THE SINGLE-TRACK THREE LEGGED MOBILE ROBOT

by

John Robert Goulding

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2013

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by John Robert Goulding, titled The Single-Track Three Legged Mobile Robot and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

____ Date: December 3, 2013
Hal S. Tharp

____ Date: December 3, 2013
Jerzy W. Rozenblit

____ Date: December 3, 2013
Jonathan Sprinkle

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

____ Date: December 3, 2013
Dissertation Director: Hal S. Tharp

STATEMENT BY AUTHOR

SIGNED: John Robert Goulding

ACKNOWLEDGEMENTS

I would like to thank a number of individuals without whom my education and this dissertation could not have been completed. First, I thank my chair, Dr. Hal Tharp, for taking me on as his student and for all of the invaluable feedback he provided. I thank my committee and past instructors for their comments, reviews and suggestions that contributed to this effort. I especially thank Dr. Jerzy Rozenblit for his support and encouragement over the years, Mr. Jim Jindrick for teaching me entrepreneurship, and Dr. Herman Migliore who taught me how to think outside the box. I thank Mr. Tim Barlow and Mr. Tom Lees for helping me patent my invention, and I thank Mr. Sam Brown for helping me with the outstanding ST3LMR concept drawings (Chapter 9).

On a more personal note, I would also like to thank my family and friends for their unwavering support and constant encouragement during this endeavor. I especially thank my wonderful wife, Erika, for believing in me throughout this very long journey. I thank DJ and Ursula Fillman for welcoming me into their family. Moreover, I thank DJ for being the father I never had – no other engineer could have been more supportive. I thank my son, Cyrus Zamani, for first allowing me and then helping me to move to Tucson. Lastly, I thank Dr. and Mrs. Victor S. Carson for setting me on my academic journey and for providing me with a stipend to take time off from work to pursue my graduate studies.

DEDICATION

This dissertation is dedicated to the loving memory of my mother, Anna Ballo, who instilled in me a passion for learning and then sacrificed so much for my happiness.

I pray that this work will contribute in some small way to the development of service robots that help persons with disabilities, taking them places they otherwise could not go.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

Unstable legged robots fall over without active stabilization, typically by repositioning the feet to maintain/regain stability of balance. This dissertation concerns the development of a Single-Track Three Legged Mobile Robot (ST3LMR) and control system. A proof-of-concept was demonstrated through digital simulation and experimentation with physical prototypes.

The ST3LMR comprises a body and three articulated legs arranged in a narrow profile, one behind the other, to walk and maneuver along narrow trails and paths. The ST3LMR walks by placing successive footfalls in a generally single-track or in-line fashion. It achieves the form and function of a motorcycle but with the added benefit of legs and robotic control. That is, the feet are stationary with respect to footholds during the support period, thus eliminating the drawback of wheels, which require continuous support (especially when used in rugged terrain). By always having at least two feet on the ground, the ST3LMR is inherently stable in the pitch axis (in the forward/backward direction of motion), which allows for decoupling stability of balance control to only the roll axis (in the left/right direction).

Suggested by recent developments in high-performance computing, walking robot locomotion and stabilization is considered from a new perspective, that of the Monte Carlo (MC) method. A high-speed MC simulation is used in a model-predictive control system to determine footholds that provide stability of balance. Stability of balance, maneuverability, and control is demonstrated through experimental results from physical

prototypes and a simple digital simulation of an impulse response, avoidance maneuver, and leaning-into-the-turn maneuver.

CHAPTER 1

INTRODUCTION

It is advantageous to develop a vehicle that uses legs, because a vehicle with legs can go where wheeled or tracked vehicles cannot go (Holste and Ciccimaro 2009). Legged vehicles have improved mobility over rugged terrain with unstable footholds, such as mountain slopes and over piles of rubble. Wheeled vehicles require a continuous path of support to perform counter steering, wherein the line of support is (quickly) repositioned to the left or right of the center of mass to maintain or regain balance (Rankine 1869). As terrain becomes more rugged, these continuous paths for support become less frequent to the point where only discrete, unevenly spaced areas of support exist in the most rugged terrain (Holste and Ciccimaro 2009). Legged vehicles have improved mobility over rugged terrain, because legged vehicles may choose discrete, optimal foot placement and vary the length of the leg with respect to the body to traverse areas of ground that do not have a continuous path of support or closely spaced footholds. Moreover, legged vehicles are able to move in man-made or cultural environments, traversing obstacles such as curbs, stairs, and narrow passageways. With respect to wheeled vehicles traversing irregular-height terrain, legged vehicles reduce body motion. This characteristic is especially well suited to the comfort of a rider or passenger. Furthermore, unlike the wheeled motorcycle, a feature and advantage of legged vehicles is the ability to move sideways to achieve footholds that wheeled vehicles cannot. For example, in the case where all three legs are in contact with the ground, the ability to shift one of the legs laterally to affect balance is a significant advantage or design feature.

This research and dissertation contributes a novel Single-Track Three Legged Mobile Robot (ST3LMR) and method of operation to advance the art of legged robot design. As such, it opens up new research opportunities in robotics and human-robot interface. The motivation for the ST3LMR was to design a legged robot capable of carrying a rider with the semi-autonomous path planning and foot placement of a horse and the compactness and endurance of a motorcycle. This dissertation presents the ST3LMR proof-of-concept. While this research did not demonstrate a full-scale ST3LMR that people ride, it nonetheless provides an incremental step towards a ST3LMR that people would ride. This chapter introduces the ST3LMR design, provides an overview of the research, and briefly outlines the content of the following nine chapters and four appendices.

1.1. ST3LMR General Description

The ST3LMR is a robot that achieves the form and function of a motorcycle but with the added benefit of legs and partial or fully automatic stability of balance (like a horse). The term ‘robot’ is to be construed broadly and includes any means of vehicular transportation, whether merely of itself or of objects other than itself, relating to a device and method that works autonomously or semi-autonomously. The term ‘machine’ or ‘vehicle’, as in the motorcycle or bicycle, relates to an operated device. As shown in Figure 1.1, the ST3LMR comprises a body and three articulated legs arranged one behind the other in a narrow profile to walk and maneuver along narrow trails and paths, by placing successive footfalls in a generally single-track or in-line fashion. The term ‘single track’ refers to the general narrowness of the foot-placement patterns developed on the ground when moving along a straight-line or curved path. The ST3LMR achieves stability of balance without motion by positioning its three legs in a tripod stance. During locomotion, the feet are dynamically placed to the left or right of the support line (projected center of mass) to maintain (or regain) stability of balance.

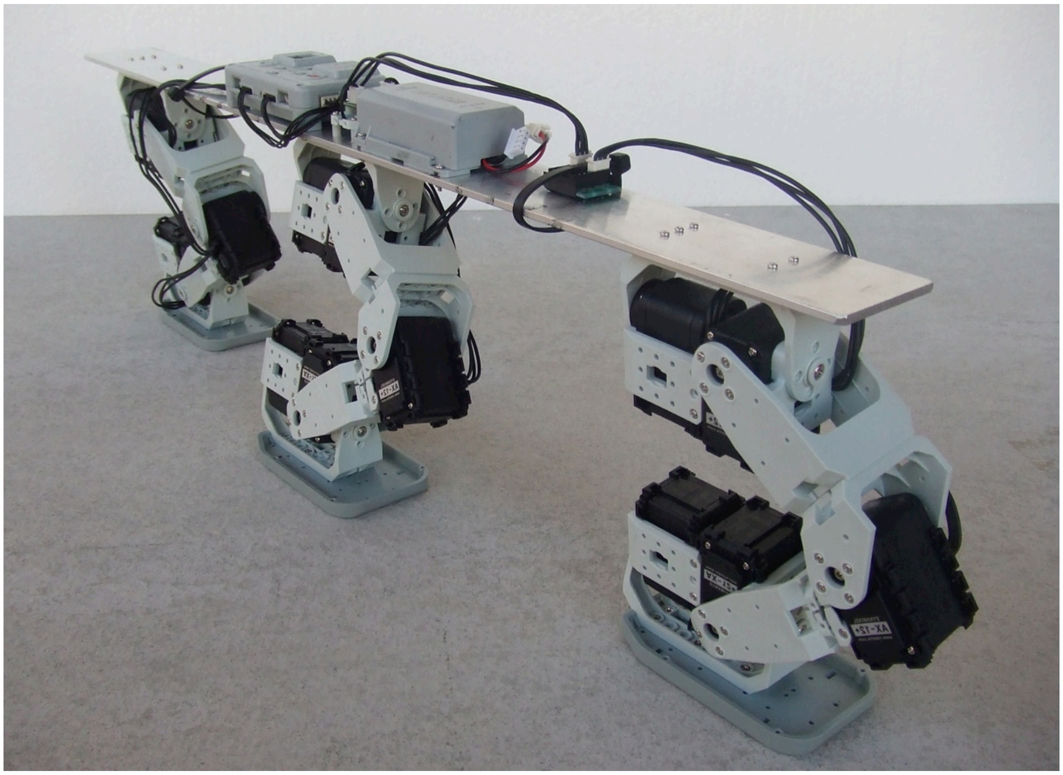


Figure 1.1. The ST3LMR proof-of-concept prototype with conventional hip-knee-ankle legs made from Bioloid robot kit parts.

Like the motorcycle, the ST3LMR executes a single-track turn by leaning the body into the turn, using gravitational forces to counteract the outward centripetal force. Unlike a motorcycle, the feet of the ST3LMR are stationary with respect to footholds during the support period, thus eliminating the drawback of wheels, which require continuous support (especially in rugged terrain). While not part of this research, it is envisioned that hopping, bounding, leaping, and jumping would enable the ST3LMR to traverse terrain that is too difficult for comparable wheeled and tracked vehicles.

1.1.1. Mechanical and Control System Overview

The elements of the ST3LMR comprise both mechanical, electrical, and software components. Mechanical elements include a body with three legs mounted on the body and arranged one behind the other in a narrow co-linear or co-planar profile, as shown in Figure 1.1. Each of the legs is actuated for movement in three dimensions. Each leg mechanism includes a foot at its distal end with an ankle actuator. Each leg mechanism provides position, force, and torque measurement feedback.

In addition to the mechanical elements, electrical elements include an operator interface, a control system, and various sensors. Note that the operator interface may comprise high-level commands from a hierarchical robot control system, rather than an operator. The control system includes at least one accelerometer and at least one gyroscope mounted on the frame and in communication with the control system. The control system receives sensed data from at least one accelerometer and at least one gyroscope, generally called an Inertial Measurement Unit or IMU. The IMU senses velocity, acceleration, attitude, and gravitational forces, and it measures body roll, pitch, and yaw angles and angle rates. Figure 1.2 diagrams the aforementioned ST3LMR elements and shows how they are interconnected.

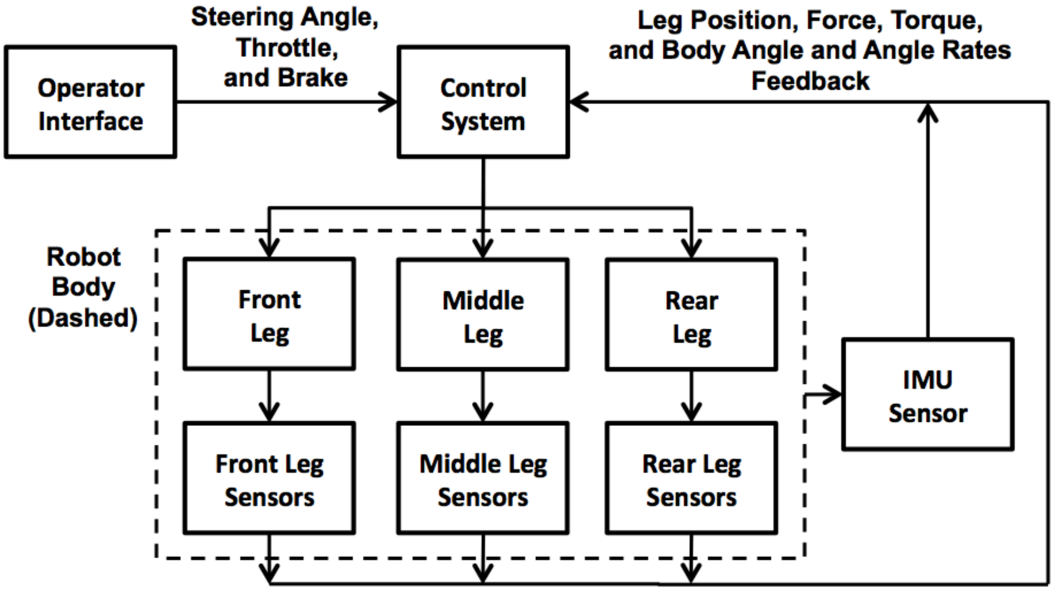


Figure 1.2. The ST3LMR mechanical and electrical system, depicted in a high-level block diagram showing element interconnectivity.

Mechanically, the ST3LMR body is generally longer than it is wide, with the long or major axis corresponding to a primary direction of travel, such as forward and backward motion. The leg attachment points are arranged in-line, one behind the other, with respect to the body. Each of the three legs has at least three degrees of freedom (DOF), such as pitch and roll at the hip and extension and retraction of the foot, to position the foot anywhere within a three dimensional volume. Each of the legs includes actuators attached between the legs and the body and between adjacent leg members. The legs are actuated for movement of the distal end in three dimensions and generally in-line with the major direction of motion.

The three legs combine to form three spatial volumes for foot placement that is spatially arranged to be generally in-line with the major direction of motion. The three spatial volumes overlap along the major direction of motion. The three legs have sufficient reach in length, width, and height to afford the three feet to be spatially positioned 1) in a triangular pattern to keep the body in static equilibrium at rest, 2) in any manner of patterns to provide locomotion and dynamic attitude stabilization, and 3) for omnidirectional motion.

The IMU and leg position, force, and torque feedback describe the current robot state. The current state is combined with the operator input (goal state) by the control system to plan the leg and foot motion for both locomotion and to balance the ST3LMR. The ST3LMR also has a power source, which is not shown in Figures 1.1 or 1.2 for clarity. The power source connects to the control system and the plurality of actuators and joints, which drive the legs.

1.1.2. Walking Motion Overview

Legged vehicles heretofore must simultaneously maintain stability of balance in both the pitch and roll directions. For the ST3LMR, a walking gait is composed of successive phases of two-leg support and brief periods of three-leg support. Because at least two legs are supporting the ST3LMR (along the length of the body) at all times, the ST3LMR is inherently stable along its major axis of motion (i.e., the pitch axis). As illustrated in Figures 1.3 and 1.4, the ST3LMR is shown executing a Backward Wave Gait (BWG). One complete cycle of the BWG is illustrated in the six sub-drawings of Figure 1.3 (top to bottom, 1 through 6, repeating) and again in Figure 1.4 (top to bottom, 7 through 12, repeating).

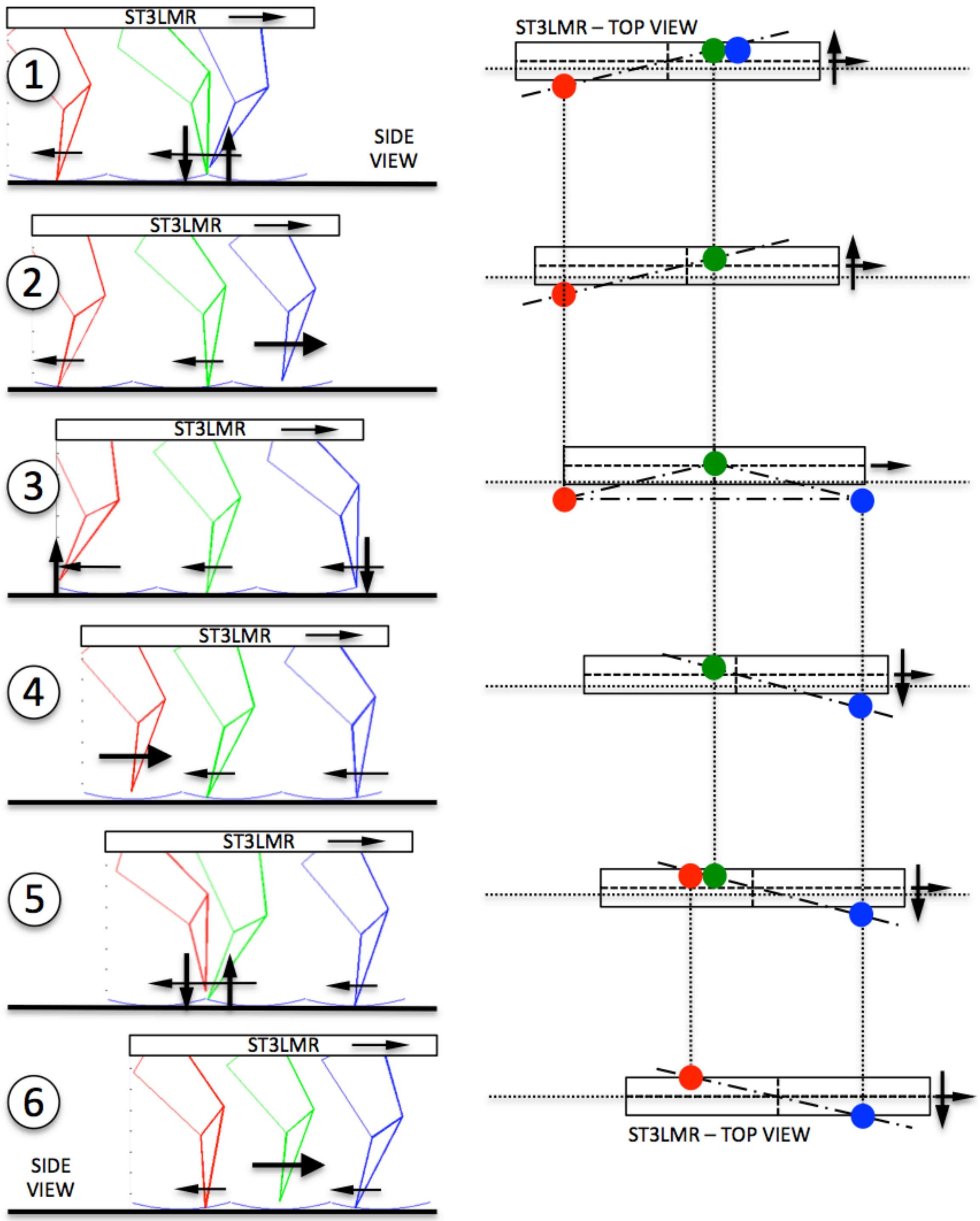


Figure 1.3. The ST3LMR is illustrated in side view (left) and top view (right) executing a follow-the-leader BWG (top to bottom, 1 through 6, and continued in Figure 1.4) while simultaneously shifting the body.

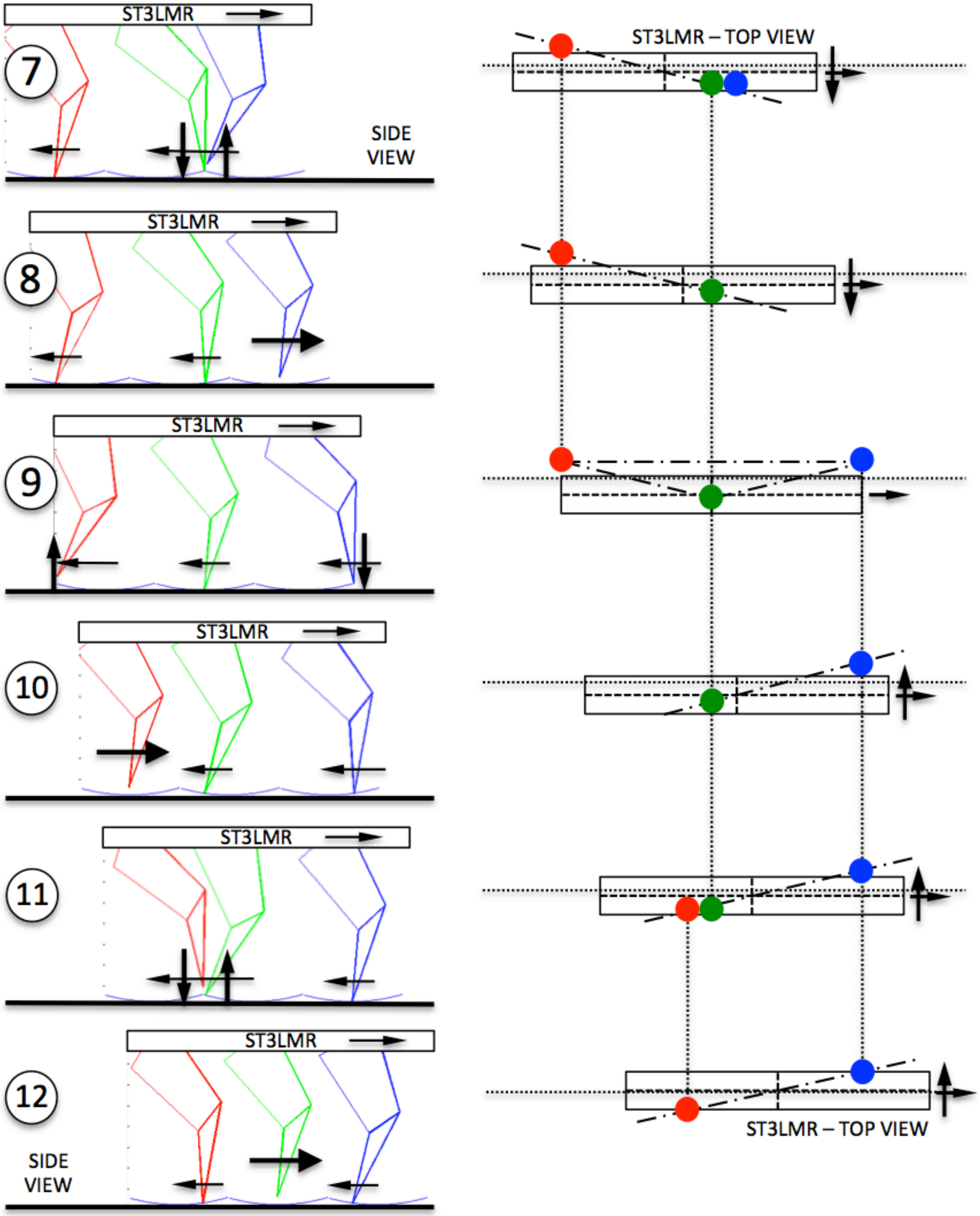


Figure 1.4. Continued from Figure 1.3, the ST3LMR is illustrated in side view (left) and top view (right) executing a follow-the-leader BWG (top to bottom, 7 through 12) while simultaneously shifting the body.

Figures 1.3 and 1.4 also illustrate the ST3LMR executing a follow-the-leader foot placement strategy. In this strategy, only the position of the first or front foot (blue leg/foot in Figures 1.3 and 1.4) is uniquely determined by the control system. The second or middle foot follows in close proximity to the footstep of the first foot, and the third or rear foot follows in close proximity to the footstep of the second foot. As the first foot oscillates from left to right with each step (sub-drawings 3 and 9 in Figures 1.3 and 1.4), it provides brief periods of three-leg or tripod stance support. Note that a make-before-break, velocity matching leg/foot transition is shown in sub-drawings 1, 3, and 5. Such leg/foot transitions provide smooth, uniform body motion. In simulation, the tripod stance was found to be key to maintaining stability of balance after an applied disturbing impulse or during an obstacle avoidance maneuver. Chapters 4 through 6 provide detailed information regarding single-track gaits, motion and temporal planning, and the control system.

Figures 1.3 and 1.4 also illustrate increasing time or distance traveled, from left to right. The side view (left) of sub-drawing 2, for example, depicts the ST3LMR moving to the right, the middle and back legs on the ground (in stance phase) and moving to the left, with respect to the body, and the front leg moving fast to the right in the air (in flight phase). The top view (right) of sub-drawing 2, for example, depicts the ST3LMR moving from left to right in the forward direction of motion and the body shifting from right to left. For low-speed walking, shifting the center of gravity along the line of support (between the green middle foot and red rear foot in sub-drawing 2) maintains stability of balance. That is, the two feet act like a hinge whereby the ST3LMR can fall to the left (counterclockwise) or right (clockwise) due to gravitational forces. In practice, a small foot with an ankle actuator is generally required for slow-speed walking. The feet widen the line of support, providing margin for errors in positioning the body and to handle unexpected disturbances.

Using the BWG, two legs support the ST3LMR at all times. This method allows for the decoupling of leg positioning along the length of the body or major axis of motion from the leg positioning along the width of the body or normal to the major axis of motion. That is, it is possible to control stability of balance in the roll direction and (for the most part) not in the pitch direction. This characteristic drastically simplifies real-time control for many single track or in-line legged gates and modes of operation.

As forward speed increases to a fast walk, the side-to-side oscillation of the body is reduced through dynamic stabilization. That is, the rate of foot placement to the left or right of the body increases with increasing speed while the counterclockwise and clockwise torque due to gravity and thus the rate at which the body falls to the left or right remains constant. As forward speed increases to infinity, (ballistic) foot placement approximates a single-track line of support (like spokes on a wheel) that can be brought under the center of gravity to keep the robot upright, just like a motorcycle. Figure 1.5 graphically illustrates the periodic torque developed throughout the walking cycle, given a left-right oscillating front foot placement in a follow-the-leader BWG.

