to make small adjustments to the lengths of the actuators or the spacing of the mounting points. In assembly of the robot, it was found that even small miscalculations of actuator length (0.25 inches on a 5 inch actuator for example) can have a very detrimental effect on the range of motion of a joint. And, actuator lengths are not simple to calculate. Although the distance between mounting points can easily be calculated, the exact stroke of each actuator set in an opposing pair is not, as it depends on the properties of the actuators, and manner in which the end-plugs were attached. In assembly, the most effective means of actuator mounting was found to be to attach one set of actuators—generally ones that defined the more important limit of motion when fully extended—fully pressurize this set, and calculate the length of the opposing actuator set based on this position. Although this method was generally effective, a small amount of adjustment after completion of this process would have been useful. In addition to initial mounting, this was also found to be an issue on those occasions when an actuator blew out an end-plug; the range of motion of a joint could change noticeably after repairing the actuator, due simply to the small variations in length after reassembly.

Prior to now, the Biorobotics lab has paid little attention to implementing actively controlled feet on robots. The focus has begun to shift more toward feet with development of more advanced designs on the Whegs robot series, but not to the extent that may be warranted. Certainly there was no complex foot design on Robots I-IV, and although foot design was considered to some extent on Robot V, it is most likely a very important avenue of research that has been ignored for far too long. It is very likely that Ajax's performance would be significantly improved with the

addition of advanced, passive feet instead of the claw/semi-rigid member design currently in use. Although the semi-rigid member can provide high levels of friction, it is not retractable in the way that a cockroaches foot it, and this can impede the foot's motion during swing.

14.3 Controller Improvements

The Cruse controller presented here does an excellent job of generating a continuum of stable walking gaits, but it is limited to this task. Ultimately, a robot capable of navigation over rough and irregular terrain would be desirable, and to accomplish this task, a much more advanced controller would be necessary. This is a monumental task, requiring not only mechanisms for obstacle and surface detection such as force feedback from all actuators (already implemented with the pressure and position sensors) and exteroceptors but also a control system equivalent to that found in an insect head. All told, there is an almost limitless range of improvements that could be made to the controller, and indeed this subject is broad enough to provide a life time of research.

14.4 Inverse Kinematics

The Cruse controller has demonstrated that a biologically inspired gait controller can be not only highly effective, but also computationally efficient and conceptually simple. All of these features are highly desirable in all aspects of the controller, and thus the question arises: can this biological approach be applied to other aspects of the controller? One of the most computationally intensive elements of the controller is the inverse kinematics that define foot position, especially in the kinematically redundant front and middle legs. In contrast, this same process is

carried out in many much more complex biological organisms on a regular basis and in some cases at very high frequency. Undoubtedly animals have some sort of kinematic self model, and indeed some take a long time learning to walk; however, once this skill is learned, the necessary inverse kinematics seem to be carried out with great efficiency. We are currently performing the inverse kinematics solution in a neural network in software. These circuits can probably be implemented as analog circuits to provide relatively simple and low power solutions.

14.5 Posture Architecture

As noted, Robot V is capable of carrying a significant payload, but much or all of this capability would be lost were the robot to be made autonomous. In contrast, insects are known for their ability to carry many times their own weight. Why this disparity? This change in relative capability is a result of the scaling of the system. Consider that the force (and thus torque) produced by most actuators—including muscle, BPAs and conventional fluidic systems—scales with the cross sectional area; that is, an actuator twice as large will have four times the cross sectional area, and thus produce four times the force. Mass, on the other hand, assuming density is approximately the same, scales with the cube of the size increase; thus a structure twice as large would have eight times the mass. The result of this is that an increase in size very quickly produces mass increases far in excess of the force increase of the associated actuators. This is compounded by the posture of the insect morphology, as the sprawled posture of Robot V demands very high torques to be applied at some joints. To put all of this into context, one could consider the animal world; for the most part, the sprawled posture of insects is replaced by a more upright

one in larger animals (some notable exceptions exist, such as alligators, but this trend is nonetheless clear) which is capable of bearing greater weights without demanding increased joint torques, at the sacrifice of turning capability (Biewener, 2003). The purpose of Robot V is to model cockroach behaviors for locomotion research therefore its morphology had to be similar to its animal model. But, if the goal is to develop a mission capable robot of similar size, it would be strongly recommended that it take this same approach to morphology, emulating mammals of roughly the same size in the hope of being able to sustain larger payloads, and hopefully making autonomy more viable.

14.6 Walking Speed

Although the robot has performed wonderfully in a variety of ways, it was not capable of achieving the desired walking speed. This was due to the volume of air required to inflate the larger 20mm actuators. A fully inflated actuator has approximately twice the diameter of an uninflated one, and with a stroke of 25%, this means an actuator triples in volume when fully pressurized. Stepping frequency was therefore limited by actuator response time, which was in turn correlated to how fast air could be moved into and out of the largest actuators; in fact, the flow rate during exhaust had previously been shown to be the limiting factor of actuator response (Kingsley 2001). One proposed method to alleviate this problem had been the addition of filler material to an uninflated actuator; however, this met with only marginal success due to the degree by which internal volume of the actuators changes (at best, the air requirements would be reduced by one third). The more practical, and easily implemented, solution would be to increase flow rates into and out of the

actuators. This could be achieved either by using larger valves rated for higher flow rates or by connecting multiple valves to individual actuator sets, and activating these valves simultaneously. It should be noted that some foundation for this second option is already in existence, as the inlet manifolds have nine ports, of which only eight are used. Regardless of the method used to increase walking frequency, there is an associated increase in robot mass, and the effect of this should be considered in any change implemented.

14.7 Conclusions

This text has presented the design methodology for both the physical structure and leg coordination mechanism of a biologically inspired, cockroach based robot. The use of muscle-like actuators in conjunction with a valve configuration that allows air to be trapped has been shown to provide a number of benefits; specifically, the actuators and valving scheme provided an exceptional level of passive stability and a means by which not only joint position can be controlled, but also joint stiffness. These properties of the actuators allow them to in fact serve as an important part of the control system as well, in effect acting as reflexes on the robot. The inter-leg coordination mechanism is based on the behavior of the walking stick insect (Cruse 1991) and provides a robust, highly adjustable means by which a continuum of gaits may be achieved. Although the robot has not yet demonstrated unsupported agile walking, it has demonstrated all of the mechanisms necessary for this: it is capable of supporting its own weight (as well as some payload) in a tripod stance, the joints have sufficient range of motion to move the feet through sufficient ranges of motion for locomotion, and the onboard sensors provide accurate data at a high enough

frequency to sufficiently control position. What remains is development of the necessary controllers: fine-tuning the foot trajectories, adding a posture controller, and adding a stretch reflex and positive load feedback.