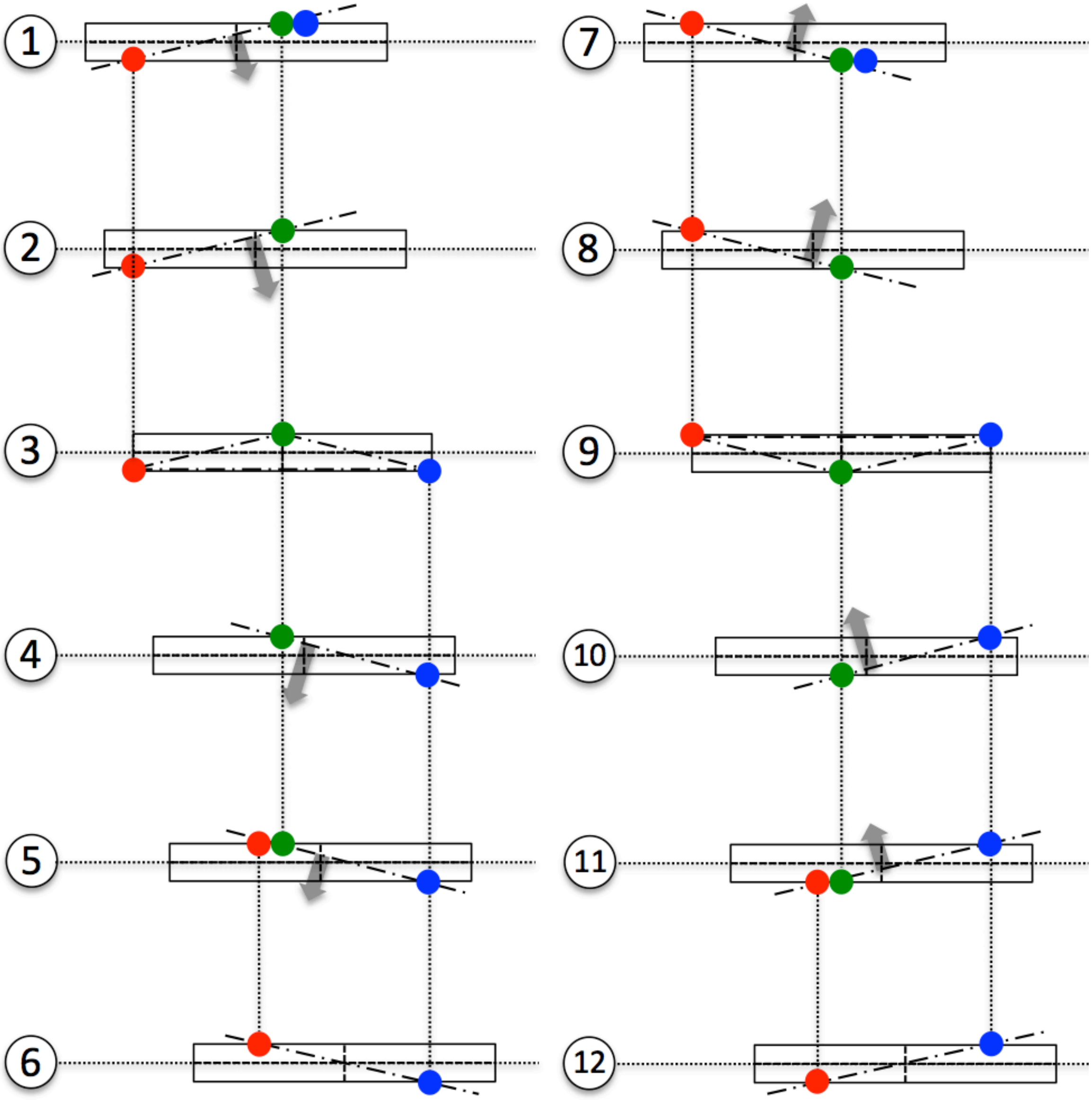


Figure 1.5. The ST3LMR illustrated in top view executing a follow-the-leader BWG (top to bottom, left to right, 1 through 12, repeating), illustrating the periodic torque developed throughout the walking cycle, given a left-right oscillating front foot (blue) placement.

An inertial measurement unit mounted on the body senses body attitude and in particular roll angle, angle rate, and angle acceleration. Joint angle, force, and torque sensors on each leg and foot determine the kinematic properties. A temporal control system uses an inverted pendulum model to plan future leg/foot placement to maneuver, avoid obstacles and contain roll. Like wheeled motorcycles and bicycles, the ST3LMR executes a single-track turn whereby the body leans into the turn, thus developing a torque about the roll axis to counteract the outward centripetal force. The ST3LMR controls stability of balance in the roll direction and (for the most part) not in the pitch direction. Like a quadruped (or horse), yaw is controlled by developing torque about a single foot, any two legs, or the combination of all three legs during the stance phase. Furthermore, the control system senses and prevents turnover of single track legged mobile robots while enabling normal riding techniques in all but out of control situations.

1.1.3. Foot Placement Planning System Overview

This dissertation follows established computer dynamic programming methods, wherein the controlled process (e.g., balancing the ST3LMR) is architected within the control element (e.g., the ST3LMR dynamics) to iteratively estimate an optimal body trajectory over the span of sensed information (Johnson 1965; and Sheridan 1966). As opposed to developing a closed-form analytic solution, the Monte Carlo (MC) method is used to investigate all possible foot placement options. Such computer modeling or simulation divides time into small increments to estimate a process. Figure 1.6 depicts the ST3LMR in perspective view executing a follow-the-leader BWG.

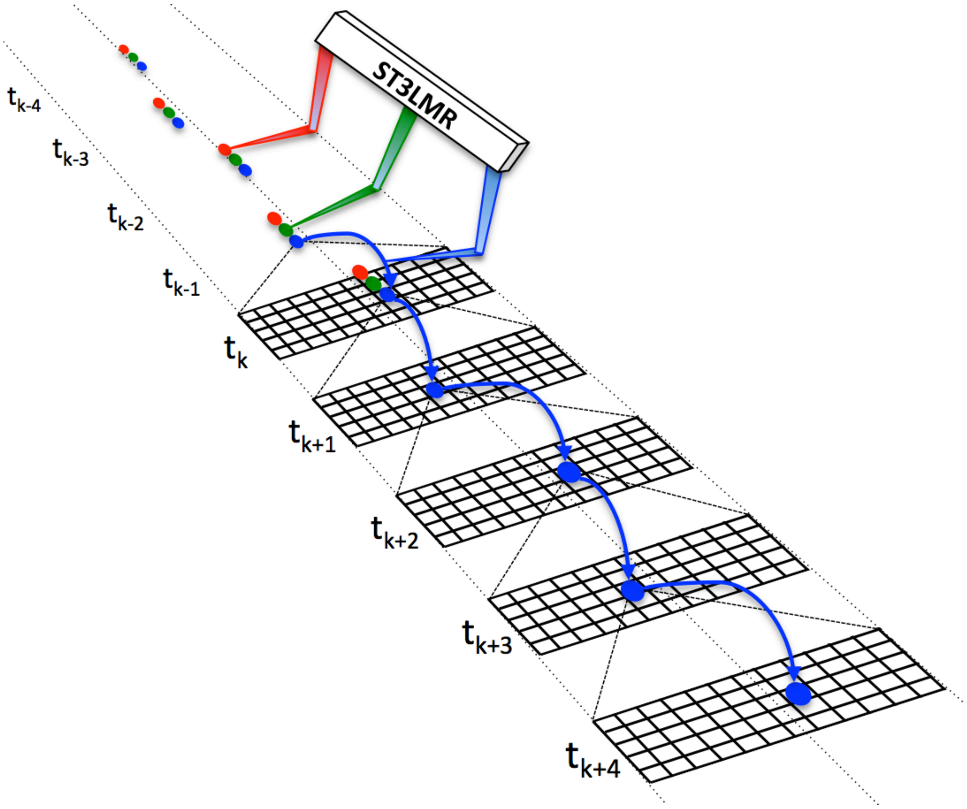


Figure 1.6. The ST3LMR depicted in perspective view executing a follow-the-leader BWG wherein a grid of possible front foot placements is shown four steps into the future.

As depicted, the ST3LMR is about to place the front foot (blue) on the ground at time t_k (step 9 in Figures 1.4 and 1.5). A grid of possible foot placement options is shown four steps into the future (to t_{k+4}). A high-speed MC simulation is used in a model-predictive control (MPC) system to determine a foot placement strategy that provides locomotion and long-term stability of balance. Search breadth corresponds to the resolution of foot placement in the side-to-side and forward-backward directions, and search depth corresponds to the N-steps ahead in time being simulated. Chapters 5 through 7 discuss the control method in more detail.

1.2. Present Study and Contribution

This dissertation researches, develops, and discusses a proof-of-concept for the invention entitled 'In-Line Legged Robot Vehicle and Method for Operating,' United States patent number 8,457,830 and the disclosure of a two-time-scale temporal tracking system architecture, US patent application publication US 2011/0231016 A1. The invention has been prototyped using parts from servo and robotics kits as well as custom-made components. Figure 1.7 shows the Draisine prototype robot, designed and built to test single-track control concepts (as a compromise to the expense of building and testing a three-legged robot). In addition, a simple computer simulation and a two-time-scale Monte Carlo type model-based temporal robot control system have been architected.

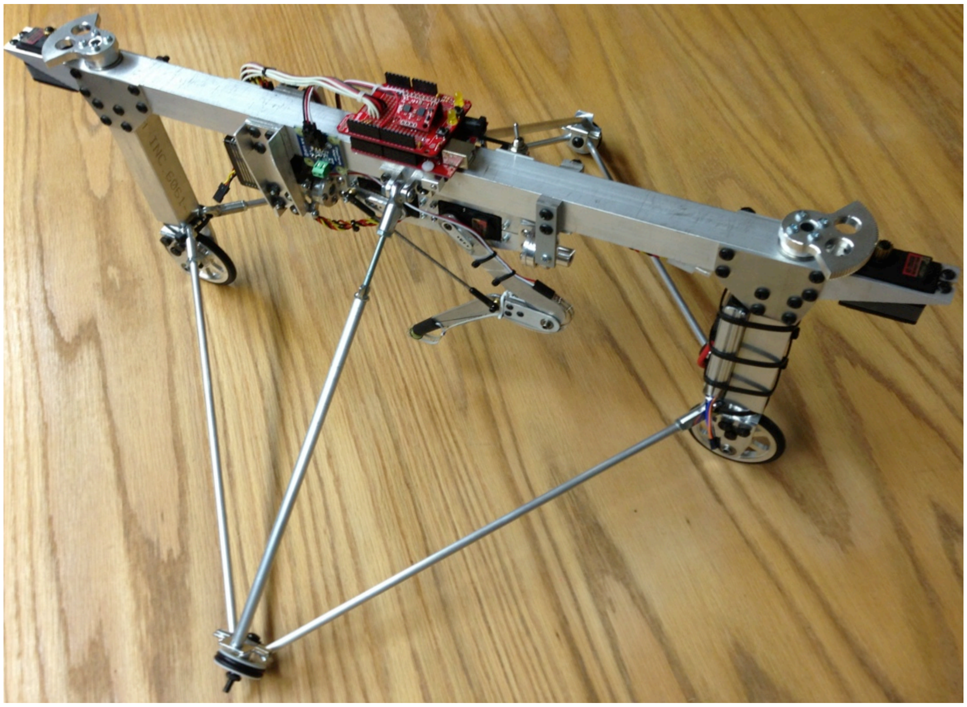


Figure 1.7. A wheel-leg-wheel prototype Draisine robot with a custom-built Hirose type leg.

The results of this dissertation are obtained from the physical models, digital simulation, and analysis. Control aspects are tested and various configurations of models are discussed. In solving this problem, the author has contributed the following:

1) Invented a new type of legged robot with an in-line or single track leg configuration that achieves the desired form and function of a motorcycle but with the added benefit of legs and full or partial robotic control;

2) Invented new gaits for three in-line or single-track leg locomotion whereby repeated intervals of a bipedal and/or tripedal stance provide for roll and pitch control;

3) Decoupled roll and pitch control using the three-legged gaits to simplify the control requirements of legged robots through separate algorithms for forward motion (gaits) and stability of balance (in the roll axis);

4) Architected a two-time-scale Monte Carlo type model-based temporal robot control system to plan the footholds and provide autonomous attitude stabilization control;

5) Demonstrated maneuverability and control through experimental results from physical prototypes and a simple digital simulation of an impulse response, an avoidance maneuver, and a lean-into-the-turn cornering maneuver;

6) Demonstrated a lean-into-a-turn maneuver (with respect to the body and primary direction of motion) for a legged mobile robot; and

7) Demonstrated a lean-into-a-turn maneuver (with respect to the body and primary direction of motion) for a legged mobile robot wherein stability of

balance is controlled in the roll direction and not in the pitch direction and torque is developed about any two or three legs during the stance phase to control the direction of motion in the yaw axis.

1.3. Dissertation Organization

This dissertation is divided into ten chapters and four appendices. This chapter introduced the ST3LMR as a robot that achieves the form and function of a motorcycle but with the added benefit of legs and partial or fully automatic stability of balance (like a horse). The ST3LMR concept addresses legged motion over rough, irregular terrain. Chapter 2 provides background prior art legged vehicles, robots, robot leg mechanisms, and legged robot control systems. Chapters 3 through 7 present detailed concepts of operation. Chapter 8 and the appendices present results from several prototype robots. Chapters 9 and 10 summarize the research and present a vision for possible future work.

Specifically, Chapter 3 presents a detailed design description and operation of the ST3LMR, and it includes a discussion of design novelty. Chapter 4 presents various single-track gaits for the ST3LMR and discusses various methods for traversing irregular terrain. Chapter 4 also provides a detailed example of how the ST3LMR traverses a vertical step. Chapter 5 discusses ST3LMR motion planning by introducing a control strategy and architecture for single-track robots, presents single-track leg/foot planning methods, and develops a probabilistic multi-hypothesis foot position planning system using branch-based scoring in a depth-limited search for dynamic balance. The results from a simple digital simulation are presented in Chapter 5, and Appendix B presents the

MATLAB code used in the simple digital simulation. Chapters 6 and 7 introduce the temporal tracking and planning approach, architecture, and model-based temporal control system of the proposed ST3LMR.

Chapter 8 demonstrates the ST3LMR proof-of-concept through two physical prototypes, providing the advantage of gait-based inherent stability along the pitch axis, and decoupling and simplifying control of balance to only the roll axis. Appendices B and C present experimental results obtained from testing a Draisine wheel-leg-wheel prototype robot, providing design feedback for body and leg dynamics, foot placement, control algorithms, foot pressure sensors, inertial measurement sensors, and real-time control systems. Appendix D presents the MicroBasic program used for the Draisine wheel-leg-wheel robot.

Finally, Chapters 9 and 10 conclude by discussing the specific contributions made through this research and identifying various topics for future research. For example, the degree to which the rider provides balance control of the roll axis (like a motorcycle) versus the robot providing stability of balance (like a horse) is proposed for future research.

CHAPTER 2

BACKGROUND

The first subsection of this chapter provides a historical background of various ideas, technology, vehicles and robots related to the Single-Track Three Legged Mobile Robot (ST3LMR) and especially those adapted for moving over rough or uneven terrain. The second subsection details various leg mechanisms developed for legged vehicles.

2.1. Prior Art

The first single-track vehicle, circa 1819, was the two-wheeled Drais Lauf-maschine or Draisine (Figure 2.1), a pedal-less precursor to the bicycle (Drais 1819 and 1832). The term 'single track' refers to the general narrowness of the wheel patterns developed on the ground when moving in a straight-line or curved path. Wheeled single-track vehicles are inherently unstable and will fall over without forward motion and some type of active control mechanism (e.g., a rider) to contain accelerations in the roll axis.

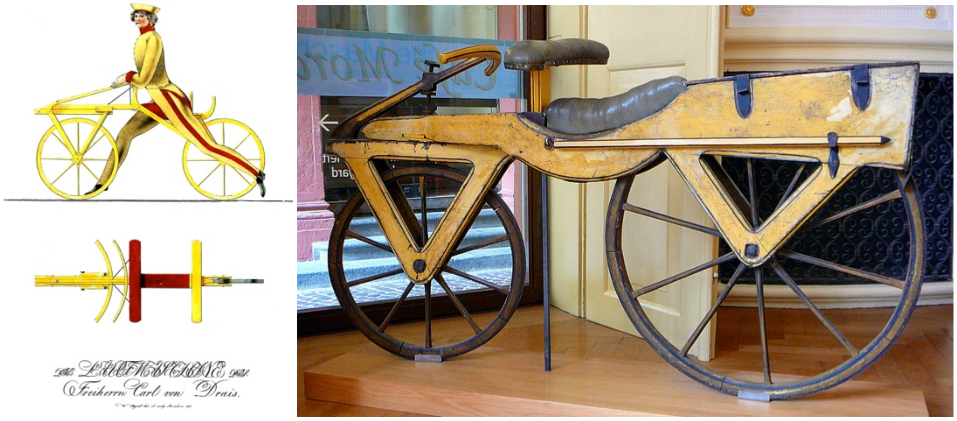


Figure 2.1. The two-wheeled Drais Lauf-maschine or Draisine, circa 1820. (Images courtesy Museum der Stadt Heidelberg)

As early as 1898, H.G. Wells described a fictional 100 foot tall, three-legged mobile robot in his science fiction novella entitled “War of the Worlds,” drawn by Warwick Globe circa 1898. In the drawing, Figure 2.2, the three legs are symmetrically positioned in a triangular pattern to form a tripod stance. About that same time, Muybridge used stop-motion photography to study legged locomotion in animals (Muybridge 1899) and later humans (Muybridge 1901). His work provided a method for structuring classical quadruped and biped walking gaits in biologically-inspired legged machines.

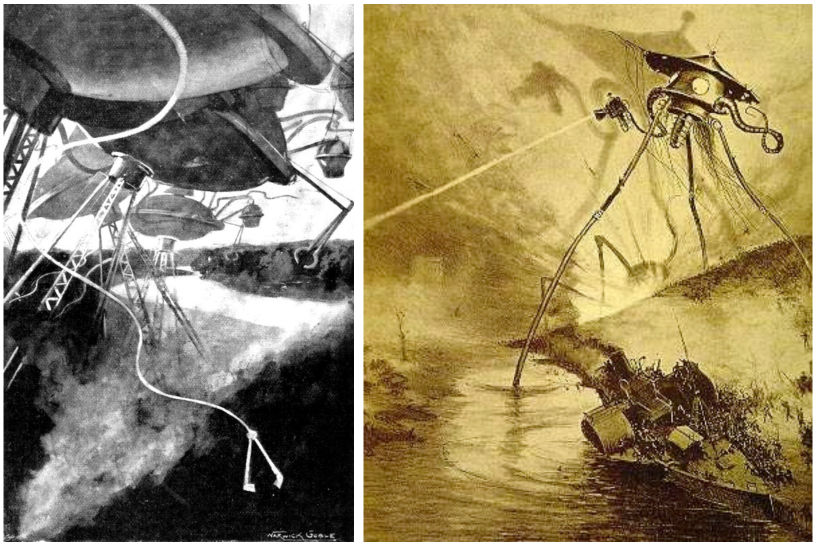


Figure 2.2. Martian tripods (left) drawn by Warwick Globe, circa 1898, and tripod ship from the 1906 French edition of H.G. Wells book (right). (Images courtesy Wikipedia)

From the early to mid-1900’s, legged locomotion used mechanical, cam-linkage type mechanisms. Figure 2.3 illustrates a Deutsches Reichspatent drawing of a 1913 walking vehicle that mimicked a quadruped trot, moving opposing legs in pairs (e.g., front right and rear left) with periods of support from all four legs. Shigley investigated four-bar linkages, cam linkages, pantograph mechanisms, and so on, and he built a mobile robot with four rectangular frames, controlled by a set of double-rocker linkages, using non-circular gears to produce uniform walking velocity for the Army (Shigley 1960). About that same time, the Space General Corporation developed a hexapod (McKenney 1961) and an eight-legged mobile robot (Baldwin and Miller 1966) for lunar rover application. Both mobile robots were controlled . Both the Shigley and Space General Corporation designs employed statically stable, symmetric walking gaits, and required moving pairs of opposing legs to keep the body in static equilibrium at all times.

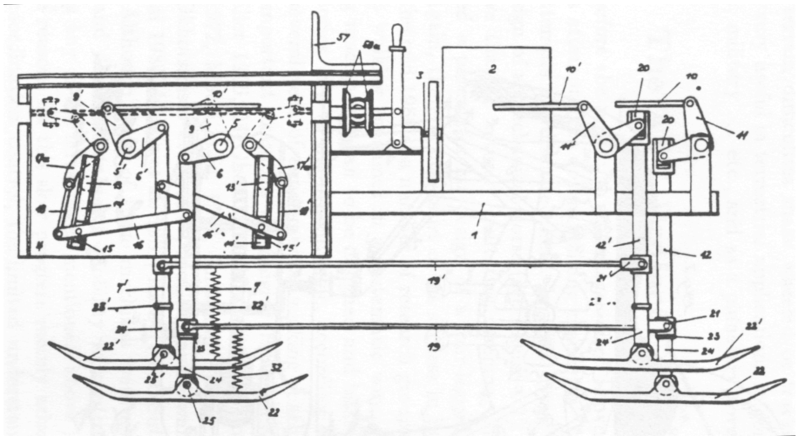


Figure 2.3. DRP 554354 patent drawing of a 1913 walking vehicle. (Image source Deutsches Reichspatent)

The aforementioned prior art vehicles are generally very large, bulky, and cumbersome, and such prior art legged vehicles generally move slower than comparable wheeled vehicles. This speed constraint highly limits their usefulness. By the mid 1960's, however, hydraulic actuators were in use, and it was believed that legged locomotion would increase the speed of vehicles traversing unimproved or rough terrain by a factor of 10x (Bekker 1960). That is, animals were observed traversing rough terrain at 35 mph while wheeled vehicles managed only 3-5 mph. Additionally, legged locomotion promised better isolation from terrain irregularities. Frank Mosher of General Electric Corporation built a 3000 pound, hydraulically-actuated quadruped that had three DOF per leg, two DOF at the hip and one DOF at the knee (Listen and Mosher 1968; and Mosher 1968). Shown in Figure 2.4, their quadruped was controlled by a human operator, and it demonstrated that legged machines could move effectively on rough terrain and climb over obstacles, with a human providing control and sensing.

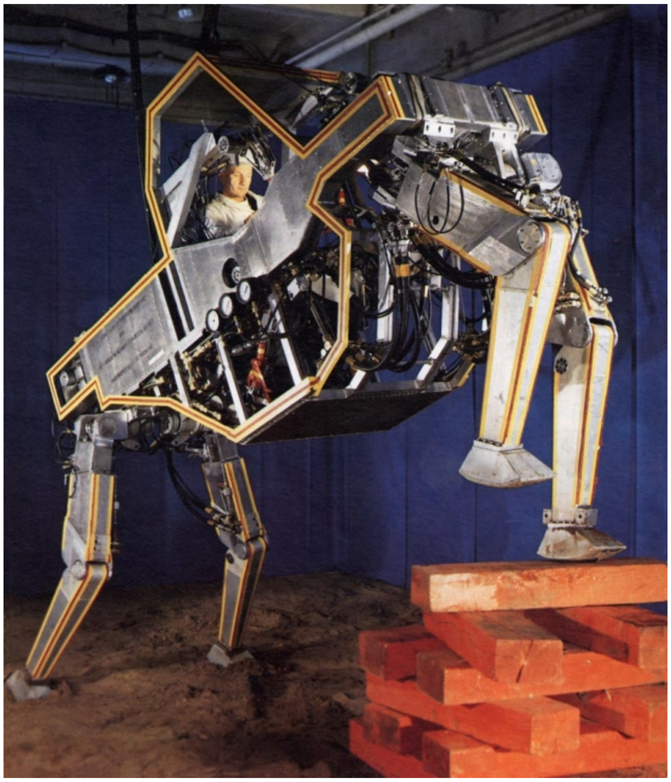


Figure 2.4. Frank Mosher riding his four-legged hydraulic walking machine in 1968. (Images courtesy Reuben Hoggett, cyberneticszoo.com).

Also about that time, gaits for legged locomotion were modeled both mathematically and diagrammatically (Tomovic and Karplus 1961; and Hildebrand 1965 and 1967). From this research, fundamental terminology was defined, such as stance, swing, stride length, duty factor, phase, stability, and so on. For example, as illustrated in Figure 2.5 (and discussed in Chapter 4), a leg is either on the ground, called the support or stance phase, or in the air, called the swing or flight phase. Stride measures the distance the body moves in one stance-to-swing locomotion cycle. Such efforts led to the development of various theories and algorithms for coordinating leg movements in bipeds, quadrupeds, hexapods, and other symmetric legged mobile robots to walk over rough terrain, evaluate footholds, and walk outdoors on various types of terrain.

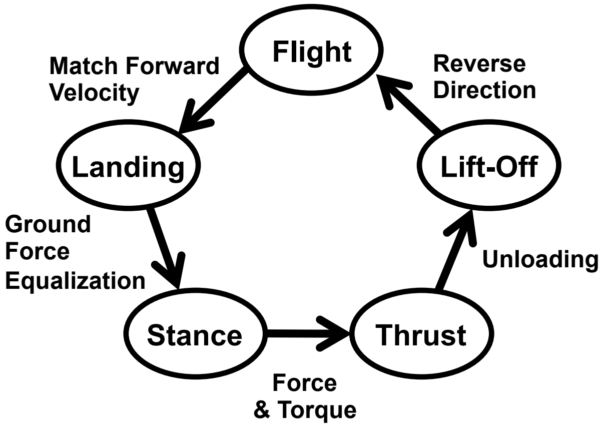


Figure 2.5. The fundamental stance-flight leg cycle. (Adapted from Figure 2.8 in Raibert 1986).

Further developments in legged mobile robots were made by Frank and McGhee at the University of Southern California when they used a computer to control the motion of an eight degrees-of-freedom (DOF) quadruped (McGhee 1966; and Frank 1968).

Their quadruped had two degrees-of-freedom (DOF) for each leg, one DOF at the hip and one DOF at the knee, with independent electromechanical actuators at each leg joint. Using the computer to coordinate or orchestrate the leg joint movements, it demonstrated the classical quadruped walk and trot gaits. In 1968, McGhee and associates at the Ohio State University proved mathematically that there is an optimal gait for a quadruped that maximizes the longitudinal stability margin (McGhee and Frank 1968). They built a 300 pound hexapod that used force sensors, gyroscopes, proximity sensors, and a camera system (Figure 2.6) to study control algorithms for legged mobile robots (Buckett 1977; Briggs 1977; and Pugh 1982).

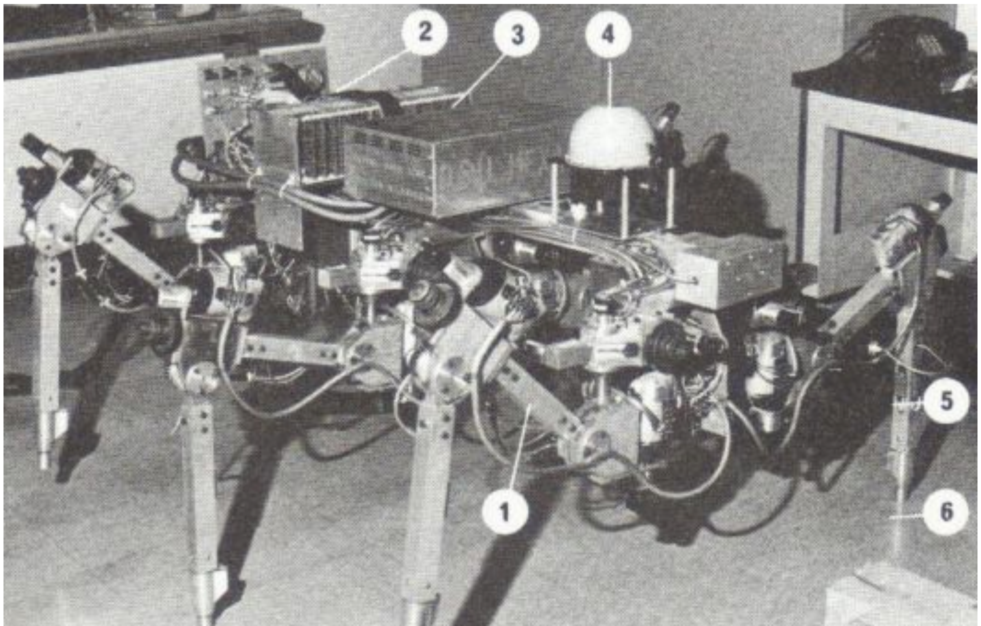


Figure 2.6. 1976 OSU Hexapod, showing (1) 3-axis leg, (2) servo control electronics, (3) digital transmission interface, (4) gyroscope, (5) strain gauges for leg feedback, and (6) foot contact sensor. (Images courtesy Jean Vertut, cyberneticszoo.com).

In 1974, Seireg at the University of Wisconsin, Madison developed a three leg robot powered by compressed air. The robot was originally designed as a biped to study prosthetics, but because it weighed 260 pounds, a third leg was added to provide extra stability (Sanborn 1971). Figure 2.7 shows both a sketch and picture of the robot achieving a tripod stance. The legs are connected to the body at the hip and are closely spaced and oriented along the width of the robot, with respect to the major direction of motion.

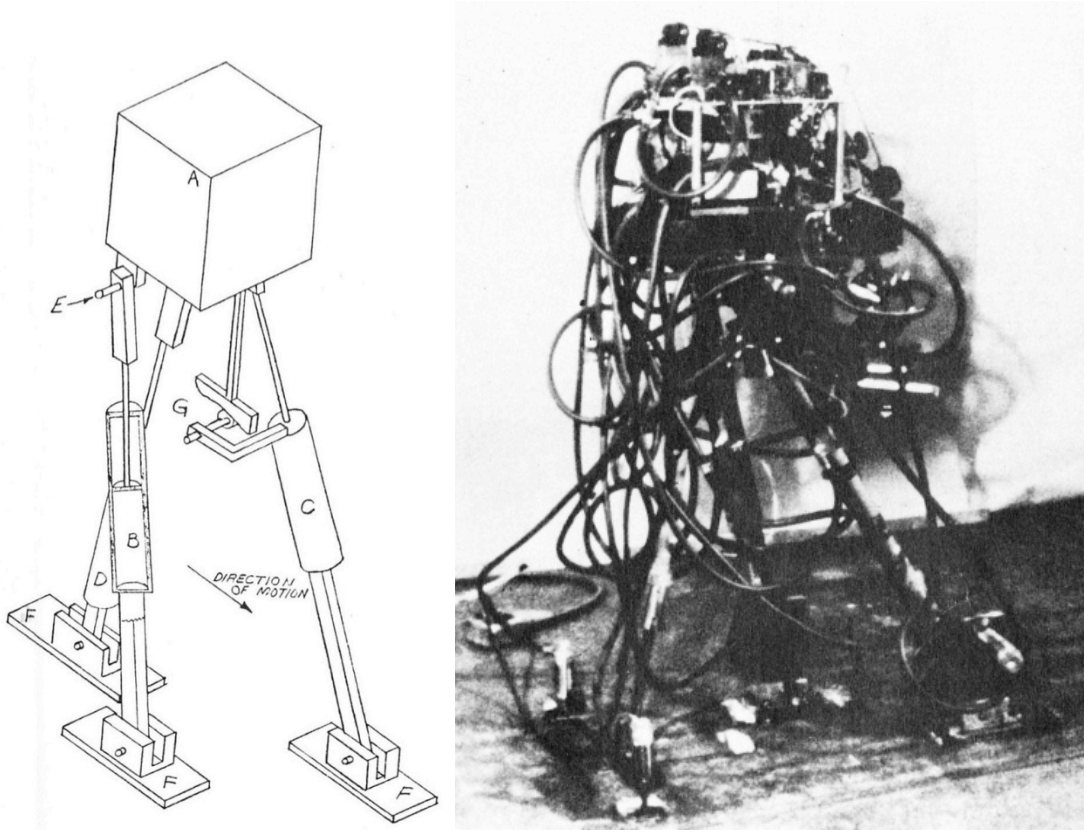


Figure 2.7. Seireg three leg robot sketch (left) and as-built photo (right). (Images courtesy *The Wisconsin Engineer*, vol. 77, no. 2, pp. 8-9, Nov. 1972).

In 1982, Ivan Southerland of Carnegie-Mellon University and Sutherland, Sproull and Associates, built a six-legged, gasoline powered, hydraulic walking machine, shown in Figure 2.8. It is credited as the first man-carrying, computer-controlled walking machine (Raibert and Sutherland 1983; and Sutherland and Ullner 1984). The computer controls the leg actuation, wherein the mechanical design of the hip actuators enables a single hydraulic valve to control forward/backward leg and foot movement. The rider steers by regulating hydraulic oil flow to the right or left side via a joy stick.

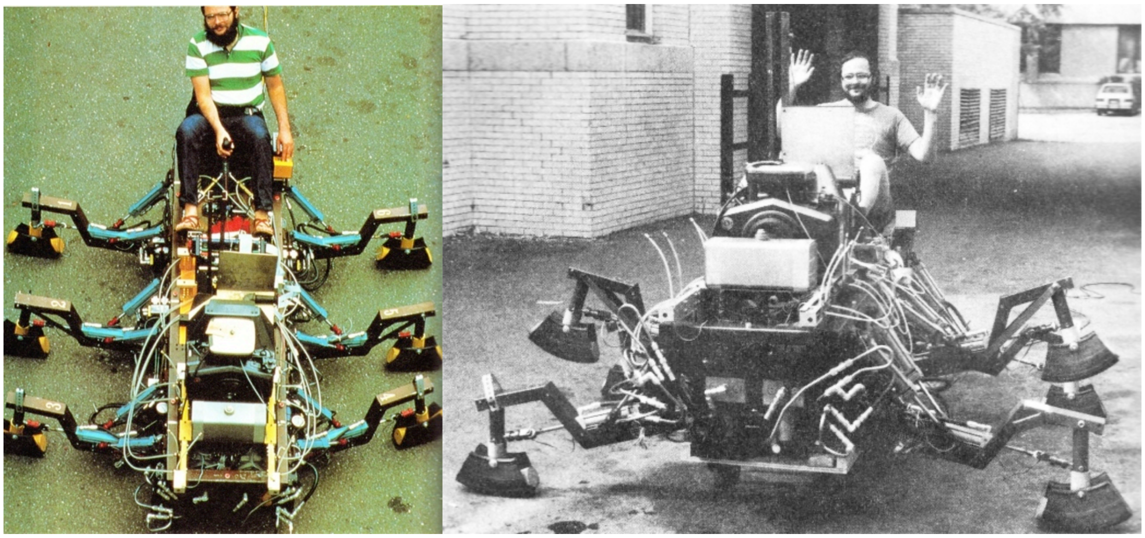


Figure 2.8. Ivan Sutherland riding six-legged hydraulic walking machine in 1982. (Images courtesy Reuben Hoggett, cyberneticszoo.com).

Further developments in legged machines were made by Pugh, Raibert, and Goldberg who investigated unstable or dynamic legged machines by studying balance of one, two, and four-leg hopping machines (Raibert 1984; Goldberg and Raibert 1987; and Hodgins 1989). Raibert investigated a one-leg hopping machine, shown in Figure 2.9, which is statically unstable and will fall down without constant placement-thrust movement of the foot to compensate for instability.

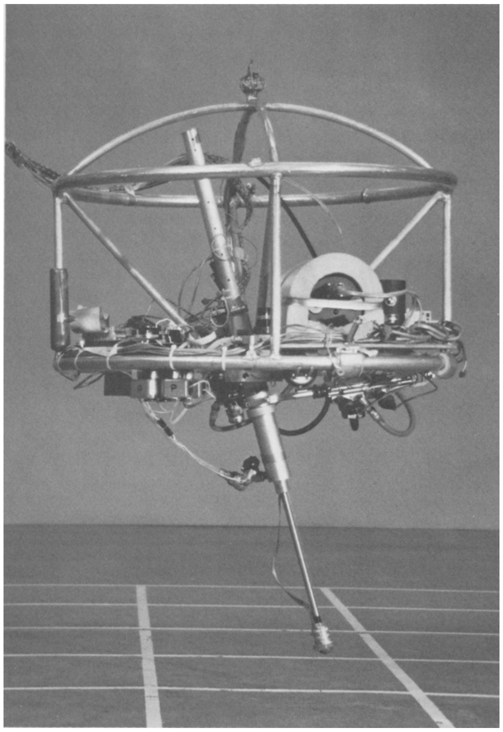


Figure 2.9. One-leg hopping machine shown balancing in three dimensions in 1983. (Image reproduced from Figure 1.9, Raibert 1986).

Raibert modeled the legged machine as an inverted pendulum and decomposed control into three separate elements: 1) supporting the body by controlling the vertical hopping height, 2) positioning the feet in key locations on each step using symmetry principles to keep the robots balanced, and 3) controlling the body attitude by controlling hip torque during the stance phase such that the dynamic momentum state of the body is estimated ahead in time to calculate the future foot placement and thrust needed to develop complementary dynamic momentum and achieve a desired hopping height, running velocity, and body attitude (Raibert 1984 and 1986). This seminal control system demonstrated dynamic re-stabilization against overturning when subject to unexpected forces that destroy the normal lateral balance of the vehicle, which could cause overturning, or when moving on unstable or slippery surfaces, the latter conditions causing foot slip to occur and could cause overturning.

Further developments in legged machines have been realized by improvements in low-power, high computational throughput, self-contained computer systems capable of receiving sensory input, calculating the system and leg kinematics, and controlling each leg joint. For example, Waseda University has built several bipedal robots designed for a rider. Waseda's WL-16RIV uses six actuators per leg, in a Stewart configuration, and is shown in Figure 2.10. See also (Suga and Yamaoka 2009).

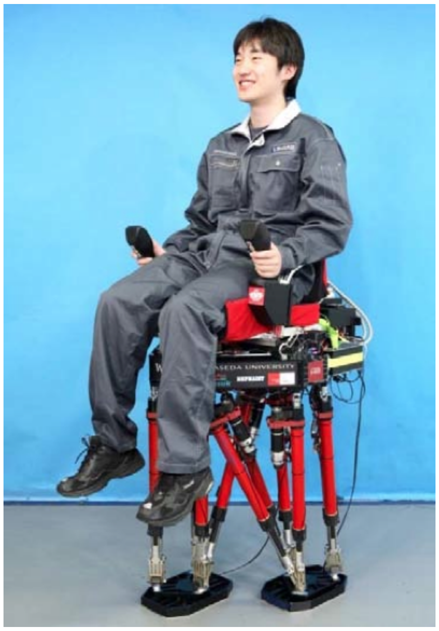


Figure 2.10. Waseda Leg No.16 Refined IV (WL-16RIV) shown with a rider. (Image reproduced from Figure 1, Hashimoto *et al*. 2008).

Bipedal designs subject the rider to oscillatory motion as the center of mass is shifted over each foot during each step (Takeda *et al*. 2001; and Hashimoto *et al*. 2008). The danger of overturning is increased when the legged vehicle is carrying at least one rider or passenger because the rider may make moves which can upset the control system, destroying the normal lateral balance of the legged vehicle and thus causing overturning of the legged vehicle. Waseda's approach involves the development of a factor of safety with regard to keeping the center of gravity within the center of pressure of the feet.

A novel tripod robot called STriDER (Self-Excited Tripedal Dynamic Experimental Robot), shown in Figure 2.11, was designed with omnidirectional legs and body such that the body rotates in the pitch and yaw axis allowing a leg to swing under the body to afford pairs of legs to contact the ground simultaneously (Heaston 2006). It balances using an estimate of its lateral velocity and acceleration, determined from the sensed behavior of the legs during stance combined with the inertial sensors. A high-level control system coordinates behaviors of the legs to regulate the velocity, attitude, and altitude of the body during locomotion.

Various experiments have been performed on quadruped robots to study walking gaits when one leg is inoperative (Mostafa *et al.* 2010). Such work concerns damaged quadruped robots, i.e., two legs are in-line and the third leg is offset with one of the in-line legs, as in a right-angle triangle orientation. The two offset legs walk in a predominately bipedal gait with the single in-line leg implementing a hopping motion. However, none of the aforementioned prior art was found to embody the ST3LMR design, conceptually illustrated in Figure 2.12.

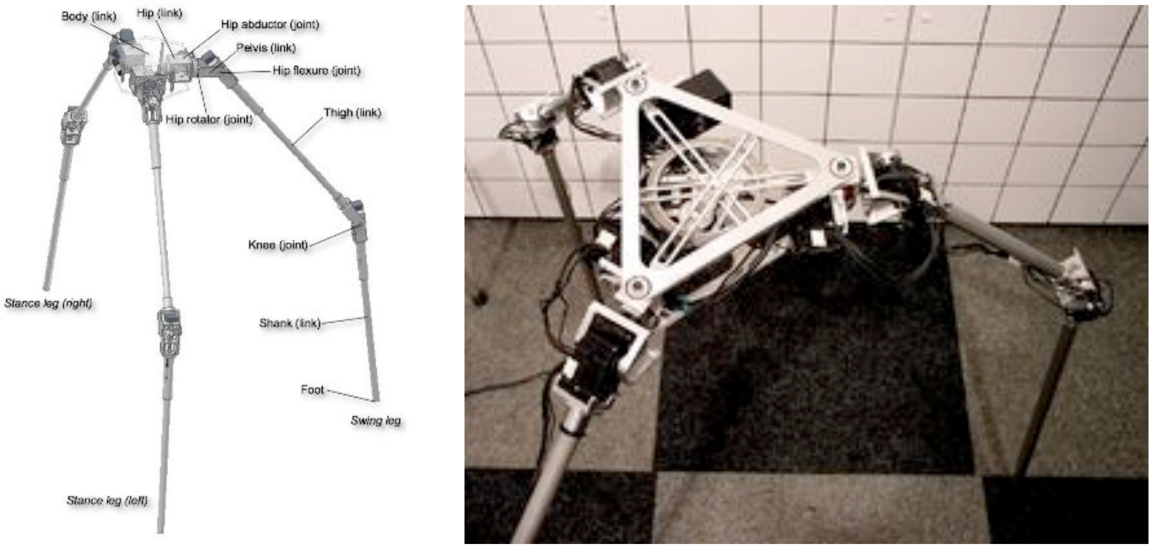


Figure 2.11. Self-Excited Tripedal Dynamic Experimental Robot called STriDER. (Image reproduced from Virginia Polytechnic Institute RoMeLa web site, 2011).

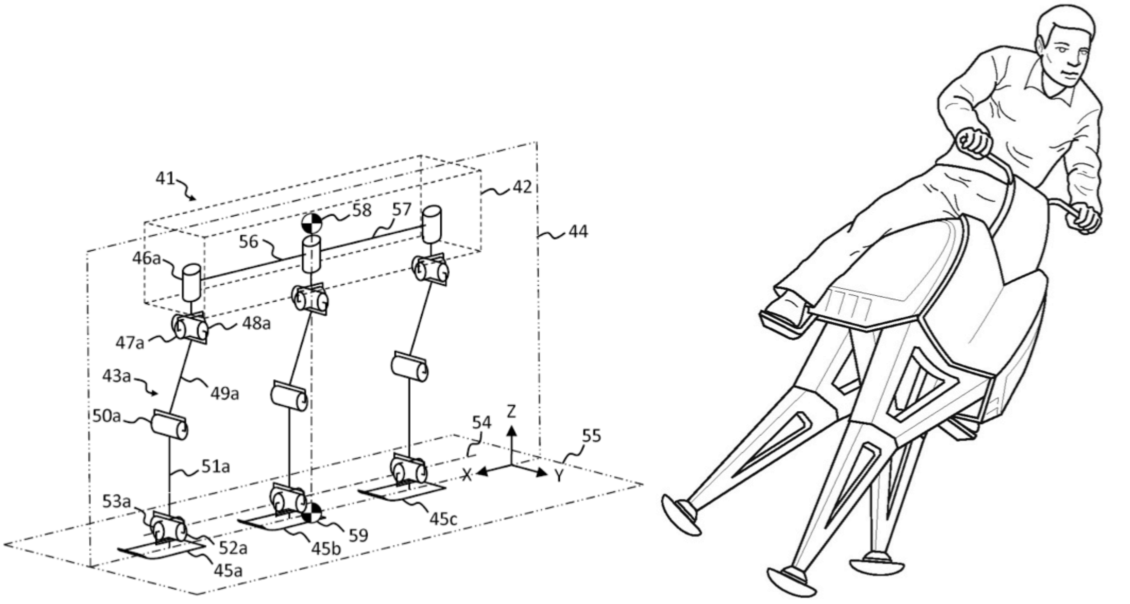


Figure 2.12. The Single-Track Three Legged Mobile Robot as shown in US patent 8,457,830 drawing 1 (left) and conceptually with a rider leaning into a turn (right).

2.2. Leg Mechanisms

A survey of the literature found numerous examples of four basic leg mechanisms: jointed, coupled, prismatic, and pantograph. The following subsections discuss these different leg types and present some typical examples for legged vehicles.

2.2.1. Jointed Leg Mechanisms

A jointed leg mechanism places the actuators on the joints, whereby the actuator housing and bearings may be integrated into the supporting leg structure itself. Similar to the leg mechanism used in the first-generation prototype, Figure 2.13 shows the University of Tokyo JSK Lab's robot with electric motors at each joint (Guizzo 2012).

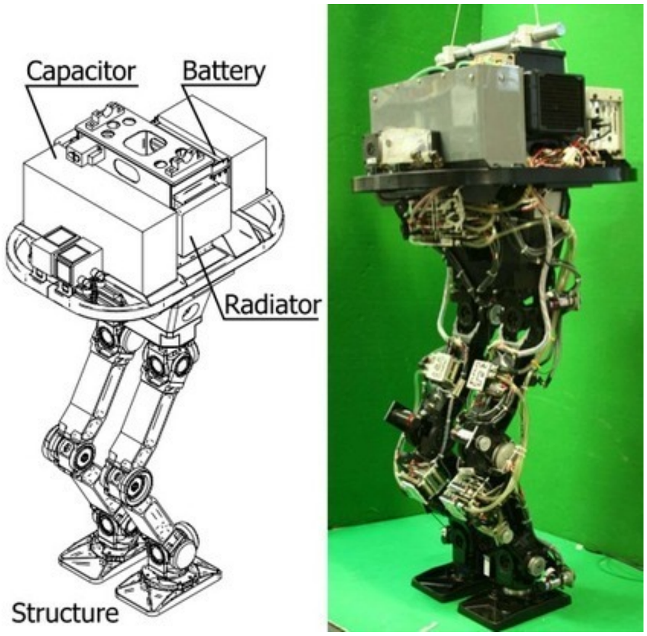


Figure 2.13. University of Tokyo robotic leg platform with 13.5 Farad battery-capacitor drive and liquid cooled motors on joints. (Image reproduced from Guizzo 2012.)

In this brute-force design, a 13.5-Farad capacitor and batteries provide enough energy to the liquid cooled electric motors (i.e., enough torque) to enable the robot to jump in the air. The legs were designed to be capable of a joint angular velocity up to 300 degrees per second.

Figure 2.14 shows the leg design of Toshiba's quadruped robot. Toshiba's design has two actuators on the hip and one on the knee, and no ankle joint. Three axis of motion is sufficient to position the foot in three dimensions. The actuators use electric motors and non-back-drivable harmonic drive (or wave type) gear reducers.

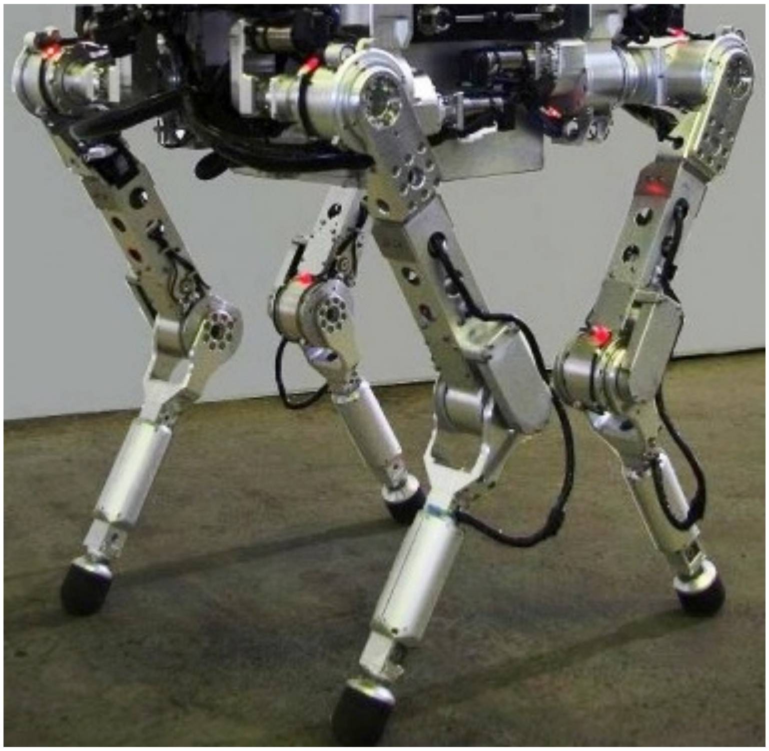


Figure 2.14. Toshiba quadruped leg design. (Image reproduced from InfoNIAC.)

Harmonic drives and other non-back-drivable gears (such as some types of worm gears and hypoid bevel gears) are advantageous for use with electric motor actuators, because a jointed leg is typically kept in a bent position at all times. As such, it requires a constant opposing torque and thus constant power consumption from the electric motor to support the weight of the body, at all times. To reduce power consumption in jointed leg mechanisms, non-back-drivable actuation is preferred. Non-back-drivable actuators only require power to position the leg. An electrical analogy is the difference in power consumed by CMOS versus TTL circuits. Furthermore, gear reducers are often advantageous when high gear reduction (high torque and force) and high accuracy when zero backlash is desired. Harmonic and worm gear reduction drives are commonly found in (typically slow moving) hexapod and quadruped robots, where three or more feet are supporting the robot at all times.

Hydraulics and pneumatics also provide the advantage of non-back-drivable actuation, and add compliance to shock and vibration handling, that can overload and damage the teeth of mechanical gearboxes. Figure 2.15 shows Boston Dynamics' Big Dog quadruped robot prototype hydraulic leg design (left) and the third generation hydraulic leg used in their Pet Man robot (right). Such servo-hydraulic leg designs use high-pressure hydraulics (up to 6,000 psi) combined with small-diameter hydraulic cylinders that are placed close to the leg joints, to provide mechanical advantage for very fast actuation of the foot. The Pet Man robot, for example, was designed to reproduce the force and joint angular speed of human limbs. Also note the skeletonizing of the leg frames to reduce weight.