Appendix A: A Sample Cruse Controller

The following is a copy of the Cruse controller written for Robot V. It should be read with the caveat that it was written by a mechanical engineer, and while the logic may be easy to follow, in no way is it suggested that this is efficiently coded.

```
/*
 * This is Dan's pathetic atempt to write a Cruse Controller for Ajax,
   easily adaptable to other robots
 */

#include <iostream>
#include <stdlib.h>
#include <stdio.h>

/*-------------------------------------------------------*/
/* Function "sign" to determine + or - value of a variable */

int sign (float val)
{
        if (val>=0)
                return (1);
        else
                return (-1);
}

/*-----------------------------------------------*/



/*--------------BEGIN MAIN PROGRAM-------------*/

int main()
{
        using namespace std;

        /* Initialize variables */



        float j;

        int duration;                   /* time limit of operation */

        float timestep;                 /* timestep variable */

        float speed;                    /* speed ratio; stance speed as fraction of swing speed */
        float swingspeed; /* swing speed for all legs, all speeds */
        float stancespeed;/* stance speed that is adjusted for given speed */
```

```
                float gaitspeed;    /* desired gait speed */
                float startspeed;   /* intital speed used in startup ramping */
                float gaitramp;             /* ramping constant for startup */
                float angle;                /* angle of motion with respect to body axis */
                float turnrate;             /* desired turn speed */
                float turnspeed;    /* turn speed that is adjusted for given speed */
                float turn;                         /* speed for turning; will vary from -1 to 1 */
                float steplength;   /* Nominal distance from AEP to PEP */
                float frontrad;             /* radius from center or rotation to front foot workspace
center */
                float midrad;               /* radius from center or rotation to middle foot workspace
center */
                float rearrad;              /* radius from center or rotation to rear foot workspace
center */

                /* Network mechanisms--same side and cross side */

                float mechanism1;
                float mechanism2;
                float mechanism5;

                float crossmechanism1;
                float crossmechanism2;
                float crossmechanism5;

                /* Network scaling factors as defined by Ken */

                float mechanismscale1;
                float mechanismscale2;
                float mechanismscale5;

                float crossmechanismscale1;
                float crossmechanismscale2;
                float crossmechanismscale5;
                float contbias;

                /* Value of each mechanism for each leg */

                float leftfront1;
                float leftfront2;
                float leftfront5;
                float leftfrontcross2;
                float leftfrontcross5;
                float leftmid1;
                float leftmid2;
                float leftmid5;
                float leftmidcross2;
                float leftmidcross5;
                float leftrear1;
                float leftrear2;
                float leftrear5;
                float leftrearcross1;
                float leftrearcross2;
                float leftrearcross5;

                float rightfront1;
```

```
float rightfront2;
float rightfront5;
float rightfrontcross2;
float rightfrontcross5;
float rightmid1;
float rightmid2;
float rightmid5;
float rightmidcross2;
float rightmidcross5;
float rightrear1;
float rightrear2;
float rightrear5;
float rightrearcross1;
float rightrearcross2;
float rightrearcross5;

/* leg states--AEP, PEP, Position, speed, contact, too many to list now */

float leftfrontaep;
float leftfrontpep;
float leftfrontpos;
float leftfronttarget;
int   leftfrontcontact; /* 1=stance, 0=swing */
int           leftfrontcount;
char  leftfrontstate;
float leftfronty;
float leftfrontend;
float leftfrontzmax;
float leftfrontground;
float leftfrontz;
float leftfrontcenterx;
float leftfrontcentery;
float leftfrontaepx;
float leftfrontpepx;
float leftfrontaepy;
float leftfrontpepy;
float leftfrontangle;
float leftfrontspeedx;
float leftfrontspeedy;
float leftfrontdir;           /*this is the final resultant direction of each leg */
float leftfrontswingx;
float leftfrontswingy;

float leftmidaep;
float leftmidpep;
float leftmidpos;
float leftmidtarget;
int   leftmidcontact;
int   leftmidcount;
char  leftmidstate;
float leftmidy;
float leftmidend;
float leftmidzmax;
float leftmidground;
float leftmidz;
float leftmidcenterx;
```

```
float leftmidcentery;
float leftmidaepx;
float leftmidpepx;
float leftmidaepy;
float leftmidpepy;
float leftmidangle;
float leftmidspeedx;
float leftmidspeedy;
float leftmiddir;
float leftmidswingx;
float leftmidswingy;

float leftrearaep;
float leftrearpep;
float leftrearpos;
float leftreartarget;
int   leftrearcontact;
int   leftrearcount;
char  leftrearstate;
float leftreary;
float leftrearend;
float leftrearzmax;
float leftrearground;
float leftrearz;
float leftrearcenterx;
float leftrearcentery;
float leftrearaepx;
float leftrearpepx;
float leftrearaepy;
float leftrearpepy;
float leftrearangle;
float leftrearspeedx;
float leftrearspeedy;
float leftreardir;
float leftrearswingx;
float leftrearswingy;

float rightfrontaep;
float rightfrontpep;
float rightfrontpos;
float rightfronttarget;
int   rightfrontcontact;
int   rightfrontcount;
char  rightfrontstate;
float rightfronty;
float rightfrontend;
float rightfrontzmax;
float rightfrontground;
float rightfrontz;
float rightfrontcenterx;
float rightfrontcentery;
float rightfrontaepx;
float rightfrontpepx;
float rightfrontaepy;
float rightfrontpepy;
float rightfrontangle;
```

```
float rightfrontspeedx;
float rightfrontspeedy;
float rightfrontdir;
float rightfrontswingx;
float rightfrontswingy;

float rightmidaep;
float rightmidpep;
float rightmidpos;
float rightmidtarget;
int   rightmidcontact;
int   rightmidcount;
char  rightmidstate;
float rightmidy;
float rightmidend;
float rightmidzmax;
float rightmidground;
float rightmidz;
float rightmidcenterx;
float rightmidcentery;
float rightmidaepx;
float rightmidpepx;
float rightmidaepy;
float rightmidpepy;
float rightmidangle;
float rightmidspeedx;
float rightmidspeedy;
float rightmiddir;
float rightmidswingx;
float rightmidswingy;

float rightrearaep;
float rightrearpep;
float rightrearpos;
float rightreartarget;
int   rightrearcontact;
int   rightrearcount;
char  rightrearstate;
float rightreary;
float rightrearend;
float rightrearzmax;
float rightrearground;
float rightrearz;
float rightrearcenterx;
float rightrearcentery;
float rightrearaepx;
float rightrearpepx;
float rightrearaepy;
float rightrearpepy;
float rightrearangle;
float rightrearspeedx;
float rightrearspeedy;
float rightreardir;
float rightrearswingx;
float rightrearswingy;
```

```
/* set time step and duration for network in seconds */

timestep = 0.05;
duration = 90.0;

/* set stance speed as a fraction of swing speed */
/* stance speed must be less than or equal to one */
/* depending on system resolution, values near 1 may not be possible*/

speed = 0.1;
swingspeed = 10;                        /* arbitrarially chosen, a little faster
than .5 hz motion */
steplength = 6;                 /* Approximates good ROM for RV. Nominal
distance from AEP to PEP*/
angle = 0 * 3.14159/180;   /* start moving forward, gotta be in radians; positive
value denotes movement left */
turn = 0.10;                            /* vary from -1 to 1; positive value
denotes left (CCW) turn */


/* startup with ramp up from wave */

gaitspeed = speed * swingspeed;
turnrate = turn * swingspeed;
startspeed = 0.0;
stancespeed = startspeed;
turnspeed = startspeed;
if ((speed != 0)||(turn != 0))
{
        gaitramp = 0.03 * timestep / 0.05;
}
else
{
        gaitramp = 0;
}

/*-------------------------------*/


/* Set Contralateral Bias */

contbias = 1.1;

/*-------------------------------------------------------------*/

mechanism1 = -1.0;
mechanism2 = 1.0;
mechanism5 = 1.0;

crossmechanism1 = -1.0;
crossmechanism2 = 1.0;
crossmechanism5 = 1.0;


/* set the network scaling factors as determined by Dan */
```

```
mechanismscale1 = 1.0325;
mechanismscale2 = 2.0;
mechanismscale5 = 7.25;

crossmechanismscale1 = 0.3;
crossmechanismscale2 = 0.1;
crossmechanismscale5 = 3.08;

/* set initial state of system */



/*-----------------------INITIALIZING--------------------------*/

/*----------------------Define Geometry-------------------------*/


/* set centers in X idrection */

leftfrontcenterx = 4;
leftmidcenterx = -7;
leftrearcenterx = -12;
rightfrontcenterx = 4;
rightmidcenterx = -7;
rightrearcenterx = -12;


/* set centers in Y idrection */