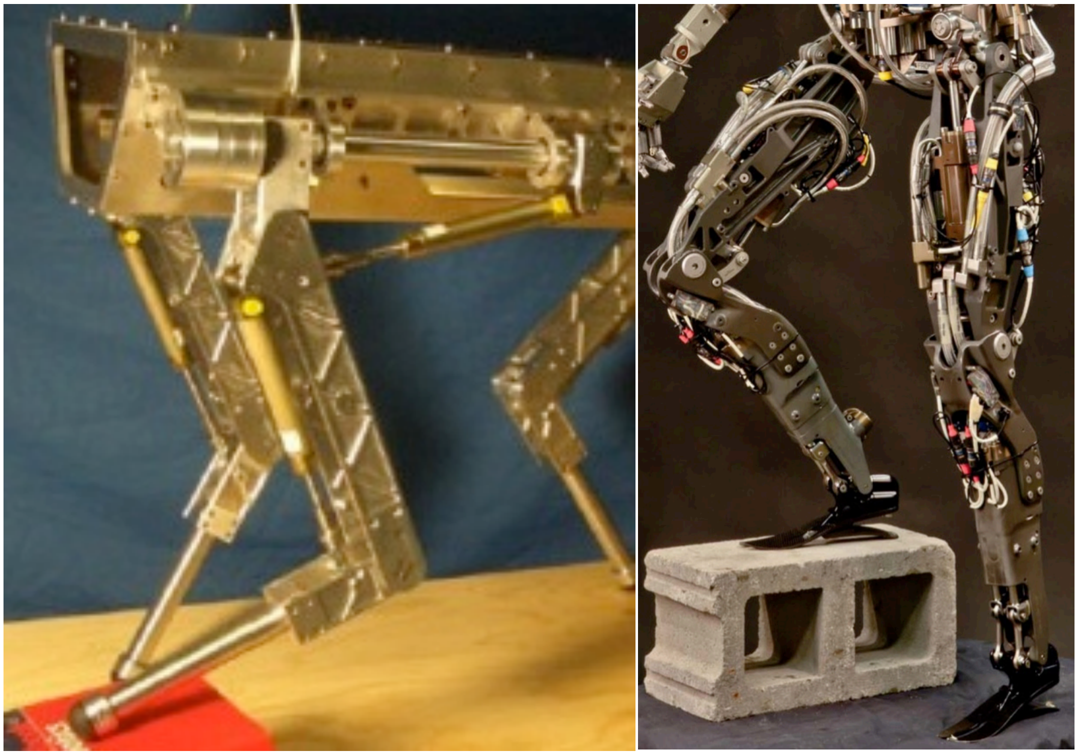


Figure 2.15. Boston Dynamics' Big Dog and Pet Man leg designs. (Images reproduced from Boston Dynamics web site.)

Unfortunately, high-pressure hydraulic systems are very expensive and are also very dangerous. Boston Dynamics received $65 million in government funding to develop the Big Dog robot. A small pinhole leak in a hydraulic hose or seal can create a jet of liquid capable of slicing through flesh. An alternative to hydraulics is pneumatics. Figure 2.16 shows a biologically inspired pneumatic leg design that achieves a high level of control by using nine actuators in a bi-articulate design. Whereas hydraulics uses an incompressible fluid, typically oil, pneumatics uses compressed air. Pneumatically actuated legs can achieve high actuation force and high joint angular speeds, but typically with lower position accuracy.

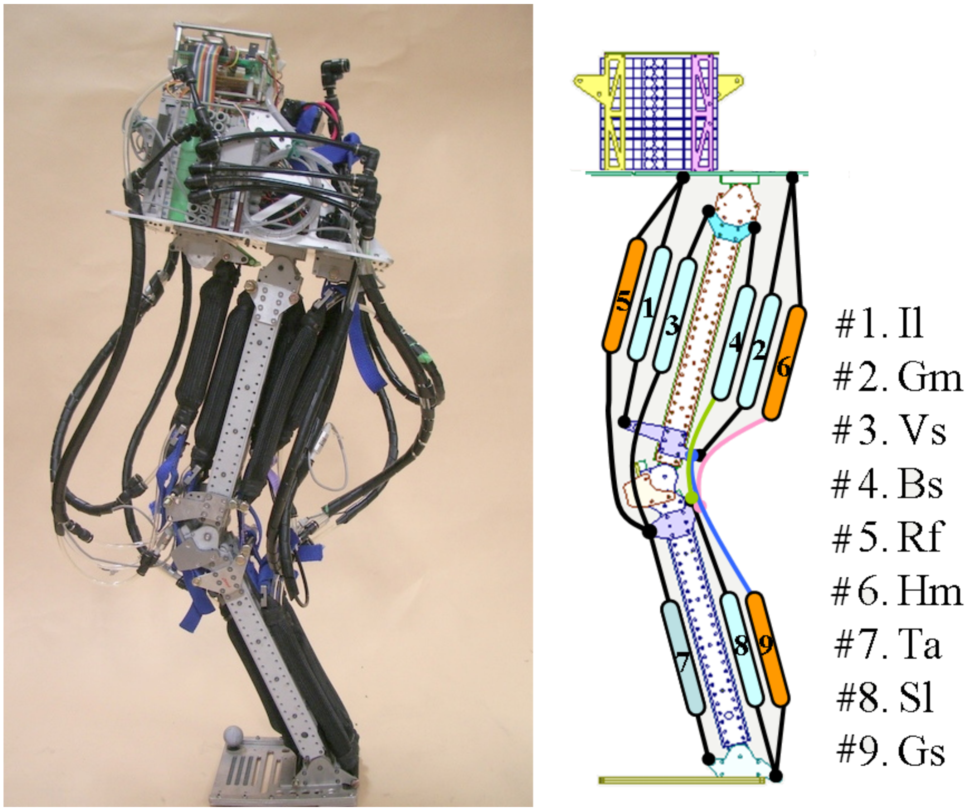


Figure 2.16. Pneumatic bi-articulate leg design. (Image reproduced from iopscience.iop.org.)

Similar to the biologically inspired leg of Figure 2.16, researchers at the University of Tokyo JSK Lab have developed a musculoskeletal-like electric actuator. A back-drivable design, it requires a second opposing actuator to elongate the actuator. Shown in Figure 2.17, the electric actuator uses high tensile strength wire looped through multiple moving pulleys and attached to a capstan. The capstan retracts the wire at high speeds, and the pulleys provide mechanical advantage as well as rotational compliance of the pulley assembly. The musculoskeletal actuator was designed for use in the Kenshiro robot, and it achieves almost the same amount of joint torque as a human, with joint angular speeds of 70-100 degrees per second (Osada 2011). Because the capstan-wire-pulley system is inelastic, a high degree of position accuracy is achieved, while the back-drivable motor provides compliance.

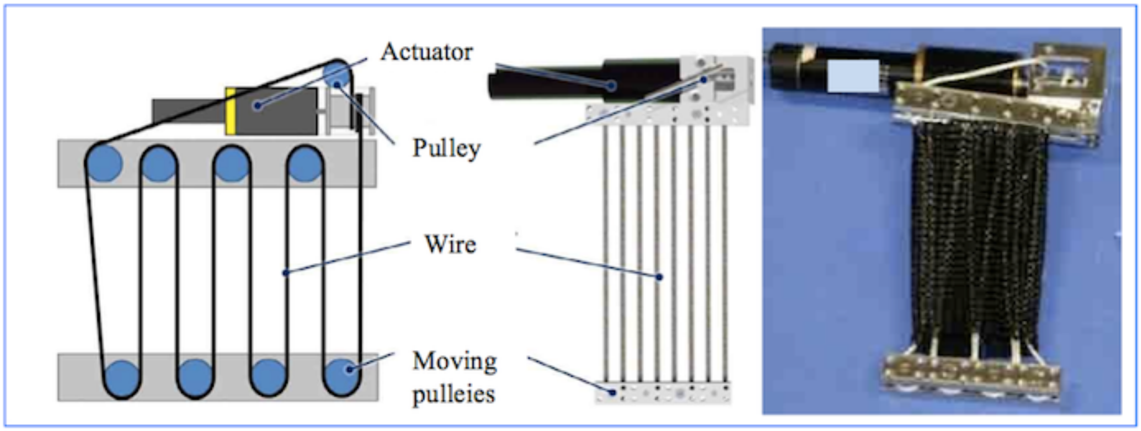


Figure 2.17. University of Tokyo's musculoskeletal actuator. (Image reproduced from JSK Lab web site.)

2.2.2. Coupled-Drive Leg Mechanisms

A lighter weight leg alternative to the actuator-on-joint leg is the coupled drive design, wherein all actuators are located at the Hip axis to minimize leg mass (Takita *et al.* 2000 and 2001). Figure 2.18 illustrates a schematic (left) and shows the coupled drive leg (right) of a biped robot. Note the upper linkages connect directly to RC servos in the image (right), denoted by rotation arcs in the schematic (left). The middle link (blue) is used to keep the foot horizontal, and may be ignored for this discussion.

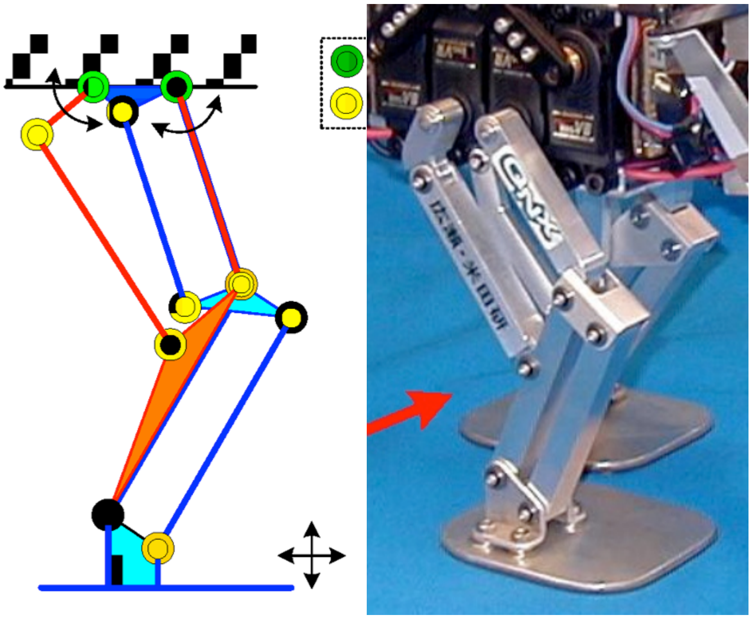


Figure 2.18. Coupled-drive leg design of the TITRUS-III dinosaur-like robot. (Image reproduced from Hirose Lab web site.)

Because the coupled drive adds an extra linkage (or lever arm) before the connecting rod to support the calf, the leg benefits in that the vertical forces are shared across both actuators. Similarly, both actuators contribute to develop reaction forces in the horizontal direction. When the leg is traveling rearward, for example, both actuators are torqueing (pulling) in a clockwise direction. This coupled-drive design achieves higher mechanical efficiency, compared with an equivalent actuator-on-joint leg, over a working range where both motors can contribute. Additionally, the leg can be tuned for mechanical efficiency or foot velocity for a given working range, by changing the linkage lengths. Longer linkages provide a longer lever arm and thus a higher foot velocity. A disadvantage (i.e., time and throughput) of this design is the complexity of calculating the inverse kinematics, over the simpler actuator-on-joint leg designs.

2.2.3. Prismatic Leg Mechanisms

A leg mechanism need not be constrained to biologically plausible designs. Figure 2.19 shows a hexapod robot with prismatic leg mechanisms. In this design, the robot leg is a simple tube captured inside a cylindrical, linear bearing (prismatic joint). The bearing constrains the motion to the vertical up and down motion of the foot. The vertical actuator is mounted to a wheeled trolley that rides within a box-type frame or track. The track constrains the motion of the trolley to the horizontal forward and backward motion of the foot. Curved motion is achieved by pivoting the front and rear tracks to the left or right with respect to the midsection.

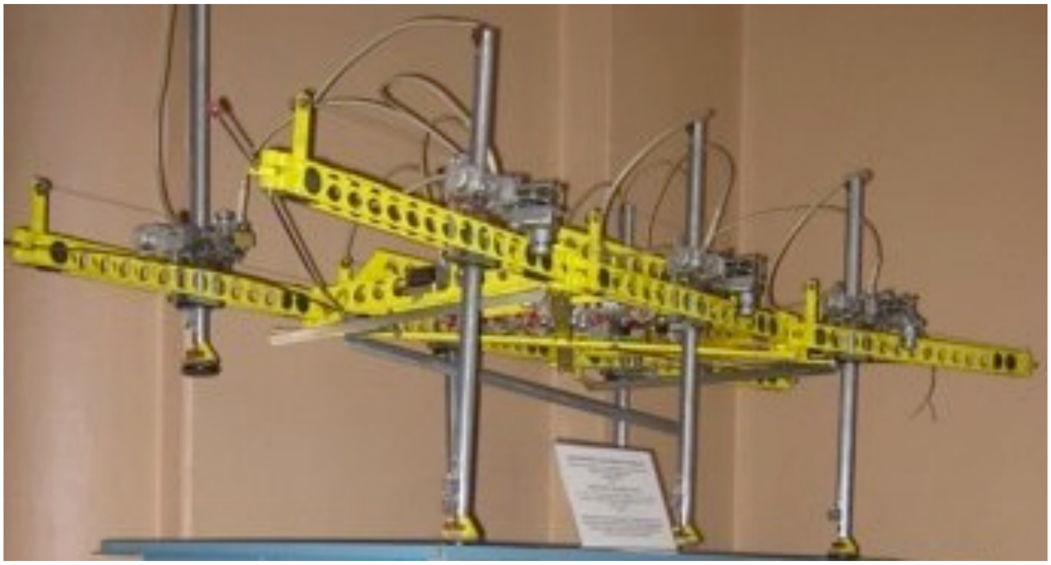


Figure 2.19. Hexapod robot with prismatic vertical and horizontal actuators. (Image reproduced from iopscience.iop.org.)

Using and controlling one actuator for leg length and a second for leg position in the x-axis (along the direction of travel, for example) greatly simplifies the control system, as compared to a jointed leg which involves computing the inverse kinematics of the leg members to determine the position of the foot from the joint or actuator angles.

Three or more prismatic actuators may be linked together at the distal end (Foot) in a Stewart-type configuration to provide three-dimensional positioning of the foot. For example, a team at Waseda University developed the biped robot shown in Figure 2.20. In this design, four prismatic or linear actuators are kinematically connected or coupled at the Hip and Foot using U-joints at each end.

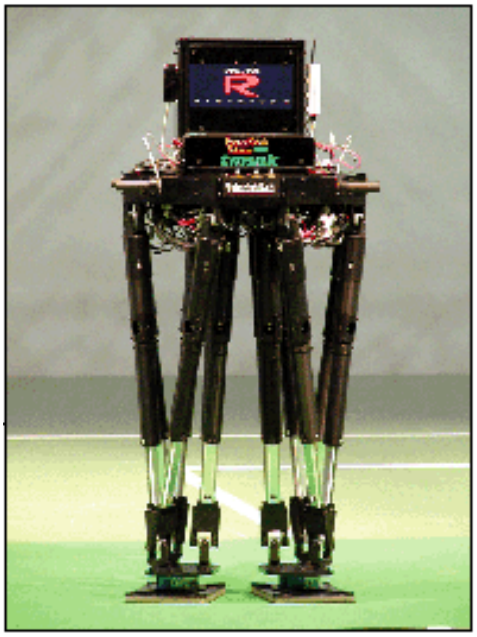


Figure 2.20. Biped robot with prismatic actuators in a Stewart configuration. (Image reproduced from Hashimoto et Al. 2008.)

A U-joint transmits force and torque across the joint, while allowing roll and pitch but not yaw motion. By simultaneously lengthening and/or shortening the coupled actuators, the foot can be positioned in three dimensions as well as rotating the ankle. Waseda University's robot uses high-speed electric ball-screw type actuators, but it does not have enough linear velocity to achieve running or jumping. Rather, it uses a Zero-Moment Positioning (ZMP) control strategy to keeps its center of mass within the support area of either foot at all times.

While the aforementioned electric-drive prismatic mechanisms are generally slow and heavy, prismatic actuators may also be high speed, high force and lightweight. Raibert's one-legged hopping robot, as described earlier and shown in Figure 2.7, uses a pneumatic cylinder to achieve direct vertical motion of the foot (Raibert 1986b). Because the accuracy requirements in the vertical axis are low, while high compliance on landing (shock absorption) and high travel rates for takeoff are desired, pneumatic actuator systems are ideal.

2.2.4. Pantograph Leg Mechanism

A pantograph mechanism adds a lever to a prismatic actuator to gain mechanical advantage, enabling the foot to travel faster than the actuator. In 1968, McGhee and associates at Ohio State University developed a computer-controlled, rideable hexapod robot that used pantograph legs to walk in unstructured outdoor terrain (Buckett 1977; Briggs 1977; and Pugh 1982). The left schematic of Figure 2.21 illustrates the pantograph mechanism overlaid on the mechanical drawing of the robot, and the right image shows the robot walking on level terrain.

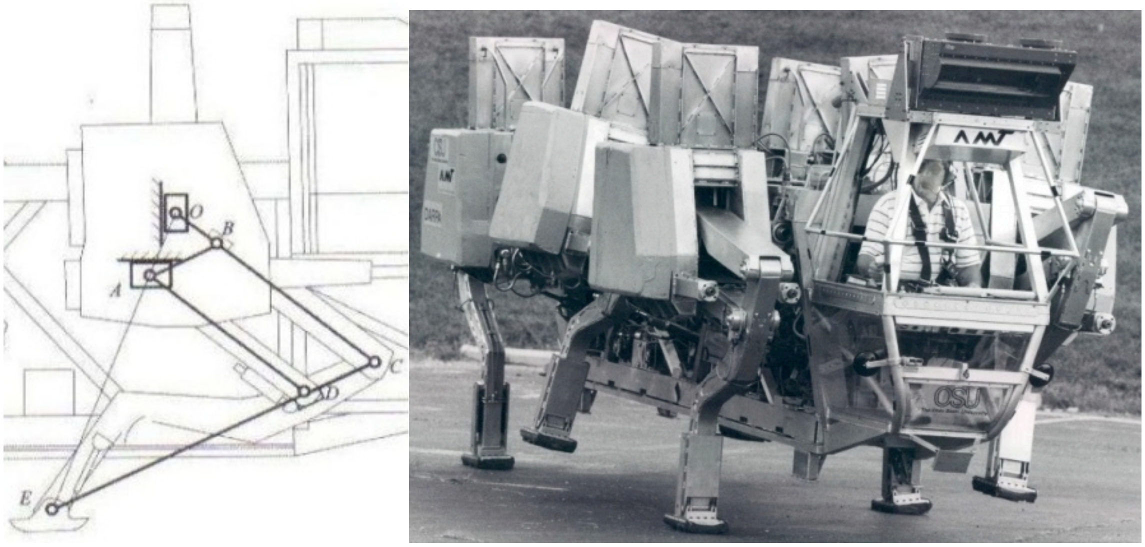


Figure 2.21. Ohio State University hexapod robot using pantograph legs. (Image reproduced from cyberneticzoo.com.)

The parallelogram or four-bar linkage (A-B-C-D) of the pantograph mechanism is constrained by two prismatic actuators, one mounted vertically (O) and the other mounted horizontally (A). A long lever (C-E) connects the four-bar linkage to the foot (E). The four-bar linkage translates linear motion of the actuators to proportional linear motion of the foot. That is, the vertical actuator (O) displaces the foot (E) proportionally in the vertical direction, and the horizontal actuator (A) moves the foot (E) proportionally in the horizontal direction.

Because the pantograph mechanism proportionally amplifies the position and velocity of the foot, the Ohio State University pantograph leg was designed with low pressure, low rate hydraulics for actuators (and a relatively slow computer control system by today's standards). The design trade-off, however, is that the pantograph's 4-bar linkage adds weight to achieve equivalent leg stiffness, as compared with simple prismatic or jointed legs.

Three-axis pantograph legs have also been developed for robots. Figure 2.22 shows the Titan III robot developed by Dr. Hirose at the University of Tokyo (Hirose *et al*. 2009). In this all-electric actuator design, a ball screw actuator vertically drives the upper bar of the parallelogram through a u-joint, and a compound actuator positions the lower bar in two dimensions horizontally.

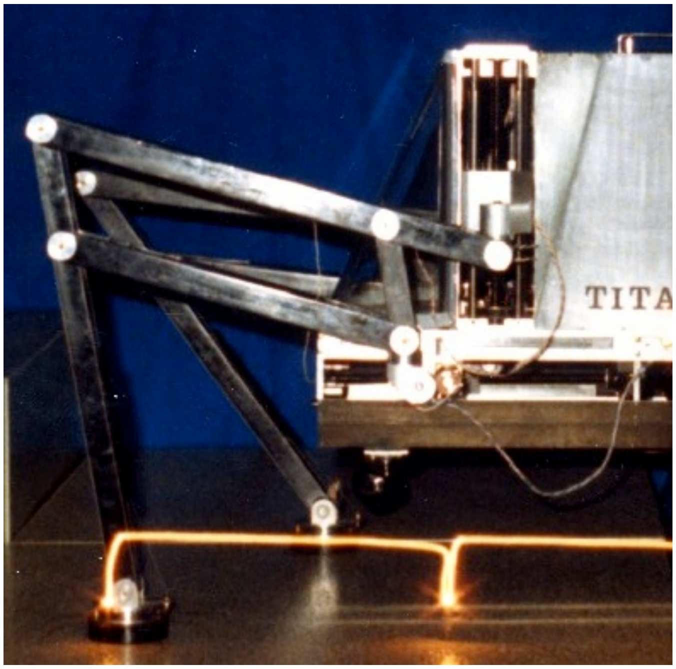


Figure 2.22. Titan III robot leg with three-axis pantograph legs. (Image reproduced from Hirose Lab web site.)

The robot shown here uses the same type of actuator (i.e., hydraulic or electric) for both vertical and horizontal actuation. Different combinations of actuators could be combined to achieve different performance or accuracy requirements in the horizontal and vertical axis. For example, a slower non-back-drivable electric actuator could be used for vertical actuation, while a high-rate pneumatic or linear electric motor could be used for horizontal actuation. Such an actuator combination would combine the best in energy efficiency (vertical axis) with compliance and high force and rate (horizontal axis) to enable running gaits.

In summary, a survey of the literature found numerous examples of four basic leg mechanisms: jointed, coupled, prismatic, and pantograph. There are literally hundreds of variations of the basic leg mechanism type. For example, the actuator-on-joint leg design may be rotated 90-degrees, and is called a Selective Compliance Articulated Robot Arm or SCARA. Other examples include snake-like designs, where a segmented body pushes against the ground in a serpentine fashion to produce motion. Nevertheless, the examples presented in this chapter provide a basic background for understanding the ST3LMR design. Chapter 3 discusses the ST3LMR design and operation in detail.

2.3. Mobility Control Methods

Research in legged robots generally falls into one of four open research topics: mobility, navigation, intelligence, and interaction (Kim *et al*. 2007). This section provides a literature review of legged robot mobility, and it includes a brief review of biologically-inspired methods to traditional analytic control methods. In addition, relevant literature on bicycles and motorcycles is presented for further background in single-track control methods. For example, Doyle outlines three fundamental control concepts for bicycles (Doyle 1987, 32; and Doyle 1988), which are also applicable to legged robots and especially to single-track legged robots:

“1) Don’t fall over;

2) Turn where and when you want to go; and

3) Avoid obstacles and go to desired places.”

Much of the reported work on legged robots prior to the mid-1980’s focused on statically stable locomotion (Krovi and Kumar 1996). Some notable examples of statically stable control systems include (Mosher 1969; Hirose 1984; and Waldron *et al*. 1984). As actuators and computers improved, legged robot research in dynamically stable control methods began. Most notably, Raibert’s work in unstable one-leg hopping machines (Raibert 1986b) provided the research community with a key proof-of-concept of a dynamically stable, running legged robot, but it would take another 20 years to refine, build, and demonstrate a full-scale quadruped robot operating in natural terrain, such as on mountain slopes and over piles of rubble, where wheeled or tracked vehicles cannot go (Raibert *et al*. 2008).

The general method of controlling legged motion involves generating a walking pattern, called a gait, and then adjusting the foot position and body attitude using sensory feedback to both provide locomotion and maintain stability of balance (Park *et al*. 2008). Again borrowing from research in single-track wheeled vehicles (i.e., bicycles), the following measurable parameters may be sensed and used as feedback by the control system (Ouden 2011): rotation, angular velocity, and angular acceleration in the roll and yaw directions; and velocity and acceleration/deceleration in the forward/backward and lateral (e.g., centripetal acceleration during cornering) directions. Where the forward/backward direction (typically sagittal) corresponds to the major direction of motion (of the ST3LMR), the roll axis is parallel to forward/backward direction, and the yaw axis is perpendicular to forward/backward direction in the vertical direction.