leftfrontcentery = 10;
leftmidcentery = 10;
leftrearcentery = 7;
rightfrontcentery = -10;
rightmidcentery = -10;
rightrearcentery = -7;

/* set raddii from CoR to workspace centers */
/* angles and radaii changed for rotation about rear axis, because that's how
cockroaches turn*/
frontrad = 36.25;
midrad = 21.21;
rearrad = 12;

/*------------Set angles for individual legs during turning-------------*/
/*this is defined by geometry of robot, legs move tangent at centerpoint*/
/* angles and radaii changed for rotation about rear axis, because that's how
cockroaches turn*/
leftfrontangle = 65.56 * 3.14159/180;
leftmidangle = 45 * 3.14159/180;
leftrearangle = 0 * 3.14159/180;
rightfrontangle = 114.44 * 3.14159/180;
rightmidangle = 135 * 3.14159/180;
rightrearangle = 180 * 3.14159/180;


/* set all AEPs as Jong calculated all motions are in negative direction */
```

```
leftfrontaep = leftfrontcenterx + steplength/2;
leftmidaep = leftmidcenterx + steplength/2;
leftrearaep = leftrearcenterx + steplength/2;
rightfrontaep = rightfrontcenterx + steplength/2;
rightmidaep = rightmidcenterx + steplength/2;
rightrearaep = rightrearcenterx + steplength/2;

/* set PEPs */

leftfrontpep = leftfrontcenterx - steplength/2;
leftmidpep = leftmidcenterx - steplength/2;
leftrearpep = leftrearcenterx - steplength/2;
rightfrontpep = rightfrontcenterx - steplength/2;
rightmidpep = rightmidcenterx - steplength/2;
rightrearpep = rightrearcenterx - steplength/2;


/* Set X and Y AEPs and PEPs */

leftfrontspeedx = speed*cos(angle) + turnspeed*cos(leftfrontangle)*frontrad/rearrad;
leftfrontspeedy = speed*sin(angle) + turnspeed*sin(leftfrontangle)*frontrad/rearrad;

leftfrontdir = atan2(leftfrontspeedy , leftfrontspeedx);

leftfrontswingx = swingspeed*cos(leftfrontdir);
leftfrontswingy = swingspeed*sin(leftfrontdir);

leftfrontaepx = leftfrontcenterx + steplength/2*cos(leftfrontdir);
leftfrontpepx = leftfrontcenterx - steplength/2*cos(leftfrontdir);
leftfrontaepy = leftfrontcentery + steplength/2*sin(leftfrontdir);
leftfrontpepy = leftfrontcentery - steplength/2*sin(leftfrontdir);

/*--------------------*/

rightfrontspeedx = speed*cos(angle) +
turnspeed*cos(rightfrontangle)*frontrad/rearrad;
rightfrontspeedy = speed*sin(angle) +
turnspeed*sin(rightfrontangle)*frontrad/rearrad;

rightfrontdir = atan2(rightfrontspeedy , rightfrontspeedx);

rightfrontswingx = swingspeed*cos(rightfrontdir);
rightfrontswingy = swingspeed*sin(rightfrontdir);

rightfrontaepx = rightfrontcenterx + steplength/2*cos(rightfrontdir);
rightfrontpepx = rightfrontcenterx - steplength/2*cos(rightfrontdir);
rightfrontaepy = rightfrontcentery + steplength/2*sin(rightfrontdir);
rightfrontpepy = rightfrontcentery - steplength/2*sin(rightfrontdir);

/*--------------------*/

leftmidspeedx = speed*cos(angle) + turnspeed*cos(leftmidangle)*midrad/rearrad;
leftmidspeedy = speed*sin(angle) + turnspeed*sin(leftmidangle)*midrad/rearrad;

leftmiddir = atan2(leftmidspeedy , leftmidspeedx);
```

```
leftmidswingx = swingspeed * cos(leftmiddir);
leftmidswingy = swingspeed * sin(leftmiddir);

leftmidaepx = leftmidcenterx + steplength/2*cos(leftmiddir);
leftmidpepx = leftmidcenterx - steplength/2*cos(leftmiddir);
leftmidaepy = leftmidcentery + steplength/2*sin(leftmiddir);
leftmidpepy = leftmidcentery - steplength/2*sin(leftmiddir);

/*--------------------*/

rightmidspeedx = speed*cos(angle) + turnspeed*cos(rightmidangle)*midrad/rearrad;
rightmidspeedy = speed*sin(angle) + turnspeed*sin(rightmidangle)*midrad/rearrad;

rightmiddir = atan2(rightmidspeedy , rightmidspeedx);

rightmidswingx = swingspeed*cos(rightmiddir);
rightmidswingy = swingspeed*sin(rightmiddir);

rightmidaepx = rightmidcenterx + steplength/2*cos(rightmiddir);
rightmidpepx = rightmidcenterx - steplength/2*cos(rightmiddir);
rightmidaepy = rightmidcentery + steplength/2*sin(rightmiddir);
rightmidpepy = rightmidcentery - steplength/2*sin(rightmiddir);

/*--------------------*/

leftrearspeedx = speed*cos(angle) + turnspeed*cos(leftrearangle);
leftrearspeedy = speed*sin(angle) + turnspeed*sin(leftrearangle);

leftreardir = atan2(leftrearspeedy , leftrearspeedx);

leftrearswingx = swingspeed * cos(leftreardir);
leftrearswingy = swingspeed * sin(leftreardir);

leftrearaepx = leftrearcenterx + steplength/2*cos(leftreardir);
leftrearpepx = leftrearcenterx - steplength/2*cos(leftreardir);
leftrearaepy = leftrearcentery + steplength/2*sin(leftreardir);
leftrearpepy = leftrearcentery - steplength/2*sin(leftreardir);

/*--------------------*/

rightrearspeedx = speed*cos(angle) + turnspeed*cos(rightrearangle);
rightrearspeedy = speed*sin(angle) + turnspeed*sin(rightrearangle);

rightreardir = atan2(rightrearspeedy , rightrearspeedx);

rightrearswingx = swingspeed * cos(rightreardir);
rightrearswingy = swingspeed * sin(rightreardir);

rightrearaepx = rightrearcenterx + steplength/2*cos(rightreardir);
rightrearpepx = rightrearcenterx - steplength/2*cos(rightreardir);
rightrearaepy = rightrearcentery + steplength/2*sin(rightreardir);
rightrearpepy = rightrearcentery - steplength/2*sin(rightreardir);


/* set targets at PEP--this is just for initializing*/
```

```
lefttfronttarget = lefttfrontpepx;
leftmidtarget = leftmidpepx;
leftreartarget = leftrearpepx;
rightfronttarget = rightfrontpepx;
rightmidtarget = rightmidpepx;
rightreartarget = rightrearpepx;


/* arbitrary start--standing */

leftfrontcontact = 1;
leftmidcontact = 1;
leftrearcontact = 1;
rightfrontcontact = 1;
rightmidcontact = 1;
rightrearcontact = 1;


/*start for ramp up, near wave */
if (sign(speed) >= 0)
{
        leftfrontpos = leftfrontcenterx + 3*cos(leftfrontdir);
        leftmidpos = leftmidcenterx + 1*cos(leftmiddir);
        leftrearpos = leftrearcenterx + 0*cos(leftreardir);
        rightfrontpos = rightfrontcenterx - 1*cos(rightfrontdir);
        rightmidpos = rightmidcenterx - 1*cos(rightmiddir);
        rightrearpos = rightrearcenterx + 1*cos(rightreardir);

        leftfronty = leftfrontcentery + 3*sin(leftfrontdir);
        leftmidy = leftmidcentery + 1*sin(leftmiddir);
        leftreary = leftrearcentery + 0*sin(leftreardir);
        rightfronty = rightfrontcentery - 1*sin(rightfrontdir);
        rightmidy = rightmidcentery - 1*sin(rightmiddir);
        rightreary = rightrearcentery + 1*sin(rightreardir);
}

else
{
        leftfrontpos = leftfrontcenterx - 1*cos(leftfrontdir);
        leftmidpos = leftmidcenterx - 1*cos(leftmiddir);
        leftrearpos = leftrearcenterx + 3*cos(leftreardir);
        rightfrontpos = rightfrontcenterx - 3*cos(rightfrontdir);
        rightmidpos = rightmidcenterx - 1*cos(rightmiddir);
        rightrearpos = rightrearcenterx - 0*cos(rightreardir);

        leftfronty = leftfrontcentery - 1*sin(leftfrontdir);
        leftmidy = leftmidcentery - 1*sin(leftmiddir);
        leftreary = leftrearcentery + 3*sin(leftreardir);
        rightfronty = rightfrontcentery - 3*sin(rightfrontdir);
        rightmidy = rightmidcentery - 1*sin(rightmiddir);
        rightreary = rightrearcentery - 0*sin(rightreardir);
}


/*----------------------------------*/
```

```
/* set all mechanisms to 0, they will sort out in first loop iteration */

leftfront1 = 0;
leftfront2 = 0;
leftfront5 = 0;
leftfrontcross2 = 0;
leftfrontcross5 = 0;
leftmid1 = 0;
leftmid2 = 0;
leftmid5 = 0;
leftmidcross2 = 0;
leftmidcross5 = 0;
leftrear1 = 0;
leftrear2 = 0;
leftrear5 = 0;
leftrearcross1 = 0;
leftrearcross2 = 0;
leftrearcross5 = 0;

rightfront1 = 0;
rightfront2 = 0;
rightfront5 = 0;
rightfrontcross2 = 0;
rightfrontcross5 = 0;
rightmid1 = 0;
rightmid2 = 0;
rightmid5 = 0;
rightmidcross2 = 0;
rightmidcross5 = 0;
rightrear1 = 0;
rightrear2 = 0;
rightrear5 = 0;
rightrearcross1 = 0;
rightrearcross2 = 0;
rightrearcross5 = 0;

/* Initializing for vertical position control */

leftfrontend = leftfrontpep;
leftfrontzmax = -5.0;
leftfrontground = -8.0;
leftfrontz = leftfrontground;

leftmidend = leftmidpep;
leftmidzmax = -6.5;
leftmidground = -8.0;
leftmidz = leftmidground;

leftrearend = leftrearpep;
leftrearzmax = -6.5;
leftrearground = -8.0;
leftrearz = leftrearground;


rightfrontend = rightfrontpep;
rightfrontzmax = -5.0;
```

```
rightfrontground = -8.0;
rightfrontz = rightfrontground;

rightmidend = rightmidpep;
rightmidzmax = -6.5;
rightmidground = -8.0;
rightmidz = rightmidground;

rightrearend = rightrearpep;
rightrearzmax = -6.5;
rightrearground = -8.0;
rightrearz = rightrearground;

/*-----------------------------------------*/

printf("    rr    rm    rf    lr    lm    lf \n");

if (leftfrontcontact == 1)
        leftfrontstate = ' ';
    else
        leftfrontstate = '|';

    if (leftmidcontact == 1)
        leftmidstate = ' ';
    else
        leftmidstate = '|';

    if (leftrearcontact == 1)
        leftrearstate = ' ';
    else
        leftrearstate = '|';

    if (rightfrontcontact == 1)
        rightfrontstate = ' ';
    else
        rightfrontstate = '|';

    if (rightmidcontact == 1)
        rightmidstate = ' ';
    else
        rightmidstate = '|';