2.3.1. Biologically-Inspired Methods

Early work in generating classical hexapod, quadruped, and biped walking gaits (either mechanically through gears and cams or electronically) was biologically inspired from real-world examples, such as from animal (Muybridge 1899; and Muybridge 1901) and insect (Wilson 1966) studies. In insects, (Wilson 1966) observed the metachronal gait (low walking speed) and the alternating tripod gait (high walking speed). Quadruped gaits described in the literature include the amble, canter, creep, gallop, lateral, pace, passage, pronk, rack (or racking lateral), revaal (or ravaal), step, tölt, trot, and walk (Harris 1993; Kar *et al*. 2003; and Thornton 2004). Interestingly, there is considerable variety in these basic gaits, including for example, the trot which include the collected, extended, fox, jog, lengthened, medium, park, racing, slow, and working types. Notable research in bipedal gaits are presented in (Chow and Jacobson 1971; and Hatze 1976), trajectories in (Alexander 1990; Vukobratovic *et al*. 1990; and Delcomyn 1999), and the optimization of cost criteria (Cabodevilla *et al*. 1996).

The use of a central pattern generator or CPG to drive the legs through sensory feedback is found in (Katoh and Mori 1984; Lewis *et al*. 1994; Kimura and Fukuoka 2000; Kuo 2002; Lewis 2002; Lewis and Bekey 2002; and Lewis *et al*. 2005). Neural networks have also been used to control legged robots (Hu *et al*. 1998; and Manoonpong 2007). A complete description of biologically-inspired control methods is beyond the scope of this dissertation, and the reader is referred to the cited literature for more information.

2.3.2. Walking Versus Running

An important concept in legged systems, regardless of the number of legs, is the difference between walking and running. The common distinction between walking and running is based on walking having at least one foot on the ground at all times whereas running has a ballistic flight phase (with all feet in the air at some point in the gait cycle). Note that there are exceptions to this definition, such as in (McMahon 1985). To this end, (Cavagna *et al*. 1976; Alexander 1990; and McMahon and Cheng 1990) provide additional definition. Figure 2.23 illustrates a flow chart of legged locomotion, adapted from (Ringrose 1996), showing three stability types, ranging from static (no motion) to quasi-static (minimal motion/momentum) to dynamic (reliance on motion/momentum).

From a stability point of view, both walking and running entail both quasi-static and dynamic stability control methods (Ringrose 1996). However, this dissertation is concerned only with walking gaits for stabilization. Note also that for the proof-of-concept, the ST3LMR is shown to be capable of maintaining static stability using a tripedal stance (e.g., three legs arranged in a tripod fashion) and is self-stabilized through the action of the supporting legs/feet. These two extremes are highlighted as green boxes in Figure 2.23.

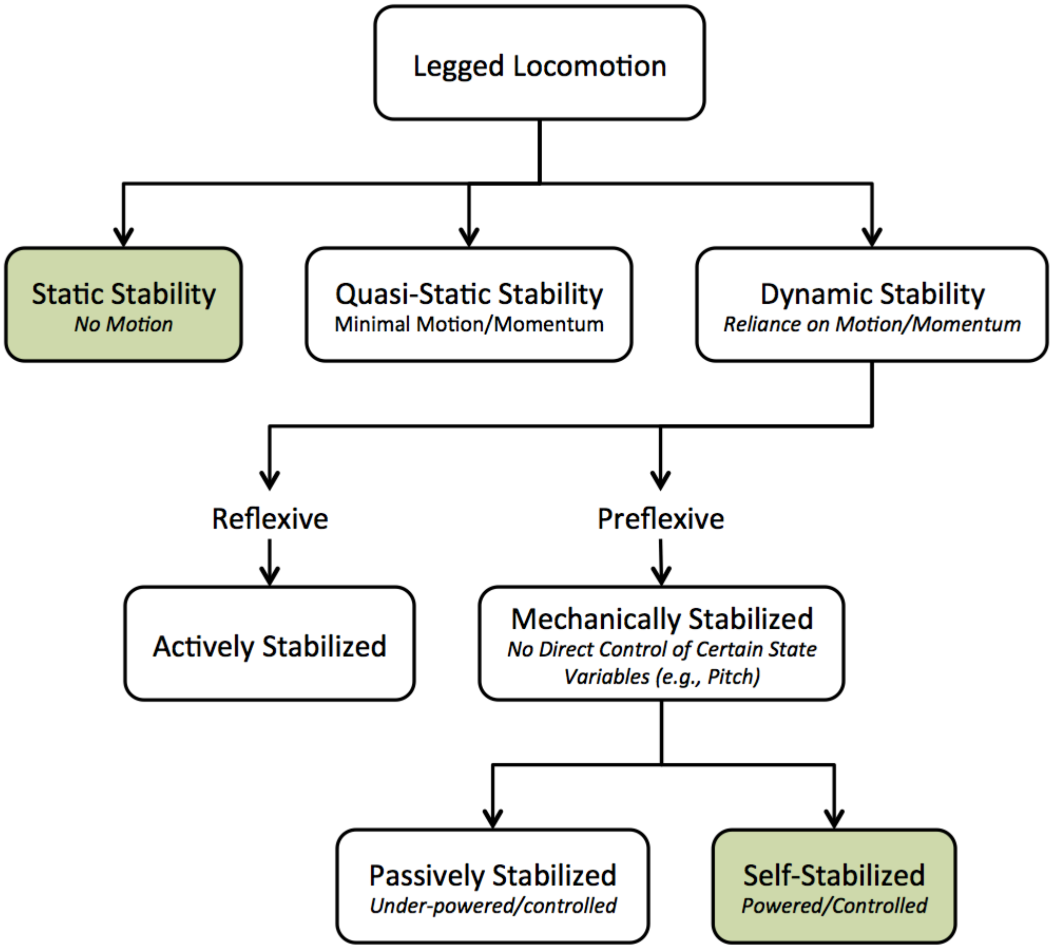


Figure 2.23. Legged locomotion stability type flow chart. (Adapted from Ringrose, 1996.)

2.3.3. Static and Quasi-Static Stability

For robots with a large foot, two large feet, or three or more supporting feet, as long as the center of pressure, called the zero moment point or ZMP (Vukobratovic and Stepanenko 1972; and Vukobratovic and Borovac 2004), remains inside the convex hull of the supporting feet, the robot will not fall over (Hirai *et al*. 1998; Ito *et al*. 2000; Park and Cho 2000; Haung *et al*. 2001; Pfeiffer *et al*. 2002; Sugihara *et al*. 2002; and Smith

2006). In other words, the torque generated about the ZMP of the robot body, by both inertial and gravitational forces is zero, and the robot does not fall over.

ZMP-type control methods typically use two controllers. A high-level controller develops a body trajectory motion whereby the ZMP remains within the convex hull of the feet, and a low-level controller tracks the ZMP motion in real-time. Quasi-static stability is achieved in robots exhibiting periodic motion by adjusting the acceleration of the robot along its trajectory (Chevallereau *et al*. 2008). (For an example of periodic motion, the reader is referred to Chapter 1, Figures 1.3 and 1.4 for an illustration of the ST3LMR implementing a follow-the-leader BWG.)

ZMP control methods are further characterized by short leg flight times and low foot ground clearance during flight. ZMP control is used on such notable robots as Sony's QRIO, Honda's ASIMO, Toyota's humanoid robot, HRP-2LR, HRP-2LT, HUBO, and TITAN-VIII (Kurazume 2001; and Sreenath *et al*. 2012). In practice, ZMP algorithms are typically implemented on multiple controllers. For example, one ZMP method involves the use of four computers: a ZMP regulation controller, an orientation correction controller, a compliance controller and an ankle torque difference controller (Kim et al. 2007).

2.3.4. Dynamic Stability

For robots that do not rely on large feet (e.g., point feet) and are characterized as 'fast walking,' such as the bipedal robots RABBIT and MABEL (Westervelt *et al*. 2004; and Park *et al*. 2011), dynamic stability is used to induce stable walking. Dynamic stability is achieved as the rate of foot placement and thus leg support increases proportionally with increasing speed. As forward speed increases to infinity, for example, foot placement approximates continuous support with respect to the force of gravity acting on the robot. There are two general methods used for dynamic stability. The first method involves reacting to and controlling one or more sensed parameters, and the second method involves pre-computing a reference trajectory and adjusting leg lengths and foot forces and torques to maintain the reference trajectory.

Examples of feedback-based methods include the regulation of angular momentum (Sano and Furusho 1990), controlling the angular position of the leg joints (Raibert 1986a; Sano and Furusho 1990; Goswani *et al*. 1996; Simmons 1996; and Pratt and Pratt 1998), heuristic walking principles (e.g., maintaining torso and hip height while walking) have also been researched in (Pratt *et al*. 1998 and 2001), and the use of fuzzy logic methods in (Marhefka and Orin 2000). Note that dynamic models have also been developed for bicycles and motorcycles (Levandowski *et* al. 2006; Limebeer and Sharp 2006; and Yi *et al*. 2006). In practice, a real-time controller is used to follow/correct the sensed torque to the desired torque, as in (Raibert *et al.* 1993; Chaillet *et al.* 1994; and Mitobe *et al.* 1995). A good discussion of parameter identification for controlling legged robots is found in (Park *et al*. 2011).

Trajectory-based methods appear more often in the literature, and various notable methods include: using reference trajectory prescriptions (Katoh 1984; Kajita and Tani 1996; Hirai *et al.* 1998; Raibert *et al.* 1993; Chevallereau *et al*. 1997; Saranli *et al*. 1998; Nakanishi *et al*. 2000; and Park *et al*. 2008), using ZMP dynamics models described by ordinary differential equations (Hurmuzlu and Marghitu 1994; Isidori 1995; Spong 1995; Berkemeier and Fearing 1999; Ito 2000; and Rosas-Flores 2000), using hybrid systems of differential equations to model the single support and swing phases and a discrete leg impact map that linearize the robot motion along a periodic orbit (Grizzle *et al*. 2001; Plestan *et al*. 2003; Westervelt *et al*. 2004; and Poulakakis 2009), regulating total energy (Goswami *et al*. 1996), using time-based ZMP trajectories (Hasegawa *et al*. 2000) or system state-based (i.e., non-time based) methods (Ono *et al*. 2000), and using virtual constraints (Fukuda *et al.* 2006). Note that trajectory tracking and balance stabilization of motorcycles was accomplished using Lagrange's equations (Sharp 1971), and rider-based multi-body dynamics models have been demonstrated (Cossalter and Lot 2002; and Kessler 2004).

Applications of Poincaré's method to trajectory estimation, for example, have been applied to Raibert's one-legged hopper (Koditschek and Buhler 1991; Canudas *et al.* 1997; and Francois and Samson 1998) and torso-less bipedal robots (Thuilot *et al.* 1997; Goswani *et al.* 1996; Krovi and Kumar 1996; and Smith and Berkemeier 1998). Grizzle, Plestan, and Abba generalized the method of Poincaré sections, laying the groundwork for a class of dimension-invariant hybrid control systems for bipedal locomotion (Grizzle *et al.* 1999). In their work, the side-to-side sway of a bipedal robot in a regular walk is viewed as a periodic orbit with an impulse occurring at each step. (For comparison, the reader is referred to Chapter 1, Figure 1.5 for an illustration of the periodic torque developed throughout the ST3LMR walking cycle, given a left-right oscillating front foot placement in a follow-the-leader BWG.)

For real-time control, an event-based PI controller is used to regulate velocity to induce an exponentially stable, periodic orbit of the body/ZMP. The integrator is adjusted based on a restricted Poincaré map of the hybrid zero dynamics, just after a foot touchdown/impulse. PID controllers have also been used to follow/correct the sensed trajectory to the desired trajectory, as in (Furusho and Masubuchi 1986; and Furusho and Sano 1990). Additionally, discrete time methods based on impulse control is found in (Chevallereau *et al*. 1997 and 2008).

Other trajectory-based methods include modeling the robot as an inverted pendulum and regulating its center of mass (Kajita and Tani 1996). It is this latter method that is followed in this research. Specifically, the ST3LMR shares many single-track properties with bicycles and motorcycles, and simple inverted pendulum methods have been successfully used for bicycle balancing (Tanaka and Murakami 2004). In this research, the Monte Carlo method is used to investigate all possible foot placement options for the ST3LMR modeled as an inverted pendulum.

Generally speaking, Monte Carlo computer simulations were originally (in the late 1940's and through the early 1970's) reserved for only a few very challenging tasks that defied conventional analysis, such as modeling nuclear processes, because the cost of running such computer simulations on big mainframe computers was very expensive. Today's computers enable high-speed simulation at a low cost. Later works, such as the A* and D* methods (Mitchell and Keirsey 1984; and Stentz 1994), hierarchical control methods (Brooks 1985; Benjamin 2002; and Benjamin, 2004), and genetic and evolutionary path planning algorithms (Lewis *et al*. 1994; and Rathbun and Capozzi

2009), to name but a few, provide foundations to consider non-analytic approaches for foot placement planning.

A simple inverted pendulum simulation is developed to demonstrate the proof-of-concept and validate the experimental results. Chapters 3 and 4 discuss the mechanics of the ST3LMR concept, and Chapters 5 through 7 present and discuss the details of the control method. Appendix A presents example MATLAB code for the ST3LMR digital simulation, and Appendices B and C discuss the experimental results.

CHAPTER 3

SINGLE-TRACK THREE LEGGED MOBILE ROBOT DESIGN

This chapter provides a detailed discussion of the Single-Track Three Legged Mobile Robot (ST3LMR), shown in Figure 3.1, from US patent 8,457,830. The first section of this chapter provides a technical description of the robot design. The second section provides a detailed description of the ST3LMR, and it includes a discussion of how the three legs are configured. The third section details the method of ST3LMR operation, and it includes a discussion of how the three legs develop legged motion.

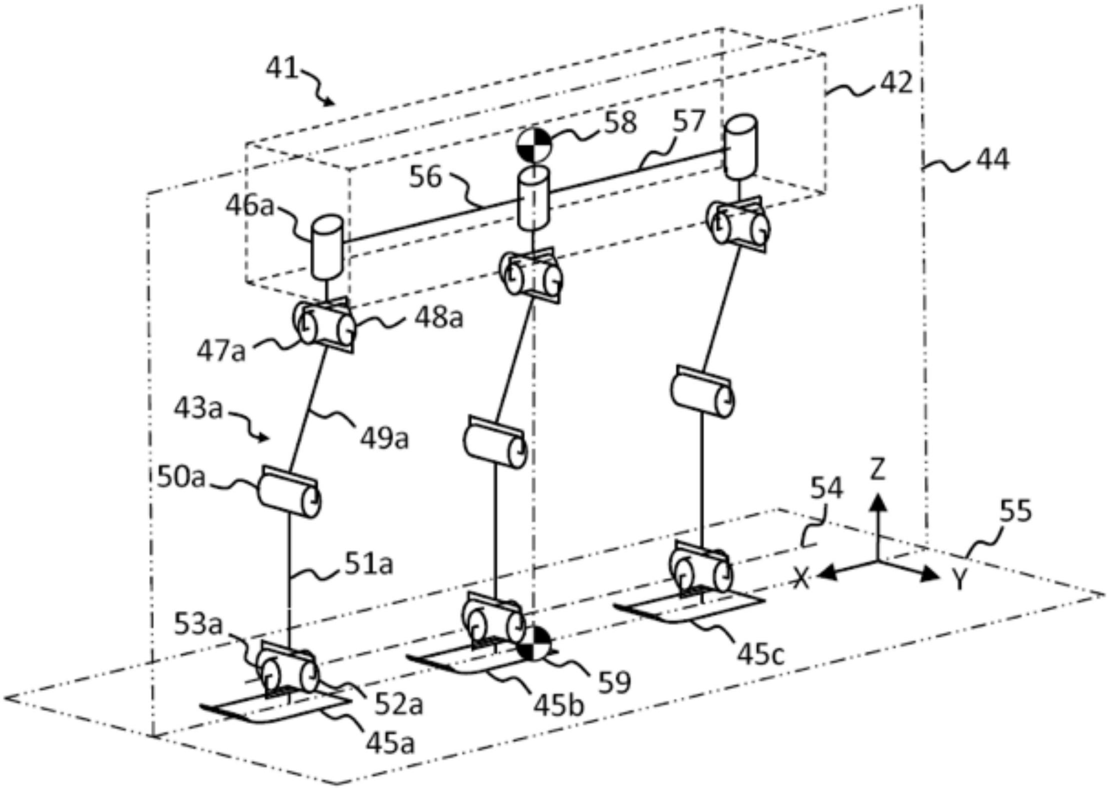


Figure 3.1. Skeletal perspective view of the single track three-legged mobile robot or ST3LMR. (From US patent 8,457,830 Fig 1.)

The legged vehicle shown in Figure 3.1 and discussed in this .dissertation includes three legs. However, an in-line or single-track legged vehicle is not restricted to three legs. The term ‘single track’ refers to the general narrowness of the foot-placement patterns along a straight or curved path. Other single-track designs having two, four or more legs are possible. The three-leg design achieves the in-line or single-track concept, minimizes cost (relative to four or more legs), and it is the focus of this design and this chapter.

3.1. ST3LMR Design

The ST3LMR design comprises a body and three legs mounted on the body in-line with the length of the body. The three legs thereof comprise a minimally narrow profile so that as a vehicle it can maneuver where prior art vehicles previously could not go, such as walking along a narrow trail or path or through a door. Each leg is connected to a single or multi-segmented body that is generally longer than it is wide. The body length establishes the major direction of motion, such as forward and backward motion. Each of the three legs are spatially arranged at the hip to be generally in-line with the major direction of motion. Each of the three legs has at least three degrees of freedom (DOF), such as pitch and roll at the hip and extension and retraction of the foot, to position the foot anywhere within a three dimensional volume. The three legs combine to form three spatial volumes for foot placement that is spatially arranged to be generally in-line with the major direction of motion. The three spatial volumes overlap along the major direction of motion. The three legs have sufficient reach in length, width, and height to afford the three feet to be spatially positioned 1) in a triangular pattern to keep the body in static equilibrium at rest, 2) in any manner of patterns to provide locomotion and dynamic attitude stabilization, and 3) for omnidirectional motion.

An arrangement of three in-line legs provides an advantage of inherent stability along the pitch axis. Pitch is rotation about the y-axis (Figure 3.1), roll is rotation about the x-axis, and yaw is rotation about the z-axis. The x-axis is generally parallel with the major axis of the body and major direction of travel. In other words, inherent pitch stability occurs naturally from having multiple support legs in the x-axis. Like the Drasine or motorcycle, having all of the support along the x-axis requires maintaining stability in a roll axis (i.e., side-to-side or perpendicular to the pitch axis along the ground plane). An arrangement of three legs also provides multiple options for traversing unsuitable terrain in regard to possible footholds and foot placement timing options (e.g., not immediately placing the foot down) by relying on dynamic stability (for a brief period of time) and the inherent pitch stability/control provided by two supporting legs.

ST3LMR may also include a frame that is jointed to include two or more segments, each segment having a major axis corresponding to and generally parallel to a forward/backward direction of travel. The frame would also include a major axis corresponding to and generally parallel to a forward/backward direction of travel. A plurality of jointed leg mechanisms would also attach to a segmented frame, one behind the other, wherein each leg is attached at its proximal end at one or more discrete attachment points. The attachment points are arranged substantially parallel to the major axis of the frame and the forward/backward direction of travel. Each of the legs includes actuators attached between the legs and the frame and between adjacent leg members. The legs are actuated for movement of the distal end in three dimensions.

An articulated frame would provide further advantage in motion flexibility, which would extend the ranges of motion of each of the legs, particularly the front-most and rear-most legs. With proper coordination of articulated frame segments, a much faster, more natural gait can be used to quickly traverse even the most challenging terrain. An articulated frame would provide numerous advantages in mobility, including the ability to travel through narrow passages, such as doorways, and along narrow paths, such as single-track trails, where traditional vehicles would be unable to go. Regardless, the single-piece frame is discussed in this chapter and is used in the simulations and prototypes. A single-piece frame simplifies the gaits and any necessary programming to traversing terrain.

Regarding the foot at the distal end of the legs, each of the feet may include at least one of, or one or more of, plates, skids, spikes, wheels, skates, skies, slides, floats, hydroplanes, and fingers. Different combinations of the different foot-types may be used to accommodate different types of terrain. Accordingly, different gaits may be used according to the combination of foot-type and terrain. These diverse feet types bring tremendous flexibility to the vehicle.

The movement range for each of the legs defines a working envelope, each of the feet having sufficient reach and movement range in length, width and height, relative to the frame, (1) to position two feet perpendicular to the major axis of the frame, with one foot positioned to the left of and one to the right of the projected center of gravity of the frame to form a generally bipedal stance along the major axis of the frame to provide a degree of stability in the roll axis, and (2) in addition to the placement of the first two feet, to displace a third foot into a position parallel to the major axis of the frame, either to the front or the rear with respect to the other feet, to form a generally tripedal stance about the projected center of gravity (center of pressure) of the frame to provide a period of stability in both the pitch and roll axes.

The movement range for each of the legs provides range-of-motion overlap in length, width and height of the working envelopes of each adjacent foot, including any foot in front of and behind each foot. This provides a tremendous amount of flexibility in achieving temporary stability while in motion and when utilizing dynamic stability, and permits a great range of possible leg positions, which are necessary when traversing difficult terrain.

The feet of two adjacent legs, one in front of and one behind the other, are positionable side-by-side on a center of pressure line generally perpendicular to the frame and perpendicular to the major axis of motion to achieve a bipedal stance. This ability permits the frame to be positioned so as to bring a zero-moment line of the legged vehicle in coincidence with the center of pressure line, which allows the other legs of the legged vehicle to be raised off the ground, at least temporarily.

The legs include at least three degrees of freedom (DOF). The three degrees of freedom may be defined by pitch and roll movement at a hip joint and may include extension and retraction of the leg by knee and ankle joints which define a spatial volume for possible leg placement. The movement range for each of the legs defines a working envelope, each of the feet having sufficient reach and movement range in length, width and height, relative to the body, to be placed in a plurality of predetermined locomotion and dynamic attitude stabilization patterns.

The movement range for each of the legs defines a working envelope, each of the feet having sufficient reach and movement range in length, width and height, relative to the body, to be placed in at least one omni-directional locomotion pattern. Motion along the major axis of the frame, is only one of the possible directions of travel. The movement ranges for the legs, and the actuator/control system interface permit the legged vehicle to move in any direction along the ground. Motion normal to the major axis may be in a side-step pattern, which will be described in detail in the following.

3.1.1. Detailed Description

Referring again to Figure 3.1, there is shown a ST3LMR, generally designated 41, including a body, generally designated 42, and three identical leg mechanisms, generally designated 43. The construction of each leg mechanism is not directly relevant to this dissertation, rather the dissertation is directed to the method and manner in which the leg mechanisms may be combined and attached to the body for forming a complete legged mobile robot and control thereof. However, leg mechanisms will be described briefly, because the teachings of which are necessary for an understanding of the present invention. Since each leg mechanism 43a, 43b, and 43c is identical, a description of one will suffice to describe all.

By convention, the left side of the page is the front side and forward direction, the right side of the page is the rear side, into the page is the right side, and out of the page is the left side of the robot. The x-axis is parallel with the length of the robot, with positive-increasing distance in the forward direction. The y-axis is perpendicular to the length of the robot (and x-axis) in the horizontal (in/out of the page), with positive-increasing distance in the left direction (out of the page). The z-axis is perpendicular to the length of the robot (and x-axis) in the vertical (bottom/top of the page), with positive-increasing distance in the upward direction (top of the page). Roll is rotation about the x-axis. Pitch is rotation about the y-axis. Yaw is rotation about the z-axis. Positive roll, pitch, and yaw follow the right-hand rule convention.

3.1.2. Leg Mechanism

It will be readily apparent by those skilled in the art, from an inspection of the drawings that a lightweight leg is the preferred embodiment. A lightweight leg mechanism uses one or more links to transmit mechanical force from actuators mounted in the body to the legs. Placing the actuators in the body reduce the weight of the legs, generally without increasing the complexity of actuation, without decreasing reliability, and without increasing computational requirements. Mechanical forces applied at selected points on individual parts of the links can be transmitted to another link, which forms the movable foot of the mechanism. However, the popularity of actuator-on-joints, found in prior art and in commercial off-the-shelf robotic kits, shall improve the clarity of understanding. Figure 3.2 illustrates the major components of the proof-of-concept

ST3LMR robot and middle leg mechanism, which is discussed in detail in Chapter 8. A perspective view of the prototype robot is also visible in the lower left corner insert. The individual legs do not have yaw-axis actuators. The robot is constructed from off-the-shelf parts from a Bioloid robotic kit manufactured by Robotis. (See www.robotis.com for more information.)