    if (rightrearcontact == 1)
        rightrearstate = ' ';
    else
        rightrearstate = '|';

printf("  %c    %c    %c    %c    %c    %c  %f  %f  %f\n", rightrearstate,
rightmidstate, rightfrontstate, leftrearstate,leftmidstate, leftfrontstate, leftfrontpos, leftfrontaepx,
stancespeed);


/* begin primary walking loop */
/*-------------------------------------------------------------------------*/

/* Ramp to desired speed */
```

```
		for (j=0; j<(duration); j=j+timestep)
		{
		if (stancespeed*sign(speed)<=(gaitspeed - gaitspeed*0.025*sign(speed)))
			stancespeed = stancespeed + gaitramp * sign(speed);
		else
			stancespeed = gaitspeed;


		if (turnspeed*sign(turn)<=(turnrate - turnrate*0.025*sign(turn)))
			turnspeed = turnspeed + gaitramp * sign(turn);
		else
			turnspeed = turnrate;

		/* each leg will get its own if statement */
		/* values are reassigned based on leg state here */

		/* ----------left front section---------- */

			leftfrontspeedx = stancespeed*cos(angle) +
turnspeed*cos(leftfrontangle)*frontrad/rearrad;
			leftfrontspeedy = stancespeed*sin(angle) +
turnspeed*sin(leftfrontangle)*frontrad/rearrad;

			leftfrontdir = atan2(leftfrontspeedy , leftfrontspeedx);

			leftfrontswingx = swingspeed*cos(leftfrontdir);
			leftfrontswingy = swingspeed*sin(leftfrontdir);

			leftfrontaepx = leftfrontcenterx + steplength/2*cos(leftfrontdir);
			leftfrontpepx = leftfrontcenterx - steplength/2*cos(leftfrontdir);
			leftfrontaepy = leftfrontcentery + steplength/2*sin(leftfrontdir);
			leftfrontpepy = leftfrontcentery - steplength/2*sin(leftfrontdir);

			if (leftfrontcontact == 1)
			{
				leftfrontz = leftfrontground;
				leftfrontpos = leftfrontpos - leftfrontspeedx*timestep;
				leftfronty = leftfronty - leftfrontspeedy*timestep;
				leftfront1 = 0;

				if((leftfrontpos * sign (leftfrontspeedx)) > ((leftfrontaepx -
(leftfrontaepx - leftfrontpepx)/10) * sign (leftfrontspeedx)))
				{

					leftfront2 = mechanism2;
					leftfrontcross2 = crossmechanism2;
				}
				else
				{
					leftfront2 = 0;
					leftfrontcross2 = 0;
				}
				if ((leftfrontpos * sign (leftfrontspeedx)) < ((leftfronttarget) * sign
(leftfrontspeedx)))
```

```
                                        {
                                                leftfrontcontact = 0;
                                                leftfrontend = leftfrontpos;
                                        }
                                        else
                                        {
                                                leftfront5 = 0.5*mechanism5*((leftfrontaepx -
leftfrontpos)/(leftfrontaepx - leftfrontpepx));
                                                leftfrontcross5 = 0.5*crossmechanism5*((leftfrontaepx -
leftfrontpos)/(leftfrontaepx - leftfrontpepx));
                                        }

                                }

                                else
                                {
                                        leftfrontpos = leftfrontpos + leftfrontswingx*timestep;
                                        leftfronty = leftfronty + leftfrontswingy*timestep;
                                        leftfront1 = mechanism1;
                                        leftfront2 = 0;
                                        leftfrontcross2 = 0;
                                        leftfront5 = 0;
                                        leftfrontcross5 = 0;
                                        leftfrontz = (4*(leftfrontzmax - leftfrontground)/((leftfrontaepx -
leftfrontend)*(leftfrontaepx - leftfrontend))) * (leftfrontpos - leftfrontend)*(leftfrontaepx - leftfrontpos)
+ leftfrontground;
                                        if ((leftfrontpos * sign (leftfrontspeedx)) > (leftfrontaepx * sign
(leftfrontspeedx)))
                                        {
                                                leftfrontcontact = 1;
                                                leftfront2 = 0;
                                                leftfrontcross2 = 0;
                                                leftfrontcount = 1;
                                                leftfrontz = leftfrontground;
                                        }

                                }

                                /* ----------end left front section---------- */




                                /* ----------right front section---------- */

                                rightfrontspeedx = stancespeed*cos(angle) +
turnspeed*cos(rightfrontangle)*frontrad/rearrad;
                                rightfrontspeedy = stancespeed*sin(angle) +
turnspeed*sin(rightfrontangle)*frontrad/rearrad;

                                rightfrontdir = atan2(rightfrontspeedy , rightfrontspeedx);

                                rightfrontswingx = swingspeed*cos(rightfrontdir);
                                rightfrontswingy = swingspeed*sin(rightfrontdir);
```

```
rightfrontaepx = rightfrontcenterx + steplength/2*cos(rightfrontdir);
rightfrontpepx = rightfrontcenterx - steplength/2*cos(rightfrontdir);
rightfrontaepy = rightfrontcentery + steplength/2*sin(rightfrontdir);
rightfrontpepy = rightfrontcentery - steplength/2*sin(rightfrontdir);


if (rightfrontcontact == 1)
{
	rightfrontz = rightfrontground;
	rightfrontpos = rightfrontpos - rightfrontspeedx*timestep;
	rightfronty = rightfronty - rightfrontspeedy*timestep;
	rightfront1 = 0;

	if((rightfrontpos * sign (rightfrontspeedx)) > ((rightfrontaepx -
(rightfrontaepx - rightfrontpepx)/10) * sign (rightfrontspeedx)))
	{

		rightfront2 = mechanism2;
		rightfrontcross2 = crossmechanism2*contbias;
	}
	else
	{
		rightfront2 = 0;
		rightfrontcross2 = 0;
	}
	if ((rightfrontpos * sign (rightfrontspeedx)) < ((rightfronttarget) *
sign (rightfrontspeedx)))
	{
		rightfrontcontact = 0;
		rightfrontend = rightfrontpos;
	}
	else
	{
		rightfront5 = 0.5*mechanism5*((rightfrontaepx -
rightfrontpos)/(rightfrontaepx - rightfrontpepx));
		rightfrontcross5 = 0.5*crossmechanism5*((rightfrontaepx
- rightfrontpos)/(rightfrontaepx - rightfrontpepx))*contbias;
	}

}
else
{
	rightfrontpos = rightfrontpos + rightfrontswingx*timestep;
	rightfronty = rightfronty + rightfrontswingy*timestep;
	rightfront1 = mechanism1;
	rightfront2 = 0;
	rightfrontcross2 = 0;
	rightfront5 = 0;
	rightfrontcross5 = 0;
	rightfrontz = (4*(rightfrontzmax -
rightfrontground)/((rightfrontaepx - rightfrontend)*(rightfrontaepx - rightfrontend))) * (rightfrontpos -
rightfrontend)*(rightfrontaepx - rightfrontpos) + rightfrontground;
	if ((rightfrontpos * sign (rightfrontspeedx)) > (rightfrontaepx *
sign (rightfrontspeedx)))
	{
```

```
                                        righttrontcontact = 1;
                                        rightfront2 = 0;
                                        rightfrontcross2 = 0;
                                        rightfrontcount = 1;
                                        rightfrontz = rightfrontground;
                                }

                        }

                        /* ----------end right front section---------- */


                        /* ----------left mid section---------- */

                        leftmidspeedx = stancespeed*cos(angle) +
turnspeed*cos(leftmidangle)*midrad/rearrad;
                        leftmidspeedy = stancespeed*sin(angle) +
turnspeed*sin(leftmidangle)*midrad/rearrad;

                        leftmiddir = atan2(leftmidspeedy , leftmidspeedx);

                        leftmidswingx = swingspeed * cos(leftmiddir);
                        leftmidswingy = swingspeed * sin(leftmiddir);

                        leftmidaepx = leftmidcenterx + steplength/2*cos(leftmiddir);
                        leftmidpepx = leftmidcenterx - steplength/2*cos(leftmiddir);
                        leftmidaepy = leftmidcentery + steplength/2*sin(leftmiddir);
                        leftmidpepy = leftmidcentery - steplength/2*sin(leftmiddir);


                        if (leftmidcontact == 1)
                        {
                                leftmidz = leftmidground;
                                leftmidpos = leftmidpos - leftmidspeedx*timestep;
                                leftmidy = leftmidy - leftmidspeedy*timestep;
                                leftmid1 = 0;

                                if ((leftmidpos * sign (leftmidspeedx)) > ((leftmidaepx -
(leftmidaepx - leftmidpepx)/10) * sign (leftmidspeedx)))
                                {

                                        leftmid2 = mechanism2;
                                        leftmidcross2 = crossmechanism2;
                                }
                                else
                                {
                                        leftmid2 = 0;
                                        leftmidcross2 = 0;
                                }
                                if ((leftmidpos * sign (leftmidspeedx)) < ((leftmidtarget) * sign
(leftmidspeedx)))