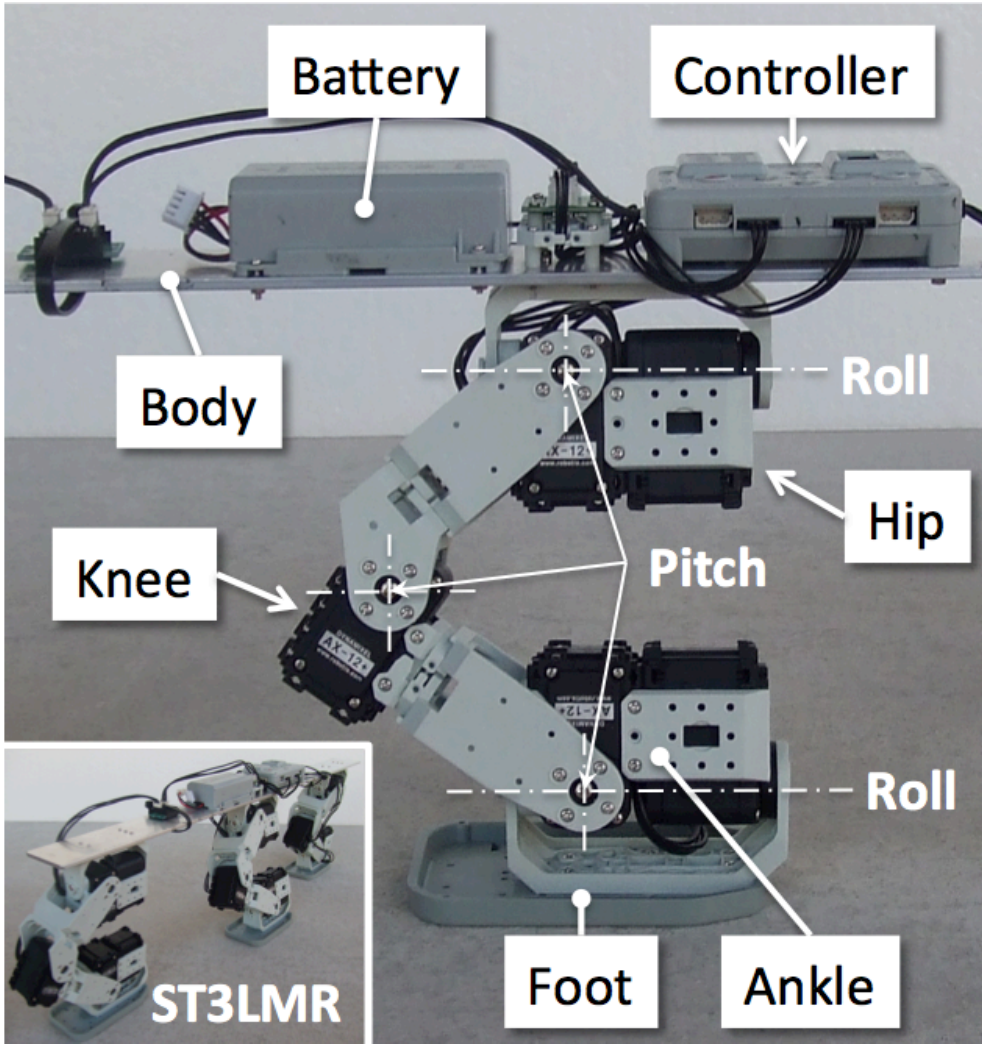


Figure 3.2. Major components of the middle leg mechanism from the ST3LMR proof-of-concept robot, with lower left perspective view insert.

Referring again to the skeletal view of the ST3LMR shown in Figure 3.1, the articulated structure of the legged mobile robot includes three legs mounted in-line or co-planar along the length of the body such that the three legs establish a plane 44 in the X-Z axis. Each leg mechanism is associated with six articulations or joints (axes) to enable each foot 45a, 45b, and 45c to be positioned in six dimensions (X-Y-Z, roll-pitch-yaw axis) with respect to the body 42. Since each leg mechanism shown in Figure 3.1 is identical, a description of the first leg will suffice to describe all the remaining legs which are unlabeled for clarity.

The joints (axes) of the leg include a yaw rotational joint (yaw axis) 46a for turning the leg and foot with respect to the body, a roll rotational joint (roll axis) 47a for moving the foot to the side (Y axis) of the body, a pitch rotational joint (pitch axis) 48a on a thigh link 49a for moving the foot forward and backward (X axis), a rotational joint (axis) 50a in the knee and at the distal end of the thigh link and on a shank link 51a for moving the foot forward and backward (X axis), a rotational joint (axis) 52a at the distal end of the shank link for moving the foot in the pitch direction, and a rotational joint (axis) 53a for moving the foot in the roll direction. The rotational joint axes are parallel to the pitch and roll axes, respectively. The foot is mounted to a small shank (not labeled for clarity) connected to a rotational joint on the lower end of the leg. The yaw, roll, and pitch rotational joints connected to the body constitute a hip joint assembly, and the pitch and roll joints at the distal end of the shank constitute a foot ankle joint assembly. The foot is moved forward or backward with respect to and kept parallel to the length of the body or parallel to the X-Z-axis plane by rotating the hip pitch, knee, and ankle pitch joints. The foot is moved to the right or left side with respect to and kept parallel to the length of the body or perpendicular to the X-Z-axis plane by rotating the hip roll and ankle roll joints. The roll axis 54 of the mobile robot is about the ankle roll joints. Note that for legs with a point-contact foot (i.e., without ankle joints), the roll axis is at the point of contact of the feet and the ground, for the condition where there is negligible slip between the foot and ground.

Simple vertical retraction or extension of the foot is accomplished by folding the leg at the knee joint, with the hip joint and the ankle joint (if the foot angle is to be maintained) simultaneously rotating in opposite directions. Simple lateral or side-stepping movement of the foot is accomplished by leg extension and rotation of the (outward radial) hip joint, wherein the leg mechanism is mounted for rotation with respect to the body. It will be readily apparent by those skilled in the art, from an inspection of the drawings that it is desirable for the body, legs, and/or feet to be mechanically compliant to comprise a spring-mass-damper system to afford a gentler ride of body.

Still referring to Figure 3.1, the body includes three main support plates to mount the leg frames to the body to afford hip rotation along the axis of motion such that the foot may be positioned laterally or radially outward with respect to the body and measured thusly. On each of the thigh and the ankle of each leg, the pitch joints and roll joints are disposed perpendicularly to each other, and have respective axes intersecting with each other at one point. The hip joint assembly, the knee joint, and the foot joint assemblies, respectively, extend parallel to each other in the X-Z-axis plane. Irrespective of movements caused by other degrees of freedom, particularly, movements of the hip yaw joint to change the direction of the legs, the hip, knee, and ankle joints remain parallel to each other. In the hip joint assemblies, the yaw joint and the pitch and roll joints extend perpendicularly to each other, so that the three axes of rotation, representing three degrees of freedom, extend perpendicularly to each other. More specifically, the yaw axis may be considered to define a first axis of the hip joint assembly, the roll axis defines a second axis of the hip joint assembly, and the pitch axis defines a third axis of the hip joint assembly. The yaw, roll, and pitch axes each provide respective degrees of freedom about which the leg of the robot may be moved, for example, the hip pitch joint provides a first degree of freedom for angularly moving the leg forward in the pitch direction, the hip roll joint provides a second degree of freedom for moving the leg laterally in the roll direction, and the hip yaw joint provides a third degree of freedom in the yaw direction for rotating the leg with respect to the body. It should be understood, however, that the designations "first," "second" and "third" are arbitrary, and are used merely to facilitate a description of the ST3LMR. Each leg thus has six degrees of freedom, so that during locomotion the legs as a whole can be caused to execute the desired motion by driving the 6×3=18 joints (axes) to an appropriate angle. Irrespective of the position or posture of the body, the feet can be placed in any position, at any angle, and in any direction. The robot is thus capable of walking freely within three-dimensional space. The joint actuations may be provided by any means such as high-pressure servo hydraulic direct drive or electric motors with reduction gear mechanisms (e.g., ball screws, harmonic drives, planetary gears, and so on) for increasing motor torque.

Still referring to Figure 3.1, the body has physical mass and thus a center of gravity 58 and its projection to the ground 55, called the center of pressure 59. The center of gravity and center of pressure are well known physical properties, especially with respect to single-track vehicles, such as motorcycles and bicycles, legged robots, and the like (Song and Waldron 1989). These concepts are developed further in Chapter 4.

3.1.3. Leg Configurations

Referring now to Figures 3.3 through 3.7, from US patent 8,457,830, several views of possible configurations of the legs and the method in which operation of the individual legs move are illustrated, whereby foot placement along the length of the body and in the major direction of motion shall now be discussed. According to the preferred embodiment, each leg can be rotated about hip, knee, and ankle joints in three dimensions. Figure 3.3 illustrates the side skeletal view of one possible configuration of the legs, wherein the x and z axes with the side or y axis into and out of the page. By convention, the left side is the forward direction. The first leg is shown with its foot extended forward with respect to its hip joints. Hereinafter, the terms forward or rearward direction shall be with respect to the hip joints along the length of the body. The second or middle foot is shown in a middle or neutral position, and the third foot is shown in a rearward extended position.

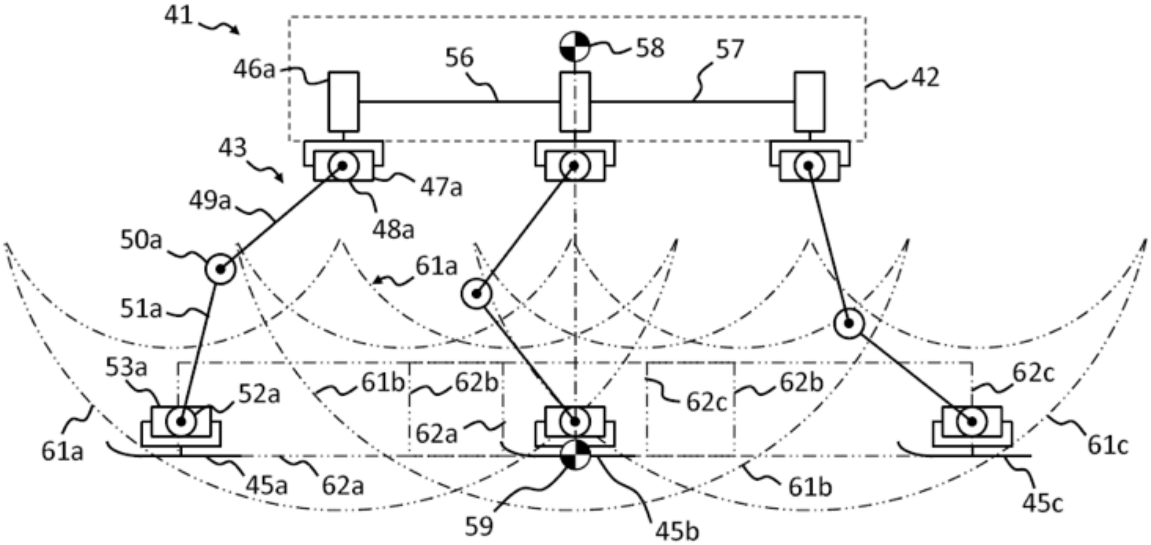


Figure 3.3. Skeletal side view of the single track or in-line three legged mobile robot illustrating the range of motion of the legs along the major direction of travel. (From US patent 8,457,830 Fig 2.)

From left to right in Figure 3.3, the first dashed cardioid 61a envelops and illustrates the total range of motion for the center bottom of foot (hereinafter the center bottom of each foot is referred to as the foot), and a second rectangular dashed box 62a inscribed inside the first dashed cardioid and illustrates the working range of motion for the foot. Two more dashed cardioids 61b and 62c and two more dashed boxes 62b and 62c illustrate the total range of motion and working range of motion for the middle and rear feet, respectively. In operation, Figure 3.3 shows the range of motion for each foot of each leg wherein there is a maximum working envelope and a typical working range for legged locomotion. Both the maximum working envelope and typical working range are three-dimensional volumes, but are shown as two-dimensional areas for clarity. Because the leg system shown uses hip, knee, and ankle joints, as previously described, the maximum working envelops are unique for that leg geometry. It would be different for pantograph legs, for example.

An aspect of the invention is the overlap of the typical leg working range, shown as dashed rectangular boxes in Figure 3.3, within the maximum working envelope, shown as dashed cardioids. It is highly desirable to have the ST3LMR operate on uneven surfaces, such as along narrow trails and paths, for example those found in parks and wilderness areas. That is, at a minimum, the maximum working envelope of the front foot overlaps with the middle foot, and the middle foot overlaps with the rear foot. At a minimum, the typical working range the front foot overlaps with the middle foot, and middle foot overlaps with the rear foot. In other words, the typical working range of the front leg overlaps with the middle leg and the middle leg overlaps with the rear leg to enable in three dimensions the front foot to be positioned alongside the middle foot and the middle foot to be positioned alongside the rear foot without mechanical interference.

Figure 3.4 schematically shows a three-dimensional perspective view of a leg configuration wherein each foot is displaced to left or right side of the projected center of gravity such that the front foot and rear foot traverse a centerline 64 that is spatially displaced but parallel to the centerline of the middle foot. Figure 3.5 schematically shows the top view of Figure 3.4 wherein the centers of the feet are shown as dots, displaced about the center of gravity and its ground projection center of pressure (not shown for clarity) of the body. In terms of the zero moment point (ZMP), the feet are positioned in a tripod stance such that the net moment or torque about the projected center of gravity, the center of pressure, is zero, which prevents the robot from falling over in either the pitch or roll direction. The legs and feet may be configured in infinite variety of tripod stances such that the projected center of gravity, the center of pressure, is contained within the foot extent, preventing the robot from falling over.

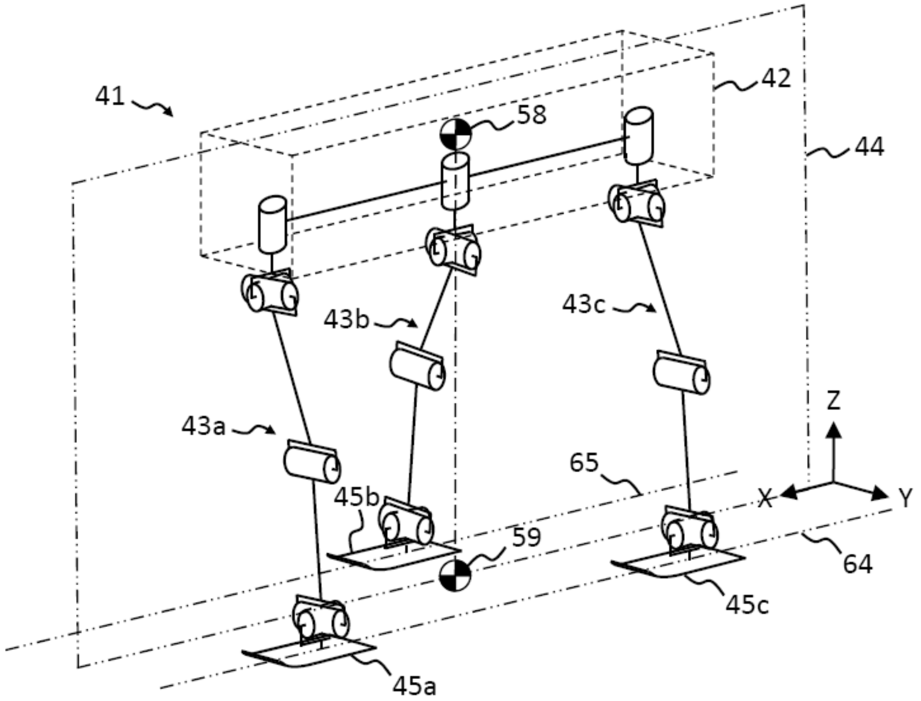


Figure 3.4. Skeletal perspective view of the single track or in-line three leg mobile robot illustrating one possible configuration of the legs, typically used in a stationary stance. (From US patent 8,457,830 Fig 3.)

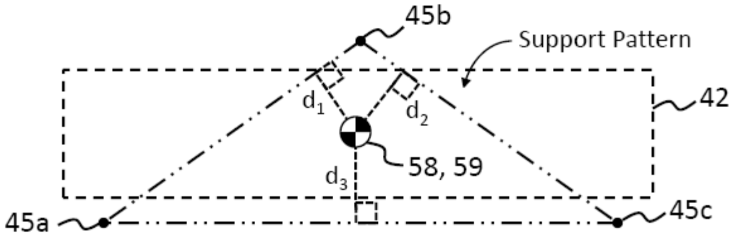


Figure 3.5. Top schematic view illustrating center of gravity, placement of the feet, and resulting area of support for the single track or in-line three leg mobile robot shown in Figure 3.4. (From US patent 8,457,830 Fig 4.)

In a second example, Figure 3.6 schematically shows a side view where the front leg is fully extended rearward and the middle leg is fully extended forward to form a stable tripod stance. Figure 3.7 schematically shows the top view of Figure 3.6 wherein the centers of feet are shown as dots, displaced about the center of gravity and its ground projection center of pressure (not shown for clarity) of body in a tripod stance.

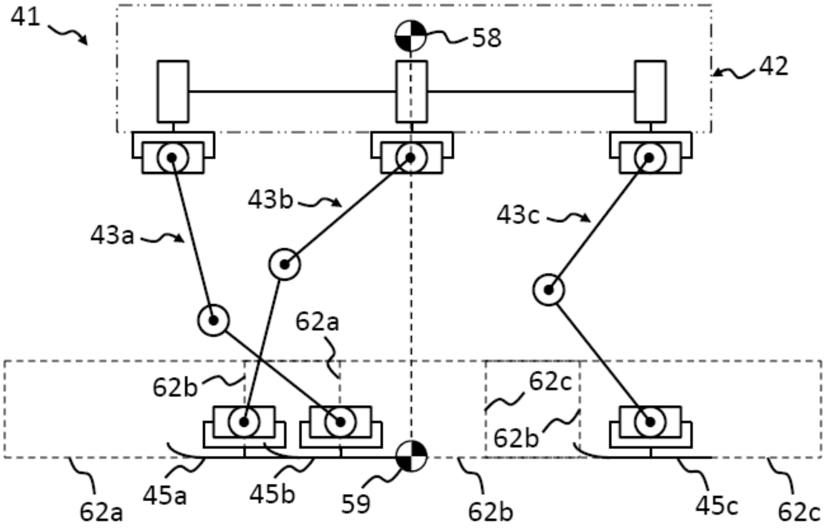


Figure 3.6. Skeletal side view of the single track or in-line three leg mobile robot illustrating overlap and crossover of the first and second legs, typically used while moving. (From US patent 8,457,830 Fig 5.)

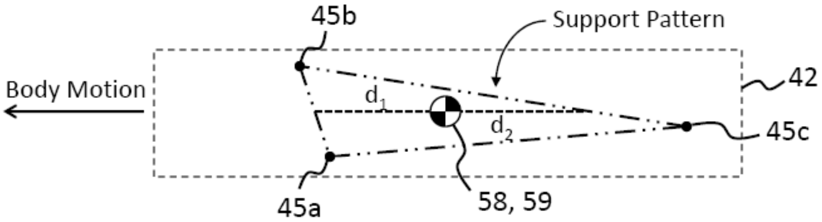


Figure 3.7. Top schematic view illustrating center of gravity, placement of the feet, and resulting area of support for the single track or in-line three leg mobile robot shown in Figure 3.6. (From US patent 8,457,830 Fig 6.)

The advantage of such a range of motion can be seen in Figure 3.6, wherein the extreme rearward position of the front leg and the extreme forward position of middle leg overlap such that the front and middle feet are positioned similar to that of a biped. This design and method is highly important so that a legged machine can achieve the desired stability of balance, leap and jump, land, and so on in addition to the legs being capable of operation in such a manner that it has a very narrow profile. A narrow profile is important so that it can maneuver in a space where mobile robots previously could not go, such as along a narrow path or trail or through a door. Additionally, this design also allows the stable two-beat cantor gait whereby stability of balance is gained through a bipedal-like stance of the front-middle or middle-rear leg pairs. Legged locomotion gaits are discussed in more detail in Chapter 4.

3.2. Method of Operation

This subsection describes the method of operating a single track legged vehicle having a body and at least three in-line legs aligned one behind the other. The method of operation comprises controlling each in-line leg of the single-track vehicle to coordinate movement of the in-line legs along a desired single-track trajectory. Each in-line leg attaches at its proximal end to a frame of the body of the vehicle arranged substantially parallel to a major axis of the frame and the forward/backward direction of travel of the vehicle. Each in-line leg has a foot at its distal end, and the in-line attachment of the legs to the body results in a center of gravity and a center of pressure that are directly in line with the legs, when the legs are simply extended straight down from the body. This arrangement results in inherent instability along the roll axis, and it requires coordinated movement controlled by causing each in-line leg to selectively perform a stance-to-flight phase, a flight phase, a flight-to-stance phase, and a stance phase.

During the stance-to-flight phase of a corresponding in-line leg, controlling foot movement unloads reaction forces and torques between the foot of the corresponding in-line leg and the ground such that the foot of the corresponding in-line leg is lifted off the ground. During the flight phase of the corresponding in-line leg, controlling leg movement maintains an upright position of the body and that moves the foot of the corresponding in-line leg in the same general direction and at a generally faster rate as a major direction of motion of the body. During the flight-to-stance phase of the corresponding in-line leg, controlling foot positioning places the foot of the corresponding in-line leg on the ground according to the desired single-track trajectory wherein reaction forces and torques are developed between the foot and the ground. During the stance phase of the corresponding in-line leg, controlling foot force and torque such that foot-to-ground interaction develops reaction forces and torques that are transferred from the foot through the corresponding in-line leg to propel, torque, and stabilize the body in the x, y, z, pitch, roll, and yaw axes. Each in-line leg is transitioned between the stance to flight phase, the flight phase, the flight to stance phase, and the stance phase to propel and torque the body along three axes according to the desired single-track trajectory.

Accomplishing the above necessitates receiving sensed data from at least one accelerometer and at least one gyroscope mounted to the vehicle. The sensed data is utilized to determine velocity, acceleration, attitude and gravitational forces. The determined velocity, acceleration, attitude and gravitational forces are used in the control of at least one of the flight-to-stance phase, flight phase, stance-to-flight phase and stance phase.

Further, controlling foot movement during the stance-to-flight phase comprises controlling at least one of: foot position, movement, force, torque, extension velocity and acceleration, and retraction velocity and acceleration. Controlling leg movement during a flight phase comprises controlling at least one of: in-line leg movement and foot trajectory. Controlling foot positioning during the flight-to-stance phase comprises controlling at least one of: foot position, movement, force, torque, extension velocity and acceleration, and retraction velocity and acceleration. Controlling foot force and torque during the stance phase comprises controlling at least one of: foot position, movement, force, torque, extension velocity and acceleration, and retraction velocity and acceleration.

Furthermore, predictive control of the in-line legs is advantageous to propel and torque the body along three axes according to the desired single-track trajectory. This is accomplished by measuring frame speed and acceleration vectors, center of mass coordinates, and ground contact duration for each foot. Then, dynamically adjusting the length, force, and torque of the in-line legs to achieve a set of expected values according to the desired single-track trajectory. Continually determining deviations from the expected values, and compensating for the deviations achieves active balance of the legged vehicle.

To simplify control and reduce the throughput requirements of the controller is accomplished by decoupling the positioning of the in-line legs along a length of the body from positioning of the in-line legs along the width of the body and parallel to the ground, and controlling stability of balance of the vehicle over time in the roll direction.

A target walking pattern model may be selected from a plurality of walking pattern models, wherein each walking pattern includes a clock-driven model of the stance and flight phases for each in-line leg. Computing the desired single-track trajectory for the body of the vehicle would use at least one of a heuristic algorithm and a simulation algorithm to select a pattern of footholds from a set of reachable footholds that most closely correspond to the desired single-track trajectory, and which minimize dynamic momentum for lateral and roll axes. Dynamic momentum maintains the desired single-track trajectory during periods of single-leg support and double-leg support.

Each of the three in-line legs of the vehicle is controlled by continually sensing body attitude and roll angle throughout each of the stance to flight phase, the flight phase, the flight to stance phase and the stance phase.

For example, performing the flight to stance phase for a select in-line leg is accomplished by accelerating the foot of the select in-line leg backward and along the curved single-track trajectory before contact with the ground, until the foot is generally stationary with respect to the ground, and then making contact with the ground to develop any ground reaction forces and torques. Performing the stance to flight phase for a select in-line leg is accomplished by generally maintaining the foot stationary with respect to the ground while the foot is being unloaded of any ground reaction forces and torques; to perform a desired single-track turn maneuver.

Controlling each in-line leg of the vehicle comprises controlling a leg so as to position the foot to a select side of the projected center of gravity on to the ground to develop, during a stance phase, ground reaction forces that are generally normal to the major direction of motion and ground reaction torques in the pitch, roll, and/or yaw axis to maintain stability of balance along a desired single track trajectory.

Controlling each in-line leg of the vehicle comprises positioning the landing foot offset from the foot lifting off according to a pre-programmed strategy for stability of balance along the desired single-track trajectory.

Controlling each in-line leg of the vehicle comprises controlling the length of the in-line legs during an associated stance phase of each in-line leg so as to be different between feet of the in-line legs positioned on, to the right of, or to the left of the projected center of gravity of the body on to the ground to level the body attitude, within a working range of the in-line legs and their feet.

Controlling each in-line leg of the vehicle comprises controlling two of the in-line legs to transition from stance-to-flight and flight-to-stance phase in a generally make before break fashion such that both feet support the body. The landing foot is placed spatially apart from the foot lifting off, for stability of balance in the pitch axis along a desired single track trajectory.

A corresponding spatial volume is defined for each leg that limits possible in-line foot placement, each spatial volume constrained based upon pitch and roll movement at a hip joint that joins a corresponding in-line leg to the body, and based upon extension and retraction of the corresponding in-line leg by knee and ankle joints of the in-line leg. To this end, each in-line leg is controlled according to a leg motion model to operate each in-line leg within its defined spatial volume. Pairs of feet corresponding to pairs of in-line legs would have sufficient reach and movement range in length, width and height, relative to the body, to be placed in a bipedal stance, with respect to the major axis and major direction of motion and travel. Three feet corresponding to three in-line legs would have sufficient reach and movement range in length, width and height, relative to the body, to be placed in a tripedal stance. The combined ground reaction forces between two or three feet during a stance phase imparts a torque to rotate the body in the pitch, roll, and/or yaw axis and/or a force to propel the body in at least one of the x, y, and z axes.