                                {
                                        leftmidcontact = 0;
                                        leftmidend = leftmidpos;
                                }
                                else
```

```
                                        {
                                                leftmid5 = 0.5*mechanism5*((leftmidaepx -
leftmidpos)/(leftmidaepx - leftmidpepx));
                                                leftmidcross5 = 0.5*crossmechanism5*((leftmidaepx -
leftmidpos)/(leftmidaepx - leftmidpepx));
                                        }

                                }
                                else
                                {
                                        leftmidpos = leftmidpos + leftmidswingx*timestep;
                                        leftmidy = leftmidy + leftmidswingy*timestep;
                                        leftmid1 = mechanism1;
                                        leftmid2 = 0;
                                        leftmidcross2 = 0;
                                        leftmid5 = 0;
                                        leftmidcross5 = 0;
                                        leftmidz = (4*(leftmidzmax - leftmidground)/((leftmidaepx -
leftmidend)*(leftmidaepx - leftmidend))) * (leftmidpos - leftmidend)*(leftmidaepx - leftmidpos) +
leftmidground;
                                        if ((leftmidpos * sign (leftmidspeedx)) > (leftmidaepx * sign
(leftmidspeedx)))
                                        {
                                                leftmidcontact = 1;
                                                leftmid2 = 0;
                                                leftmidcross2 = 0;
                                                leftmidcount = 1;
                                                leftmidz = leftmidground;
                                        }

                                }

                                /* ----------end left mid section---------- */

                                /* ----------right mid section---------- */

                                rightmidspeedx = stancespeed*cos(angle) +
turnspeed*cos(rightmidangle)*midrad/rearrad;
                                rightmidspeedy = stancespeed*sin(angle) +
turnspeed*sin(rightmidangle)*midrad/rearrad;

                                rightmiddir = atan2(rightmidspeedy , rightmidspeedx);

                                rightmidswingx = swingspeed*cos(rightmiddir);
                                rightmidswingy = swingspeed*sin(rightmiddir);

                                rightmidaepx = rightmidcenterx + steplength/2*cos(rightmiddir);
                                rightmidpepx = rightmidcenterx - steplength/2*cos(rightmiddir);
                                rightmidaepy = rightmidcentery + steplength/2*sin(rightmiddir);
                                rightmidpepy = rightmidcentery - steplength/2*sin(rightmiddir);


                                if (rightmidcontact == 1)
                                {
                                        rightmidz = rightmidground;
                                        rightmidpos = rightmidpos - rightmidspeedx*timestep;
```

```
                                rightmidy = rightmidy - rightmidspeedy*timestep;
                                rightmid1 = 0;

                                if ((rightmidpos * sign (rightmidspeedx)) > ((rightmidaepx -
(rightmidaepx - rightmidpepx)/10) * sign (rightmidspeedx)))
                                /* keep mechanisms clear */
                                {
                                        rightmid2 = mechanism2;
                                        rightmidcross2 = crossmechanism2*contbias;
                                }
                                else
                                {
                                        rightmid2 = 0;
                                        rightmidcross2 = 0;
                                }
                                if ((rightmidpos * sign (rightmidspeedx)) < ((rightmidtarget) * sign
(rightmidspeedx)))
                                {
                                        rightmidcontact = 0;
                                        rightmidend = rightmidpos;
                                }
                                else
                                {
                                        rightmid5 = 0.5*mechanism5*((rightmidaepx -
rightmidpos)/(rightmidaepx - rightmidpepx));
                                        rightmidcross5 = 0.5*crossmechanism5*((rightmidaepx -
rightmidpos)/(rightmidaepx - rightmidpepx))*contbias;
                                }

                        }
                        else
                        {
                                rightmidpos = rightmidpos + rightmidswingx*timestep;
                                rightmidy = rightmidy + rightmidswingy*timestep;
                                rightmid1 = mechanism1;
                                rightmid2 = 0;
                                rightmidcross2 = 0;
                                rightmid5 = 0;
                                rightmidcross5 = 0;
                                rightmidz = (4*(rightmidzmax - rightmidground)/((rightmidaepx -
rightmidend)*(rightmidaepx - rightmidend))) * (rightmidpos - rightmidend)*(rightmidaepx -
rightmidpos) + rightmidground;
                                if ((rightmidpos * sign (rightmidspeedx)) > (rightmidaepx * sign
(rightmidspeedx)))
                                {
                                        rightmidcontact = 1;
                                        rightmid2 = 0;
                                        rightmidcross2 = 0;
                                        rightmidcount = 1;
                                        rightmidz = rightmidground;
                                }

                        }

                        /* ----------end right mid section---------- */
```

```
/* ----------left rear section---------- */

leftrearspeedx = stancespeed*cos(angle) + turnspeed*cos(leftrearangle);
leftrearspeedy = stancespeed*sin(angle) + turnspeed*sin(leftrearangle);

leftreardir = atan2(leftrearspeedy , leftrearspeedx);

leftrearswingx = swingspeed * cos(leftreardir);
leftrearswingy = swingspeed * sin(leftreardir);

leftrearaepx = leftrearcenterx + steplength/2*cos(leftreardir);
leftrearpepx = leftrearcenterx - steplength/2*cos(leftreardir);
leftrearaepy = leftrearcentery + steplength/2*sin(leftreardir);
leftrearpepy = leftrearcentery - steplength/2*sin(leftreardir);


if (leftrearcontact == 1)
{
        leftrearz = leftrearground;
        leftrearpos = leftrearpos - leftrearspeedx*timestep;
        leftreary = leftreary - leftrearspeedy*timestep;
        if (((leftrearpos * sign (leftrearspeedx))) > ((leftrearaepx -
(leftrearaepx - leftrearpepx)/10) * sign (leftrearspeedx)))
        /* keep mechanisms clear */
        {
                leftrear2 = mechanism2;
                leftrearcross2 = crossmechanism2;
        }
        else
        {
                leftrear2 = 0;
                leftrearcross2 = 0;
        }
        leftrearcross2 = 0;
        if ((leftrearpos * sign (leftrearspeedx)) < ((leftreartarget) * sign
(leftrearspeedx)))
        {
                leftrearcontact = 0;
                leftrearend = leftrearpos;
        }
        else
        {
                leftrear5 = 0.5*mechanism5*((leftrearaepx -
leftrearpos)/(leftrearaepx - leftrearpepx));
                leftrearcross5 = 0.5*crossmechanism5*((leftrearaepx -
leftrearpos)/(leftrearaepx - leftrearpepx));
        }

}
else
{
        leftrearpos = leftrearpos + leftrearswingx*timestep;
        leftreary = leftreary + leftrearswingy*timestep;
        leftrear1 = mechanism1;
        leftrearcross1 = crossmechanism1;
        leftrear2 = 0;
```

```
                                leftrearcross2 = 0;
                                leftrear5 = 0;
                                leftrearcross5 = 0;
                                leftrearz = (4*(leftrearzmax - leftrearground)/((leftrearaepx -
leftrearend)*(leftrearaepx - leftrearend))) * (leftrearpos - leftrearend)*(leftrearaepx - leftrearpos) +
leftrearground;
                                if ((leftrearpos * sign (leftrearspeedx)) > (leftrearaepx * sign
(leftrearspeedx)))
                                {
                                        leftrearcontact = 1;
                                        leftrear2 = 0;
                                        leftrearcross2 = 0;
                                        leftrearcount = 1;
                                        leftrearz = leftrearground;
                                }

                        }

                        /* ----------end left rear section---------- */

                        /* ----------right rear section---------- */

                        rightrearspeedx = stancespeed*cos(angle) + turnspeed*cos(rightrearangle);
                        rightrearspeedy = stancespeed*sin(angle) + turnspeed*sin(rightrearangle);

                        rightreardir = atan2(rightrearspeedy , rightrearspeedx);

                        rightrearswingx = swingspeed * cos(rightreardir);
                        rightrearswingy = swingspeed * sin(rightreardir);

                        rightrearaepx = rightrearcenterx + steplength/2*cos(rightreardir);
                        rightrearpepx = rightrearcenterx - steplength/2*cos(rightreardir);
                        rightrearaepy = rightrearcentery + steplength/2*sin(rightreardir);
                        rightrearpepy = rightrearcentery - steplength/2*sin(rightreardir);

                        if (rightrearcontact == 1)
                        {
                                rightrearz = rightrearground;
                                rightrearpos = rightrearpos - rightrearspeedx*timestep;
                                rightreary = rightreary - rightrearspeedy*timestep;
                                rightrear1 = 0;

                                if (((rightrearpos * sign (rightrearspeedx))) > ((rightrearaepx -
(rightrearaepx - rightrearpepx)/10) * sign (rightrearspeedx)))
                                /* keep mechanisms clear */
                                {
                                        rightrear2 = mechanism2;
                                        rightrearcross2 = crossmechanism2*contbias;
                                }
                                else
                                {
                                        rightrear2 = 0;
                                        rightrearcross2 = 0;
                                }
                                if ((rightrearpos * sign (rightrearspeedx)) < ((rightreartarget) * sign
(rightrearspeedx)))
```

```
                                        {
                                                rightrearcontact = 0;
                                                rightrearend = rightrearpos;
                                        }
                                        else
                                        {
                                                rightrear5 = 0.5*mechanism5*((rightrearaepx -
rightrearpos)/(rightrearaepx - rightrearpepx));
                                                rightrearcross5 = 0.5*crossmechanism5*((rightrearaepx -
rightrearpos)/(rightrearaepx - rightrearpepx))*contbias;
                                        }

                                }
                                else
                                {
                                        rightrearpos = rightrearpos + rightrearswingx*timestep;
                                        rightreary = rightreary + rightrearswingy*timestep;
                                        rightrear1 = mechanism1;
                                        rightrearcross1 = crossmechanism1*contbias;
                                        rightrear2 = 0;
                                        rightrearcross2 = 0;
                                        rightrear5 = 0;
                                        rightrearcross5 = 0;
                                        rightrearz = (4*(rightrearzmax - rightrearground)/((rightrearaepx -
rightrearend)*(rightrearaepx - rightrearend))) * (rightrearpos - rightrearend)*(rightrearaepx -
rightrearpos) + rightrearground;
                                        if ((rightrearpos * sign (rightrearspeedx)) > (rightrearaepx * sign
(rightrearspeedx)))
                                        {
                                                rightrearcontact = 1;
                                                rightrear2 = 0;
                                                rightrearcross2 = 0;
                                                rightrearcount = 1;
                                                rightrearz = rightrearground;
                                        }

                                }

                                /* ----------end right rear section---------- */

                                /* Now, reassign PEP target values with above network values */

                                leftfronttarget = leftfrontpepx + (rightfrontcross2*crossmechanismscale2 +
rightfrontcross5*crossmechanismscale5 + leftmid1*mechanismscale1 + leftmid2*mechanismscale2) *
cos (leftfrontdir);

                                rightfronttarget = rightfrontpepx + (leftfrontcross2*crossmechanismscale2 +
leftfrontcross5*crossmechanismscale5 + rightmid1*mechanismscale1 + rightmid2*mechanismscale2)
* cos (rightfrontdir);

                                leftmidtarget = leftmidpepx + (rightmidcross2*crossmechanismscale2 +
rightmidcross5*crossmechanismscale5 + leftfront5*mechanismscale5 + leftrear1*mechanismscale1 +
leftrear2*mechanismscale2) * cos (leftmiddir);
```

```
                    rightmidtarget = rightmidpepx + (leftmidcross2*crossmechanismscale2 +
leftmidcross5*crossmechanismscale5 + rightfront5*mechanismscale5 + rightrear1*mechanismscale1 +
rightrear2*mechanismscale2) * cos (rightmiddir);

                    leftreartarget = leftrearpepx + (rightrearcross1*crossmechanismscale1 +
rightrearcross2*crossmechanismscale2 + rightrearcross5*crossmechanismscale5 +
leftmid5*mechanismscale5) * cos (leftreardir);

                    rightreartarget = rightrearpepx + (leftrearcross1*crossmechanismscale1 +
leftrearcross2*crossmechanismscale2 + leftrearcross5*crossmechanismscale5 +
rightmid5*mechanismscale5) * cos (rightreardir);


                    /* Just because I worry */
                    /*   < and > may need to be changed depending on AEP and PEP

               if (sign(speed) * leftfrontpos < sign(speed) * (leftfrontpep-(leftfrontaep-
leftfrontpep)/2))
               {
                    leftfrontpos = (leftfrontpep-(leftfrontaep-leftfrontpep)/2);
                    printf("left front PEP overshoot \n");
               }


               if (sign(speed) * rightfrontpos < sign(speed) * (rightfrontpep-(rightfrontaep-
rightfrontpep)/2))
               {
                    rightfrontpos = (rightfrontpep-(rightfrontaep-rightfrontpep)/2);
                    printf("right front PEP overshoot \n");
               }


               if (sign(speed) * leftmidpos < sign(speed) * (leftmidpep-(leftmidaep-
leftmidpep)/2))
               {
                    leftmidpos = (leftmidpep-(leftmidaep-leftmidpep)/2);
                    printf("left mid PEP overshoot \n");
               }


               if (sign(speed) * rightmidpos < sign(speed) * (rightmidpep-(rightmidaep-
rightmidpep)/2))
               {
                    rightmidpos = (rightmidpep-(rightmidaep-rightmidpep)/2);
                    printf("mid front PEP overshoot \n");
               }


               if (sign(speed) * leftrearpos < sign(speed) * (leftrearpep-(leftrearaep-leftrearpep)/2))
               {
                    leftrearpos = (leftrearpep-(leftrearaep-leftrearpep)/2);
                    printf("left rear PEP overshoot \n");
               }
```

```
                if (sign(speed) * rightrearpos < sign(speed) * (rightrearpep-(rightrearaep-
rightrearpep)/2))
                {
                        rightrearpos = (rightrearpep-(rightrearaep-rightrearpep)/2);
                        printf("right rear PEP overshoot \n");
                }

                        /* Now, just put together some output */

                        if (leftfrontcontact == 1)
                                leftfrontstate = ' ';
                        else
                                leftfrontstate = '|';

                        if (leftmidcontact == 1)
                                leftmidstate = ' ';
                        else
                                leftmidstate = '|';

                        if (leftrearcontact == 1)
                                leftrearstate = ' ';
                        else
                                leftrearstate = '|';

                        if (rightfrontcontact == 1)
                                rightfrontstate = ' ';
                        else
                                rightfrontstate = '|';

                        if (rightmidcontact == 1)
                                rightmidstate = ' ';
                        else
                                rightmidstate = '|';

                        if (rightrearcontact == 1)
                                rightrearstate = ' ';
                        else
                                rightrearstate = '|';


                                printf("    %c     %c     %c     %c     %c     %c   %f  %f  %f\n",
rightrearstate, rightmidstate, rightfrontstate, leftrearstate,leftmidstate, leftfrontstate, leftrearpos,
leftrearaepx, leftreartarget);

                }
                /*end main network loop */

                return 0;
        }
```