Feedback and control signals from an operator interface system, in communication with the control system, wherein the operator interface feedback and control signals provide at least steering angle, throttle and braking control signals to the control system to enable the operator to control stability of balance in the roll axis of the vehicle.

Each in-line leg of the vehicle comprises controlling at least three in-line legs according to an elastic-mechanical and dynamical model to compute foot position of each in-line leg to maintain body stability along a desired single-track trajectory. Controlling at least two in-line legs by inducing select ones of roll, pitch and yaw torques between the foot of a corresponding in-line leg and the ground and/or at least two in-line legs by selectively inducing roll, pitch and yaw torques between each foot of at least two controlled in-line legs and the ground to control a body trajectory along a desired single-track trajectory. Elastic deformation of at least one leg is predicted when in support with the ground to maintain desired ground reaction forces and torques to control a body trajectory along a desired single-track trajectory. Wherein the body comprises a segmented frame, segmented frame joint angles are computed in at least one axis between adjacent ones of the in-line legs to ground undulations and to the curvature of a single track turn maneuver with a minimally narrow profile relative to the major axis of motion. Dynamically adjusting the length, force, and torque of the in-line legs to maintain body stability is based upon the aforementioned computations. Further, computing elastic energy storage and release components between frame segments and in the in-line legs, wherein the elastic components operate in at least one axis, wherein the elastic components store and release kinetic energy for transfer between body segments and in-line legs, would enable dynamically adjusting the length, force, and torque of the in-line legs to maintain body stability. Computing the dynamic momentum forces and torques developed by moving frame segments and corresponding ones of the in-line legs relative to each other, wherein the components operate in at least one axis, would enable dynamically adjusting the length, force, and torque of the in-line legs to maintain body stability.

An elastic-mechanical and dynamical model would also enable performing the flight phase for a select in-line leg by computing dynamic momentum forces and torques developed by moving an in-line leg in flight phase, wherein the components operate in at least one axis, to dynamically adjust the center of mass of the leg in the flight phase to maintain body stability. This behavior can be accomplished by sweeping a select in-line leg inward or outward normal to the body and major direction of travel, reducing or extending the leg and a corresponding position of the center of mass of the leg, so as to impart a desired torque in the pitch, roll and/or yaw axis to maintain stability of balance along the desired single-track trajectory.

For semi-autonomous behavior, controlling the single track vehicle to operate autonomously regardless of at least steering angle and throttle control signals to sense and prevent turnover while enabling normal riding techniques in all but out of control situations.

Controlling each in-line leg of the vehicle comprises controlling each in-line leg so as to vary the length of each in-line leg with respect to the average body height during a stance phase such that the body is stable in height, roll, and pitch over uneven ground.

Controlling each in-line leg of the vehicle comprises controlling the in-line legs so as to incline the body in the pitch axis to lower the front of the body and raise the back of the body when ascending a gradient, or to raise the front of the body and lower the back of the body when descending the gradient. Similarly, controlling the in-line legs to roll the body in the roll axis when traversing a gradient normal to the major direction of travel, within the working range of the in-line legs and their feet, provides maximum rider comfort.

Finally, a follow-the-leader gait is accomplished by positioning the foot of the forward most in-line leg to control the trajectory of the single-track path by placing a second foot corresponding to a second one of the in-line legs in proximity of a first foot corresponding to a first one of the in-line legs, placing a third foot corresponding to a third one of the in-line legs in proximity of the second foot, and repeating positioning the first foot, placing the second foot, and placing the third foot such that each foot follows a desired single-track trajectory. Single-track gait theory for level and irregular terrain is discussed further in Chapter 4.

3.3. Novelty

Regarding originality or novelty of the ST3LMR, US patent law establishes a prima facie case of obviousness using a three-prong analysis. First, there must be no knowledge of the design, either in the references or in the knowledge generally available among those of ordinary skill in the art. Second, there cannot be a reasonable expectation that modifying the prior art will result in the design. Third, the prior art must present or suggest all of the limitations. In US patent 5,929,585, entitled “Robot System and Its Control Method”, Fujita first describes a robot using three out of four legs to support and propel a robot (Fujita 1999, column 2, lines 52-61):

> “In the walking attitude, the shell of the robot system is held almost parallel with the horizontal plane (or support 55 surface). In this condition, the controller causes three out of four of the leg mechanisms to move along the support surface in the same direction without causing the remaining leg mechanism to be supported on the support surface. Therefore, the robot can walk by obtaining the horizontal 60 component of the reaction received from the horizontal plane by the

> supported three leg mechanisms and moving in the same direction as the forward movement of the shell."

Then Fujita describes a method wherein the three out of four legs are used to walk (Fujita 1999, column 8, lines 9-13):

> "As shown in FIGS. 6 to 9, the walk robot 10 when walking holds the longitudinal direction of the shell 11 almost in parallel with the horizontal plane 42. Moreover, the walk robot 10 grounds three out of four leg mechanisms 12 to 15 to move to the same direction such as the backward direction and moreover rotates, for example, forward (in this case, toward the head 40 in the longitudinal direction of the shell 11) without grounding one remaining leg mechanism. Thereby, the walk robot 10 can walk by obtaining the horizontal component of the reaction received from the horizontal plane 42 by three leg mechanisms grounded and moving in the same direction as a thrust for forward movement of the shell 11."

However, Fujita describes that the legs are attached to the frame in a rectangular arrangement at four points 17, 22, 27, 32 with the center of gravity 41 between them. This arrangement is inherently stable. The ST3LMR leg arrangement discussed in this dissertation is not known in the prior art, including Fujita. Specifically, the ST3LMR in-line attachment of the legs to the body results in a center of gravity and a center of pressure that are directly in line with the legs, when the legs are simply extended straight down from the body, resulting in inherent instability. Fujita fails to disclose a one-behind-the-other (in-line) arrangement of leg attachments to the body.

Fujita also fails to disclose a control system having a gyro and an accelerometer. This may be due to Fujita’s inherent stability. A close review of Fujita discloses no gyro and no accelerometer. Instead, Fujita discusses an inherently stable four-legged robot which determines horizontal not by gyro or accelerometer, but by “three leg mechanisms grounded and moving in the same direction as a thrust for forward movement…” (Fujita 1999, column 8, lines 8-12). No gyro or accelerometer is disclosed. This is possible with Fujita due to the inherent stability of Fujita.

Herr, Seyfarth, and Geyer describe a legged robot controller that uses a gyroscope (Herr et al. 2002, paragraph 28):

> “When the proximal end 19 of the leg contacts a surface 20 (e.g. the ground) a leg-force ("FLEG") is generated. The joint 16 is equipped with a position sensor that, when combined with a gyroscope, measures an orientation angle x of the leg with respect to the surface 20. The leg angle ax thus describes the leg orientation with respect to the surface 20.”

Then a method wherein the three out of four legs are used to walk is described (Herr et al. 2002, paragraph 71):

> “It should be further noted that, although an angle position sensor and a gyroscope is disclosed in the embodiment, the invention will also be applied to any sensory setup, which yields the leg orientation with respect to gravity. It should be further noted that, although an angle position sensor and a leg length sensor is disclosed in the embodiment, the invention will also be applied to any sensory setup, which yields the system energy and the instant of apex. It should further be noted that,

> although a force sensor is disclosed in the embodiment, the invention will also be applied to any sensory setup, which detects contact and flight phases of the mobile legged robot."

In the description, the gyroscope is not identified individually, rather it seems to be a part of the joint 16, as illustrated in Figure 1, and it is part of a sensor system designed to yield the leg orientation with respect to gravity. To the contrary, however, the ST3LMR body-mounted accelerometer and gyroscope determine body motion in an in-line three-legged robot. There is no suggestion or motivation to modify Fujita in view of Herr, and the combination of Fujita and Herr fails to disclose the ST3LMR. Thus, one skilled in the art would not find ST3LMR obvious in view of the combination of Fujita and Herr.

In continuing the detailed design, Chapter 4 presents various single-track gaits for the ST3LMR and discusses various methods for traversing irregular terrain. Chapter 4 also provides a detailed example of how the ST3LMR traverses a vertical step.

CHAPTER 4

SINGLE-TRACK GAITS FOR A LEG STATE MACHINE

In the late 1800’s and early 1900’s, Muybridge used stop-motion photography to study legged locomotion in animals (Muybridge 1899) and later humans (Muybridge 1901). His work is an invaluable reference for understanding quadruped and biped walking gaits, and it has inspired and shaped the design of countless biologically-inspired legged machines. This work departs from that legacy by introducing new walking gaits based on the non-biologically inspired single-track, three legged design, described in the previous chapter. However, the reader will find that these new single-track gaits share similar traits with animal gaits. For example, the backward wave gait, the trot, the pace, the three beat, and pronking gaits as well as bounding and jumping performed by animals may all be accomplished with three in-line legs.

These gaits form the knowledge base for the Leg State Machine and Monte Carlo based motion-planning systems, both described in detail in later chapters. A Leg State Machine (LSM) is responsible for implementing the various gaits. The purpose of a LSM is to prevent the legs from becoming uncoordinated, which could then lead to the robot falling over. This chapter introduces and presents Gait Diagrams. A single-track gait strategy is proposed that keeps two feet on the ground at all times and decouples control in the pitch and roll axes. This method drastically simplifies control and the modes of operation. This chapter also introduces various gaits for level terrain to illustrate the feasibility of the design.

4.1. Gait Diagrams

A foot provides a discrete area of support on the ground for a finite period of time. This chapter makes use of the Gait Diagram to illustrate when the foot is in contact with the ground and supporting the robot, called the stance phase of the leg, and when the foot is swinging freely in the air, called the flight phase of the leg. The Gait Diagram shows the possible states of each leg and prescribes the leg state machine, but it does not specify the foot rate or how the ankle, knee, or hip are moved. A pattern of footholds from the set of reachable footholds can be found that would provide a desired trajectory, afford balance, and minimizes the dynamic momentum for lateral and roll axes.

Figure 4.1 is an example of a Gait Diagram illustrating a Backward Wave Gait (BWG) comprising a 2/3 support phase (illustrated as a black horizontal line, also called beta) and 1/3 swing phase (illustrated as no line) along the horizontal axis for each of the three legs (a, b, and c) shown on the vertical axis. It is called a BWG because the flight phase of the legs progresses from the rear leg c to the middle leg b to the front leg a. In this example, the three flight phases are evenly dispersed over the period. For clarity of understanding, Figure 4.2 illustrates how the legs and feet of a robot move when executing the BWG illustrated in Figure 4.1.

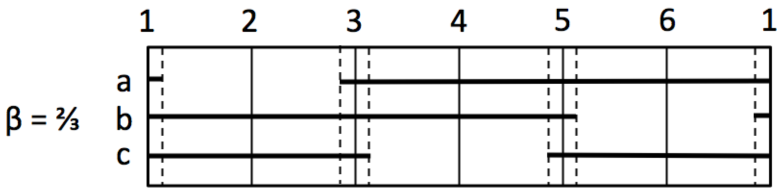


Figure 4.1. Example of a backward wave gait with 2/3 stance and 1/3 flight phase, showing make-before-break overlap at time 1, 3, and 5.

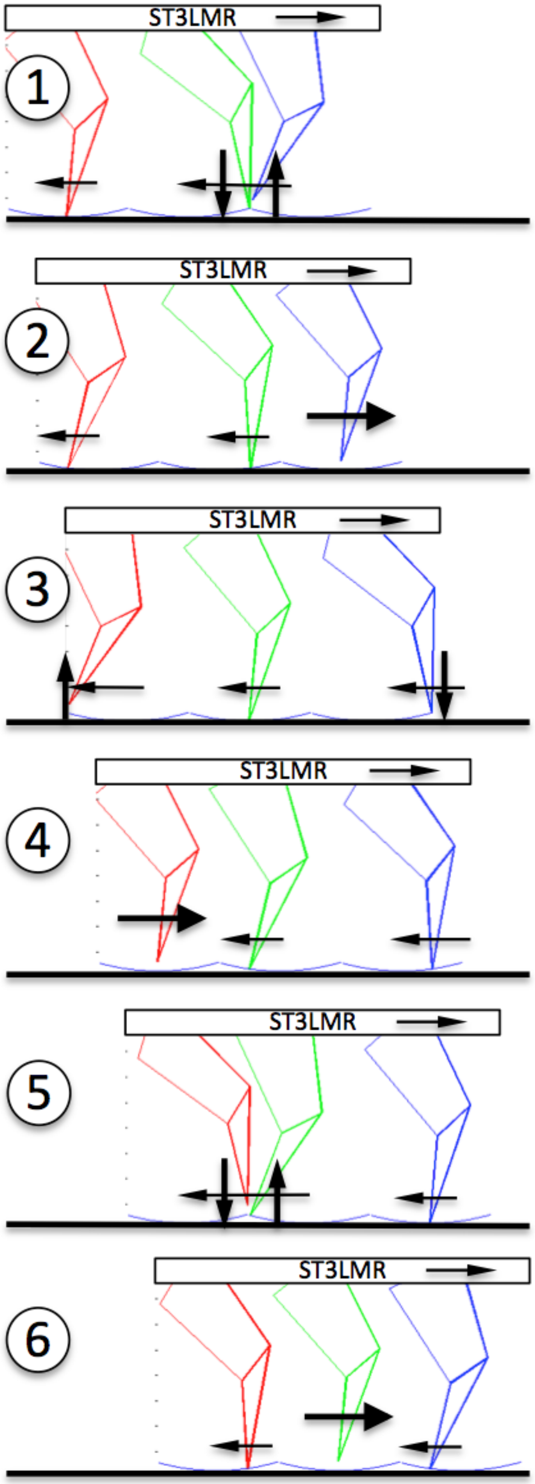


Figure 4.2. Example of the ST3LMR legs executing a backward wave gait with 2/3 stance and 1/3 flight phase.

Referring to Figures 4.1 and 4.2, at (1) all three feet (front, middle, and back or a, b, and c or blue, green, and red, respectively) are on the ground. The middle leg b is transitioning from the flight to stance state and the front leg a from stance to flight state. At (1), there is a make before break transition between the middle and front foot such that both feet are on the ground supporting the robot simultaneously. All three legs are moving the feet backward with respect to the robot body, propelling the robot forward. Further, the legs may be spaced apart to the left and right of the robot (in and out of the page, respectively) such that the middle and front legs for a bipedal stance and all three legs form a tripod stance to (briefly) stabilize the robot in the roll axis, preventing (or recovering) the robot from falling over to the left or right.

At (2), both the middle leg b and back leg c are in stance and moving rearward while the front leg a is in flight, moving the foot forward. Generally speaking, the forward velocity is always greater than a rearward velocity, and this is depicted as a heavier arrow in Figure 4.2. At (3), the front leg a transitions from the flight to stance state and the rear leg c from stance to flight state. At (4), both the front leg a and middle leg b are in stance and moving rearward while the rear leg c is in flight, moving the foot forward. At (5), the rear leg c transitions from the flight to stance state and the middle leg b from stance to flight state. At (6), both the front leg a and rear leg c are in stance and moving rearward while the middle leg b is in flight, moving the foot forward. The cycle then returns to (1), with the middle leg b transitioning from the flight to stance state and the front leg a from stance to flight state. The (1)-(6) cycle repeats, propelling the robot forward.

Regarding the stability of the robot along its major axis of motion or the pitch axis, note that from (1) to (2) the middle leg b is ahead of the center of the robot body and the center of mass, preventing the robot from pitching or tipping forward. At (3) all three legs momentarily support the robot. From (4) to (5), the middle leg b is behind the center of the robot body and the center of mass, preventing the robot from pitching or tipping backwards. From (5) to (1), the front a and middle b legs support the robot. This gait method prevents the robot from pitching forward or backward. It also serves to decouple control in the pitch and roll axes, which as will be shown in later chapters, simplifies the control system and hardware requirements.

The horizontal axis of Figure 4.1 is a dimensionless representation of both time and distance traveled, and the Gait Diagram typically illustrates one complete cycle or gait. In practice, the horizontal axis is scaled as a function of the velocity of the robot body. At any unit or fractional unit on the horizontal axis, the state of the legs (e.g., stance or flight) is determined by reading the diagram (e.g., line or no line, respectively). In this fashion, the LSM determines which foot to reposition (i.e., which foot to transition from stance to flight and from flight to stance) to support the robot as the body moves forward in time. The gait diagram and LSM constrains the allowable motion of the feet.

Note that the LSM may switch between different gaits (e.g., the different Gait Diagrams presented in this chapter) to achieve different robot velocities or to produce useful actions, such as jumping. For example, the BWG is typically employed during slow robot velocities and then transitioned (during periods of equivalent foot states) to a three beat gait for higher velocities. Further, rear legs may re-use the same footholds of the forward legs, resulting in a follow-the-leader gait. Having multiple gates is also useful for recovering balance during periods of instability. While gait planning and selection is still an open research topic, Chapter 7 develops a Monte Carlo method that solves this problem.

4.2. Single-Track Three Legged Gaits

The operation of the single-track, three legged mobile robot LSM will now be explained with reference to a gait model or foot state diagram. The Gait Diagram, illustrated in Figure 4.3, shall now be explained as an example for all of the following gait diagrams. From clock 0 to 8, the front leg a is on the ground and supporting the robot, called stance state. From clock 8 to 0 (the clock is reset at 12), the front leg a is in the air and not supporting the robot, called flight state. The ratio of the stance units to the total cycle time is 8/12 or 2/3. In Gait Diagram terminology, this ratio is called a "2/3 stance" or "2/3 Beta." The duration of the flight state is similarly defined by the ratio of flight state to total cycle time, or "1/3 flight" as illustrated in Figure 4-3. From clock 8 to 4 (the clock is reset to 0 at 12), the middle leg b is in the stance state. From clock 4 to 8, the middle leg b is in the flight state. From clock 4 to 0 (the clock is reset to 0 at 12), the rear leg c is in the stance state. From clock 0 to 4, the rear leg c is in the flight state. It is called a BWG because the flight phase of the legs progresses from the rear leg c to the middle leg b to the front leg a.

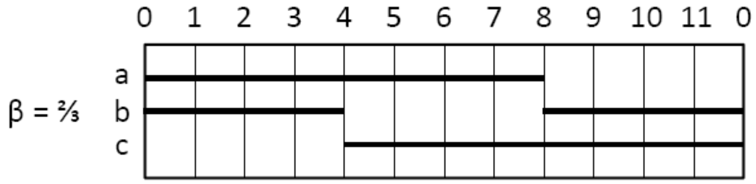


Figure 4.3. Example of an ideal backward wave gait with 2/3 stance and 1/3 flight phasing.

The forward velocity of any foot during flight state is twice the velocity of the foot during stance state. This is determined from the Gate Diagram by measuring the ratio of the stance to flight units along the horizontal axis, i.e., 8 units in stance divided by 4 units in flight yields 2x. If the robot is moving forward at 5 m/s, for example, the feet on the ground (or in stance state) are moving rearward at 5 m/s with respect to the robot body. When a foot transitions from stance to flight to stance, it must be moving forward at 5 m/s with respect to the robot body to reposition the foot for the next support period. Thus, the Gait Diagram is also used to determine the (minimum) required leg actuator velocities. In practice, the forward leg swing typically occurs at maximum leg velocity to maximize ground support and stability of balance.

The following five Gait Diagrams illustrate variations of the BWG wherein the Beta is changed and the timing of the flight states varies from evenly dispersed over the gait period to more grouped relationships. Again, having multiple gates to select from is used for recovering balance during periods of instability as well as performing different actions, such as jumping. Achieving a fast forward velocity, for example, requires a different gait, such as a two-beat gait, where two or more legs are in flight at the same time.

Figure 4.4 is an example of a backward wave gait model for the single track three legged mobile robot with a 5/6 stance (beta) and 1/6 flight phasing. Generally, a long stance with respect to swing (large beta) reflects a slow body velocity, as the forward swing typically occurs at maximum leg velocity to maximize ground support and stability of balance. Assuming constant flight velocity, a 5/6 beta corresponds to a slower robot forward velocity than a 2/3 beta.

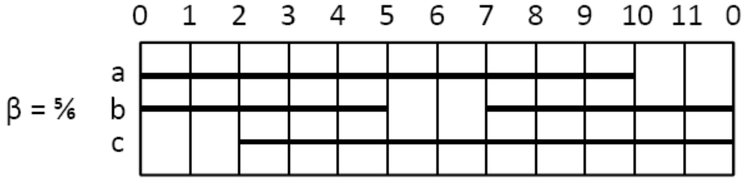


Figure 4.4. Example of a backward wave gait with 5/6 stance and 1/6 flight phase.

Figure 4.5 is an example of a gait model or leg state diagram illustrating a backward wave gait comprising a 10/11 stance (beta) and 1/11 flight phasing. Assuming constant flight velocity, a 10/11 beta corresponds to a slower robot forward velocity than a 5/6 beta, which is slower than a 2/3 beta.

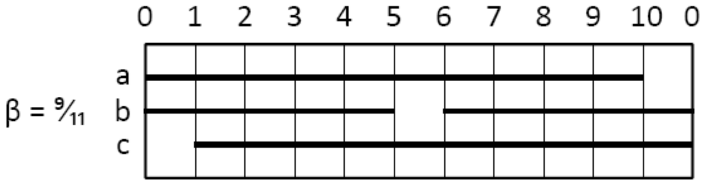


Figure 4.5. Example of a backward wave gait with 10/11 support phase and 1/11 swing phase.

In practice, the LSM would dynamically adjust beta in response to the commanded velocity and other factors, such as actuator temperature. For electric actuators, a smaller Beta results in slower peak and average actuator velocities, lower current consumption, and cooler operation. On the other hand, a 2/3 beta (Figure 4.3) as compared to a 10/11 (Figure 4.5) provides less time to implement bipedal and tripod stances, which provide stability of balance. Chapter 7 develops a Monte Carlo method that addresses this problem.

Figure 4.6 is an example of a backward wave gait model for the single track three legged mobile robot 41 with a 5/6 stance (beta) and 1/6 flight phasing where the swing cycles are grouped together. Grouping the leg flight states is useful for jumping, landing, and quickly repositioning all three legs to catch a fall.

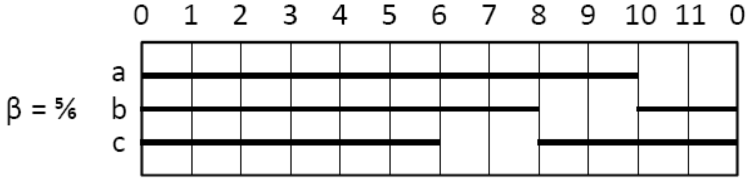


Figure 4.6. Example of a backward wave gait with a 5/6 stance and 1/6 flight phase where the swing cycles are grouped together.

Figure 4.7 is an example of a backward wave gait model for the single track or in-line three legged mobile robot 41 with a 8/11 stance (beta) and 3/11 flight phasing with two intervals where all three legs are simultaneously supporting the body. This backward wave gait closely resembles a trot gait. Trotting is running at a rapid speed, and pacing is running at constant speed. Note that from (3) to (4) and (7) to (8), all three legs may be arranged in a tripod stance, which provides stability of balance and prevents the robot from falling over.

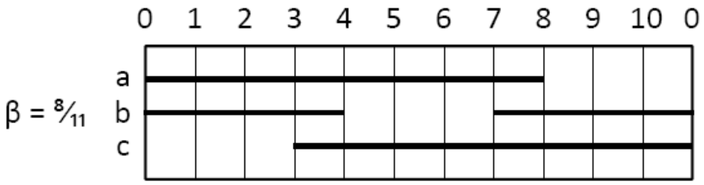


Figure 4.7. Example of a backward wave gait with 8/11 stance and 3/11 flight phasing with two intervals where all three legs are simultaneously supporting the body.

Figure 4.8 is an example of an equal phase backward wave gait model for the single track or in-line three legged mobile robot. Note at (6) there may be a make before break or break before make transition between the first leg a and rear leg b. Such a gait is useful for implementing a bipedal stance (viewed along the length of the body) to maintain stability in the roll axis and preventing the robot from falling over.

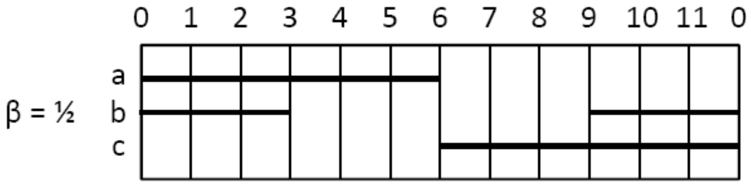


Figure 4.8. Example of an equal phase backward wave gait.

Figure 4.9 is an example of a variation on an equal phase backward wave gait model for the single track or in-line three legged mobile robot. Note that the Gait Diagram illustrates two cycles for clarity. This gait uses the dynamic momentum of the body to afford balance during (5) through (7), where only the middle leg b supports the robot. Brief support periods from (1) to (2) and (4) to (5) allow all three legs to maintain the general stability, forward speed, and direction of travel of the robot body.

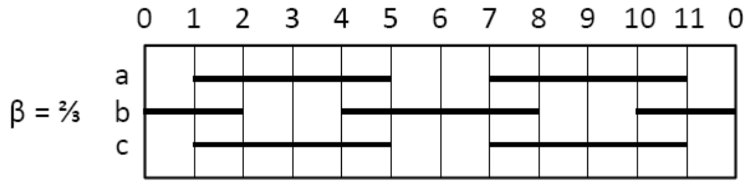


Figure 4.9. Example of a variation on a two-beat equal phase backward wave gait.