Appendix B: Actuators Onboard Robot V

		Length (inches)	Number (per joint)	Diameter (in)
Front Leg	Tibia Flexor	5.11	1	0.38
	Tibia Extensor	5.11	1	0.38
	Femur Flexor	3.86	1	0.38
	Femur Extensor	3.75	1	0.78
	Alpha Flexor	4.625	1	0.38
	Alpha Flexor	4.625	1	0.78
	Alpha Extensor	5.31	2	0.38
	Beta Flexor	4.875	2	0.38
	Beta Extensor	5.625	2	0.38
	Gamma Flexor	4.36	2	0.38
	Gamma Extensor	4.125	1	0.38
Middle Leg	Tibia Flexor	6.36	1	0.38
	Tibia Extensor	6.26	1	0.38
	Femur Flexor	5.21	2	0.38
	Femur Extensor	4.81	2	0.78
	Alpha Flexor	4	1	0.38
	Alpha Flexor	4	1	0.78
	Alpha Extensor	5.95	2	0.38
	Beta Flexor	5	2	0.38
	Beta Extensor	5.75	2	0.78
Rear Leg	Tibia Flexor	6.8	1	0.38
	Tibia Extensor	6.8	1	0.38
	Femur Flexor	5.65	2	0.38
	Femur Extensor	6.35	2	0.78
	Beta Flexor	5	2	0.38
	Beta Extensor	5.75	2	0.78

BIBLIOGRAPHY

Alexander, R.M., 1988. Elastic Mechanisms in Animal Movement, Cambridge University Press, Cambridge.

Bachmann, R.J. (2000) A Cockroach-Like Hexapod Robot for Running and Climbing. M.S. Thesis, CWRU.

Bachmann, R. J., Nelson, G. M., Quinn, R. D., Watson, J., Ritzmann, R. E. "Design of a Cockroach-like Robot, Proceedings of the 11th VPI&SU Symposium on Structural Dynamics and Control, May 12-14, 1997.

Bares, J.E., and Whittaker, W.L. (1993) *Int. Journal of Robotic Research*, **12**(6).

Beer, R.D., Ritzmann, R.E., and McKenna, T. Eds. *Biological Neural Networks in Invertebrate Neuroethology and Robotics.* Academic Press, San Diego, 1993.

Biewener, A. A., Animal Locomotion (Oxford Animal Biology Series) 2003 Oxford University Press.

Bidaud, P., and Amar, F. B. "5th International Conference on Climbing and Walking Robots" Professional Engineering and Publishing Limited, London, UK, 2002.

Berns, K. and Dillmann, R. "4th International Conference on Climbing and Walking Robots" Professional Engineering and Publishing Limited, London, UK, 2001.

Binnard, M.B. (1995) Design of a Small Pneumatic Walking Robot. M.S. Thesis, M.I.T.

Brooks, R.A. (1989) A robot that walks: Emergent behaviors from a carefully evolved network. *Neural comp.* 1:253-262

Caldwell, D.G, Medrano-Cerda, G.A., and Bowler C.J. "Investigation of Bipedal Robot Locomotion Using Pneumatic Muscle Actuators" . IEEE International Conference on Robotics and Automation (ICRA'97), Albuquerque, NM.

Chou, C.P., B. Hannaford "Measurement and Modeling of McKibben Pneumatic Artificial Muscles," *"IEEE Transactions on Robotics and Automatio*n," Vol. 12, No. 1, pp. 90-102, Feb 1996.

Colbrunn, R.W. (2000) Design and Control of a Robotic Leg with Braided Pneumatic Actuators M.S. Thesis, CWRU.

Cooke, D.S., Hewer, N.D., White, T.S, Galt, S., and Luk, B.L. Implementation of Modularity in Robug IV—Preliminary Results. International Conference on Climbing and Walking Robots 1999.

Costa, N., Artrit, P., Caldwell, D.G. "Soft Interfaces for a Humanoid Robot—Karate Robot" Proceedings from 5th International Conference on Climbing and Walking Robots" Professional Engineering and Publishing Limited, London, UK, 2002.

Cruse, H. (1990). What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences* **13**:15-21.

Delcomyn, F *Foundations of Neurobiology* W.H. Freedman and Company, New York, 1998.

Delcomyn, F. and Nelson, M.E. (1999). Architectures for a Biomimetic Hexapod Robot. *Robotics and Autonomous Systems*.

Durr, V., Schmitz, J., Cruse, H. "Behaviour-based Modelling of Hexapod Locomotion: Linking Biology and Technical Application" Arthropod Structure and Development vol. 33, 3 July 2004.

Espenschied, K. S., Quinn, R. D., Chiel, H. J., and Beer, R. D. (1993) Leg Coordination Mechanisms in Stick Insect Applied to Hexapod Robot Locomotion. *Adaptive Behavior.* **1**(4):455-468.

Espenschied, K. S., Quinn, R. D., Chiel, H. J., and Beer, R. D. (1995). Biologically-Inspired Hexapod Robot Project: Robot II. IEEE International Conference on Robotics and Automation (ICRA'95) Video Proceedings, Nagoya, Japan.

Espenschied, K.S., Quinn, R.D., Chiel, H.J., Beer, R.D. (1996). Biologically-Based Distributed Control and Local Reflexes Improve Rough Terrain Locomotion in a Hexapod Robot. Robotics and Autonomous Systems, Vol. 18, 59-64.

Frik, M., Guddat, M., Karatas, M., Losch, D.C. "A novel approach to autonomous control of walking machines" Virk, G.S., Randall, M., Howard, D., 2nd International Conference on Climbing and Walking Robots, Professional Engineering and Publishing Limited, London, UK, 1999.

Hirose, S., Nagakubo, A. and Toyama R. (1991). Machine that can walk and climb on floors, walls, ceilings. Proceedings of the 5th International Conference on Advanced Robotics. Pp753-758.

Jenkins, J.B. *Hollinshead's Functional Anatomy of the Limbs and Back* W.B. Saunders Company, Philadelphia, 1998.

Jindrich, D.L. and Full, R.J. 2002. Dynamic stabilization of rapid hexapedal locomotion. J. exp Bio. 205, 2803-2823.

Juvinall, R.C. and Marshek, K.M. *Fundamentals of Machine Component Design* John Wiley and Sons, New York, 1991.

Kerscher, T., Albiez, J., Berns, K. Joint Control of the Six-Legged Robot AirBug Driven by Fluidic Muscles. Third International Workshop on Robotik Motion and Control, Bukowy - 2002

Kingsley, D.A. (2001), A Study of the Viability of Braided Pneumatic Actuators For Use On Walking Robots. M.S. Thesis, CWRU.

Kingsley, D. A., Quinn, R. D., Ritzmann, R. E. "A Cockroach Inspired Robot With Artificial Muscles," International Symposium on Adaptive Motion of Animals and Machines (AMAM 2003), Kyoto, Japan.

Klute, G.K., B. Hannaford (1998). Fatigue Characteristics of McKibben Artificial Muscle Actuators. Proc. Of IROS 1998, Victoria B.C. Canada, pp. 1776-82.

Klute, G.K., B. Hannaford "Modeling Pneumatic McKibben Artificial Muscle Actuators: Approaches and Experimental Results," Submitted to the ASME Journal of Dynamic Systems, Measurements, and Control, November 1998, revised March 1999.

Kram, R., Wong, B., Full, R. J. "Three-Dimensional Kinematics and Limb Kinetic Energy of Cockroaches" The Journal of Experimental Biology 200 (1997).

Loeb, G.E., Brown, I.E., Cheng, E.J. A Hierarchical Foundation For Models of Sensorimotor Control. *Experimental Brain Research* 1999, 126: 1-18.

McGhee, R.B. and Orin, D.E. (1977). A mathematical approach to control of joint positions and torques in legged locomotion systems. Theory and practice of robots and manipulators, ed. by A. Morecki and K. Kedzior, Elsevier, pp. 225-232.

Mu, L. A.L. Ridgel, , B.E. Alexander, R.E. Ritzmann (2003) The role of brain neuropils during turning in the cockroach. *Soc. Neuroci. Abstr. CD ROM* **29**: Prog Numb. 606.4.

Muscato, G., and Longo, D. "6th International Conference on Climbing and Walking Robots" Professional Engineering and Publishing Limited, London, UK, 2003

Nelson, G. M. and Quinn, R. D. "Posture Control of a Cockroach-like Robot," IEEE Control Systems, Vol. 19, No. 2, April 1999.

Nelson, G. M. (2002), Learning About Control of Legged Locomotion Using a Hexapod Robot With Compliant Pneumatic Actuators. Ph.D. Dissertation, CWRU.

Nickel, V.L., J. Perry, and A.L. Garrett "Development of useful function in the severely paralyzed hand," "*Journal of Bone and Joint Surgery*," Vol. 45A, No. 5, pp. 933-952, 1963.

Powers, A.C. (1996) Research in the Design and Construction of Biologically-Inspired Robots. M.S. Thesis, University of California, Berkeley.

Pratt, J., Dilworth, P., Pratt, G. (1997). Virtual Model Control of a Bipedal Walking Robot. Proc. of the IEEE Int. Conf. on Robot. and Automat. (ICRA 97), Albuquerque, NM.

Prochazka, A., Gillard, D., and Bennett, D.J. "Implications of Positive Feedback in the Control of Movement" The American Physiological Society, 1997.

Quinn, R. D. and Espenschied, K. S. "Control of a Hexapod Robot Using a Biologically Inspired Neural Network," Biological Neural Networks in Invertebrate Neuroethology and Robotics, Academic Press, 1993.

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

Raibert, M.H., Hodgins, J.K., Legged Robots, "*Biological Neural Networks in Invertebrate Neuroethology andRobotics*" ed. By Beer, R.D., Ritzmann, R.E., and McKenna, T. 1993.

Ritzmann, R.E., Quinn, R.D. Watson, J.T., and Zill, S.N. "Insect Walking and Biorobotics: A Relationship with Mutual Benefits," "*BioScience,"* Vol 50, No. 1 pp. 23-33, 2000.

Ritzmann, R.E., Quinn, R.D., Fischer, M.S. "Convergent Evolution and Locomotion Through Complex Terrain by Insects, Vertebrates and Robots," Arthropod Structure & Development 33 (2004).

Schwarz, S.E., Oldham, W.E. *Electrical Engineering: An Introduction* Saunders College Publishing, New York, 1993.

Song, S.M., Waldron, K.J., *Machines That Walk* MIT Press, Cambridge, Mass., 1989.

Theobold, C., "Air Compressor for Robot V" Senior Project, Case Western Reserve University, 2003.

Virk, G.S., Randall, M., Howard, D., "2nd International Conference on Climbing and Walking Robots" Professional Engineering and Publishing Limited, London, UK, 1999.

Watson, J. T., A. K. Tryba, R. E. Ritzmann and S. N. Zill (1997). Coordination of Leg Joints During Complex Locomotion in the Cockroach. *Soc. Neurosci. Abstr* **23**: 767.

Watson, J.T. and Ritzmann, R.E. "Leg Kinematics and Muscle Activity During Treadmill Running in the Cockroach, *Bladerus Discoidalis*: I. Slow Running". J. Comp Physiol (1998)

Watson, J.T., Ritzmann, R.E., and R.D. Quinn, S.N. Zill (1999) Control of Climbing Behavior in the Cockroach, Blaberus discoidalis . J. Comp. Physiol. A (in preparation).

Watson, J.T., Ritzmann, R.E., Zill, S.N., Pollack, A.J. "Control of Obstacle Climbing in the Cockroach, *Blaberus Discoidalis*. I. Kinematics" J Comp Physiol A (2002)

Watson, J.T., Ritzmann, R.E., Pollack, A.J. "Control of Obstacle Climbing in the Cockroach, *Blaberus Discoidalis*. II. Motor Activities Associated With Joint Movement" J Comp Physiol A (2002).

Weidemann, H.J., Pfeiffer, F., and Eltze, J., (1994). The six-legged TUM Walking Robot. IEEE Int. Conf. On Intelligent Robots and Systems (IROS'94), Munich, Germany, pp. 1026-1033.

White, F.M. *Fluid Mechanics* McGraw Hill, New York, 1994.

Wilson, D.M. (1966) "Insect Walking" Annual Review of Entomology, vol. 11 (ed. R.F. Smith and T.E. Mittler), pp. 103-1222. California: Annual Reviews Inc.

Zajac, F.E. "Muscle and Tendon: Properties, Models, Scaling, and Application to Biomechanics and Motor Control," *Critical Reviews in Biomedical Engineering,*" Volume 17, Issue 4, pp. 395-372 1989

Zill, S. N. and E.-A. Seyfarth (1996). Exoskeletal sensors for walking. *Sci. Am.* **275**(1): 86-90.