Figure 4.10 is an example of a variation on an equal phase backward wave gait model for the single track or in-line three legged mobile robot. This is an example of a trot or pace gait wherein two legs provide pitch stability (a bipedal stance) during one half of the gait period (from (0) through (6)) and rely on dynamic momentum of the body to provide or coast through the other half period (from (6) through (0)).

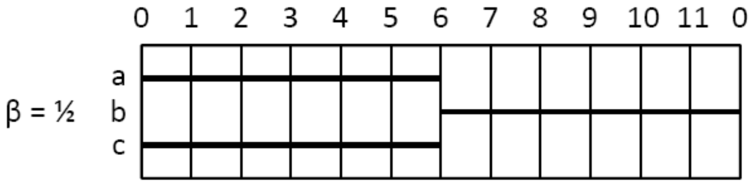


Figure 4.10. Example of a variation on a two-beat equal phase backward wave gait.

Figure 4.11 is an example of a variation of the 2/3 stance and 1/3 flight backward wave gait for the single track or in-line three legged mobile robot, wherein affordance is given to reposition the middle leg during the front and rear leg support. Note at (0) there is a make before break transition between the legs to provide stability in the pitch axis. This gait is useful for recovering from a sideways impulse, such as when a rider shifts weight suddenly.

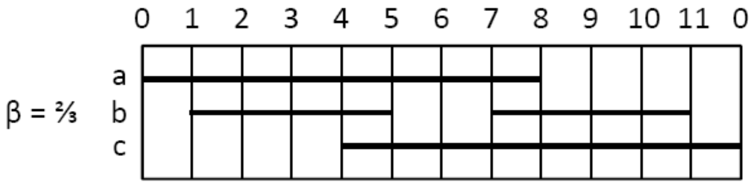


Figure 4.11. Example of a variation of a 2/3 stance and 1/3 flight backward wave gait, wherein affordance is given to reposition the middle leg during the front and rear leg support.

Figure 4.12 is an example of a variation of a 2/3 stance and 1/3 flight backward and forward wave gait for the single track or in-line three legged mobile robot 41, wherein affordance is given to reposition the middle leg to accommodate changes in front and rear leg stance. Note at (2), (4), (8), and (10) there is a make before break transition between the legs to provide stability in the pitch axis. This gait is useful for recovering from a front leg or rear leg slip.

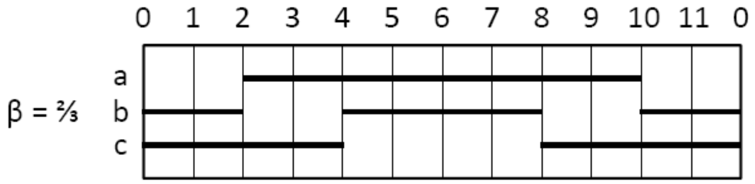


Figure 4.12. Example of a variation of a 2/3 stance and 1/3 flight backward and forward wave gait, wherein affordance is given to reposition the middle leg to accommodate changes in front and rear leg stance.

Figure 4.13 illustrated three examples of high robot velocity gaits. A two-beat fast trot gait combined with a hopping model is left, a three-beat running gait is middle, and a one-beat pronking gait is right. In the fast trot gait combined with a hopping model, the middle leg b provides the vertical hopping force and front and rear legs provide pitch stability (bipedal stance) during landing. Hopping, bounding, leaping, running, and jumping are characterized by periods wherein all feet leave the ground and the body is in ballistic flight. Legged robots that use a ballistic flight phase, are called dynamic legged robots, because leg extension directly affects forward momentum. Such gaits may be used in rugged terrain where only one foothold exists for support. Such ability of motion affords the legged machine to traverse terrain that is too difficult for comparable wheeled and tracked machines. Pronking is jumping with all three legs simultaneously followed by a period where the body is in ballistic flight.

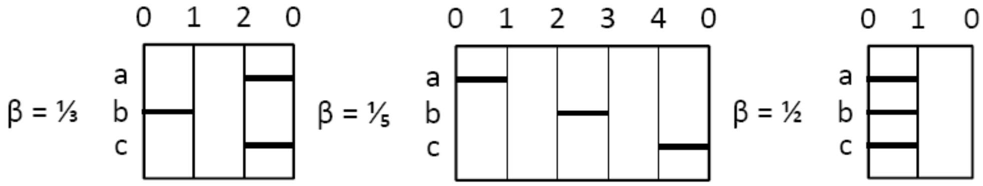


Figure 4.13. Examples of a two-beat fast trot gait combined with a hopping model (left), a three-beat running gait (middle), and a one-beat pronking gait (right).

Figure 4.14 is an example of a bounding gait for the single track or in-line three legged mobile robot. The gait is repeated twice for clarity. Bounding is jumping wherein the front, middle, and back legs alternately touch the ground. For bounding, the middle leg extends to the point of first overlap and the rear leg to the maximum forward extent for jumping. By positioning the feet in a wide stance, as opposed to an in-line or single-track stance, both legs contribute to the jumping force and stability is afforded in the roll and yaw axis with the first leg responsible for the pitch and yaw control. A series of adjustments in step length are required to arrive at a suitable takeoff point and correct leg states for leaping and jumping. The legs must be in a state to impart a vertical impulse to the ground such that the dynamic momentum of the body affords balance at landing. As such, the legs must not be at maximum extension.

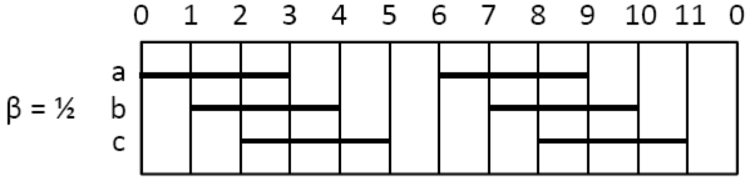


Figure 4.14. Example of a bounding or three-beat running gait with flight phase.

Figure 4.15 is a side skeletal view of the single track three legged mobile robot illustrating an example of a bounding takeoff (top) and landing (bottom). While airborne, the front leg extends to anticipate the first point of overlap and the middle leg becomes fully extended forward for landing. Note the symmetry between takeoff and landing.

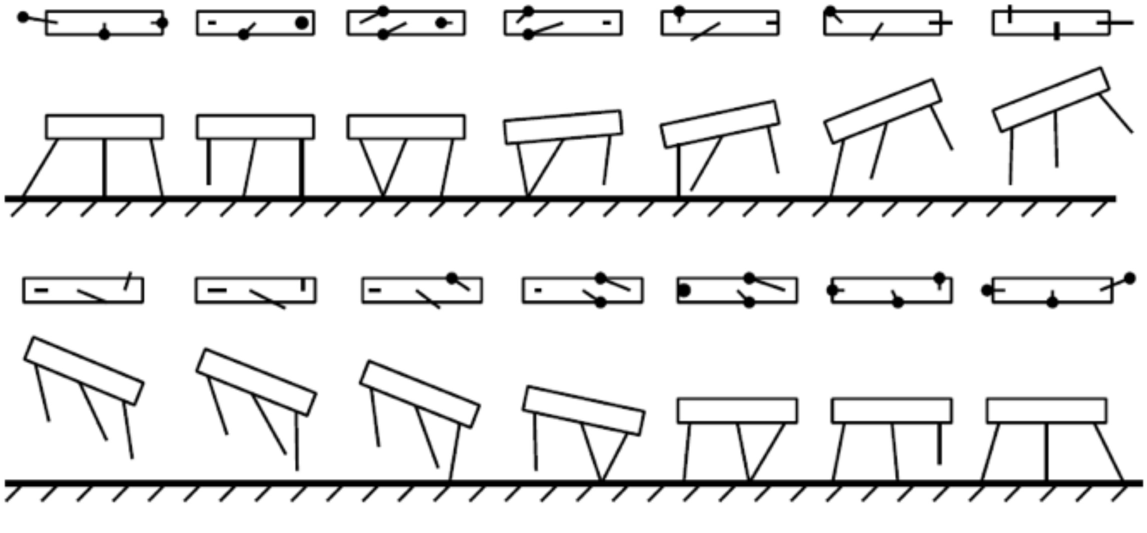


Figure 4.15. Side view of the single track three legged mobile robot illustrating a bounding takeoff (top) and landing (bottom) gait (viewed left to right).

4.3. Methods for Traversing Irregular Terrain

Irregular terrain is described by four geometric feature primitives: (a) gradient, (b) ditch, (c) vertical step, and (d) isolated wall, with the following assumptions (Song and Waldron 1989):

1) Vertical surfaces cannot be used as footholds;

2) Point contacts are used to model foot-to-ground interaction;

3) The terrain is solid, i.e., no foot-to-ground slip;

4) The legs are massless with the center of gravity on the body; and

5) Dynamic motion effects are not considered.

Figure 4.16 illustrates all four geometric features, showing a horse and rider on an uphill, rock strewn outdoors trail. The terrain is uphill, i.e., a gradient. The loose debris along the hill is unsuitable as footholds, i.e., ditches. At various points along the trail, rocks form vertical steps, and there are multiple rocks (foreground) and groups of rocks (middle) that form natural isolated walls to be used as footholds or traversed. The ST3LMR would accommodate shallow to moderate inclines by making slight adjustments to body posture, while it would accommodate steep inclines by also adjusting the walking gate pattern and using smaller steps.

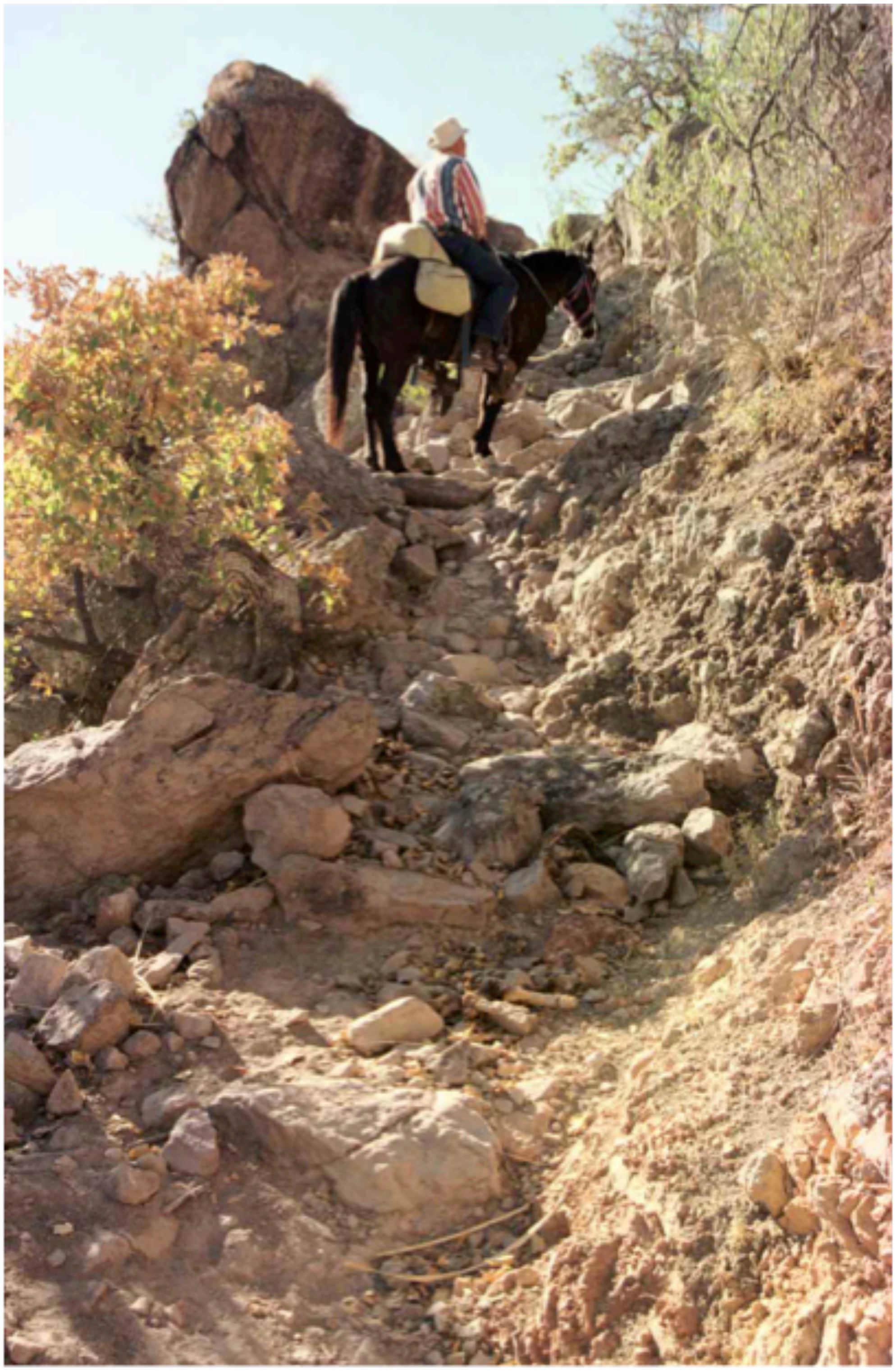


Figure 4.16. A mule and rider on an outdoors uphill trail, exemplifying irregular terrain.

By design, the three feet of the ST3LMR can be positioned in a tripod stance, forming an area of support on the ground. When the projected center of mass of the robot to the ground is kept within the triangular area of support, the robot is fully supported and will not fall over. Also by design, the front and middle or middle and rear feet can be positioned along side each other to form a bipedal stance and a line of support. As long as the projected center of mass of the robot to the ground is kept along this line of support, the robot is fully supported and will not fall over. By adjusting the position of the projected center of mass to the ground, a wide range of possible body orientations and reachable footholds is achieved. The following section presents a detailed example of traversing irregular terrain to illustrate the feasibility of the design, and by doing so develops the case for traversing the unlimited variety of irregular terrain found in nature.

The second method of adjusting the duration of the flight phase, i.e., adjusting the velocity of the leg actuators, is critical to quickly moving the foot from one position of support to another. This method would be used to "catch a fall" as the motion planning system leans the robot into the direction of the next foothold. A variant of the second method is to use vertical impulse (e.g., jumping, leaping, bounding, or pronking) to control step length. Note that both horizontal and vertical impulse may be used to control step length, with adjustments made during the flight phase.

The third method of adjusting the duration of the stance phase yields a small range of step lengths, and is likely to be used only for transitioning between gaits. The fourth method of stumbling to recover balance may be useful in response to a drop-step perturbation, but it is beyond the scope of this work. Rather, legs closest to their kinematic limits in the direction of motion of the body are lifted first, and legs with the largest kinematic range in the direction of motion are placed first. This method increases the probability that two legs will overlap in the next support phase.

4.3. Traversing A Vertical Step Example

To better understand the benefits of the ST3LMR design, an example of the ST3LMR traversing a vertical step is provided. Referring to Figures 4.17 through 4.26, the ST3LMR is illustrated in side view, and a method for positioning the feet to traverse a vertical step gradient is shown to demonstrate the concepts of mobility over irregular terrain. Through this single example, demonstrated mobility concepts include adjusting the walking height and/or body attitude on a gradient, the horizontal range, the vertical range, maintaining balance on three legs, maintaining balance on two legs, and a lateral side-step or shuffle maneuver. Such methods are fundamental to traversing uneven or rugged terrain, ditch crossing, wall crossing, climbing, and so on.

Starting with Figure 4.17, the ST3LMR illustrates one possible configuration of legs and foot placement before traversing the vertical or step gradient. Note the upper right insert showing the top view schematic of the ground to illustrate placement of the feet with respect to the robot body, in the (x-z) reference plane, and projected center of pressure. By convention, the left side is in the forward direction.

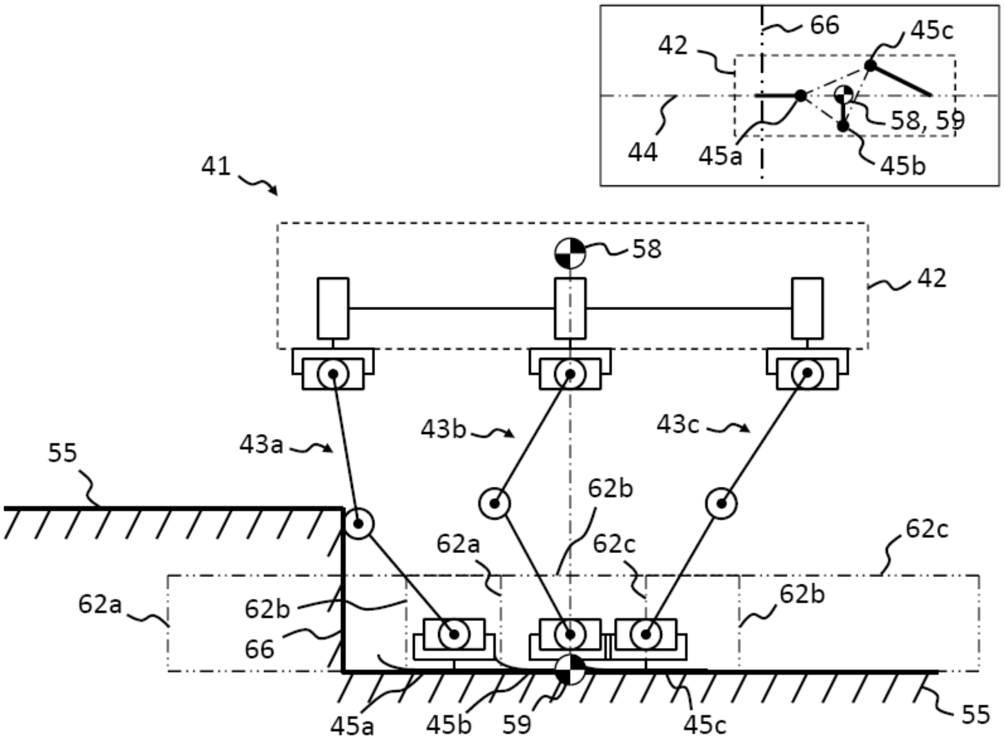


Figure 4.17. One possible configuration of leg and foot placement before traversing a vertical step gradient.

Next, Figure 4.18 shows the ST3LMR adjusting the walking height and/or body attitude in preparation to traverse the vertical or step gradient, by dropping the rear portion of the body. This maneuver has the effect of shifting the projected center of pressure rearward such that it intersects with a zero moment line bisecting the centers of the middle foot and rear foot. At this time, the middle and rear legs form a biped stance, and the robot is stabilized in the pitch and roll direction. Also note that the working range of motion for each foot has shifted. The front foot is now positioned at the lowest and most rearward position within its working range, the front portion of the working range has risen off the virtual ground plane of the lower ground, the middle foot and rear foot are within their working ranges, and the rear portion of the working ranges have sunk below the virtual ground plane of the lower ground. Note that there is still some margin between the working range and the maximum working envelope (not shown for clarity) to provide a safety margin for external dynamic events, such as a force imparted by a rider or the wind.

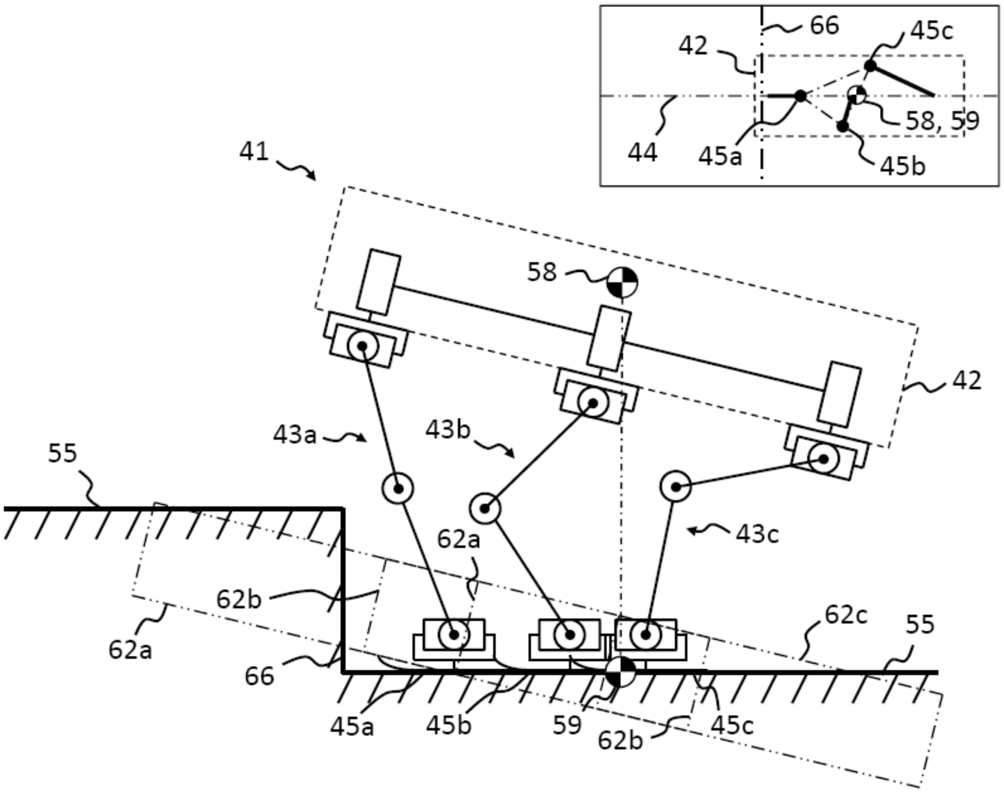


Figure 4.18. The single track legged mobile robot shown adjusting the walking height and/or body attitude in preparation to traverse a vertical step gradient.

Next, Figure 4.19 shows the ST3LMR lifting itself on the middle and rear legs while simultaneously lifting the front foot off the lower ground and repositioning it beyond the vertical or step gradient to be above the upper ground. Lifting the body is necessary to raise the working range to above the level of the upper ground. The body is lifted vertically until the middle foot reaches the bottom of the working envelope. Note again that there is still some margin between the working range and the maximum working envelope (not shown for clarity) to provide a safety margin for external dynamic events, such as a force imparted by a rider or the wind. This note is typical for further maneuvers and will not be re-noted for readability.

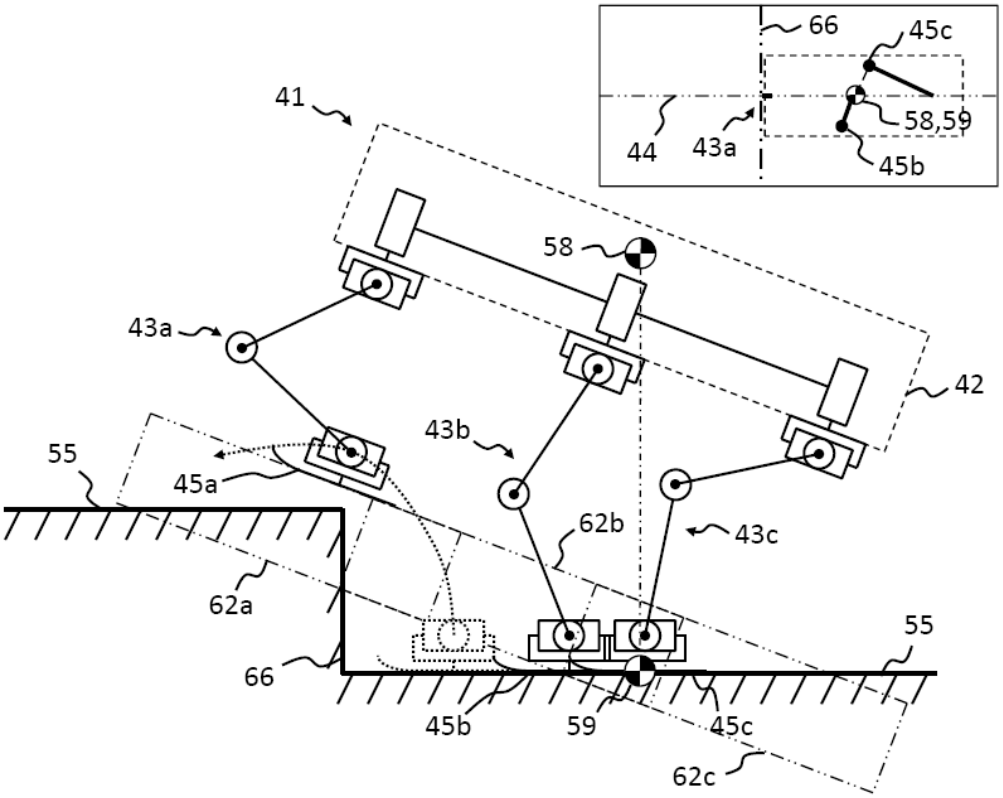


Figure 4.19. The single track legged mobile robot shown lifting itself on the middle and rear legs while simultaneously lifting the front foot off the ground and repositioning it beyond a vertical or step gradient.

Figure 4.20 shows the ST3LMR shifting its body and thus its projected center of pressure on the middle and rear legs to move the body and thus the front foot forward, while maintaining balance by keeping the center of pressure along the zero moment line that bisects the centers of the middle foot and rear foot. Note that in the case where all three legs are in contact with the ground, as in the stance phase of a wave gait, a similar shift of the body is used to maintain balance, and the legs may be successively repositioned in a similar manner such that the body moves sideways. This method of shifting the body and legs allows the legged mobile robot to move omnidirectionally.

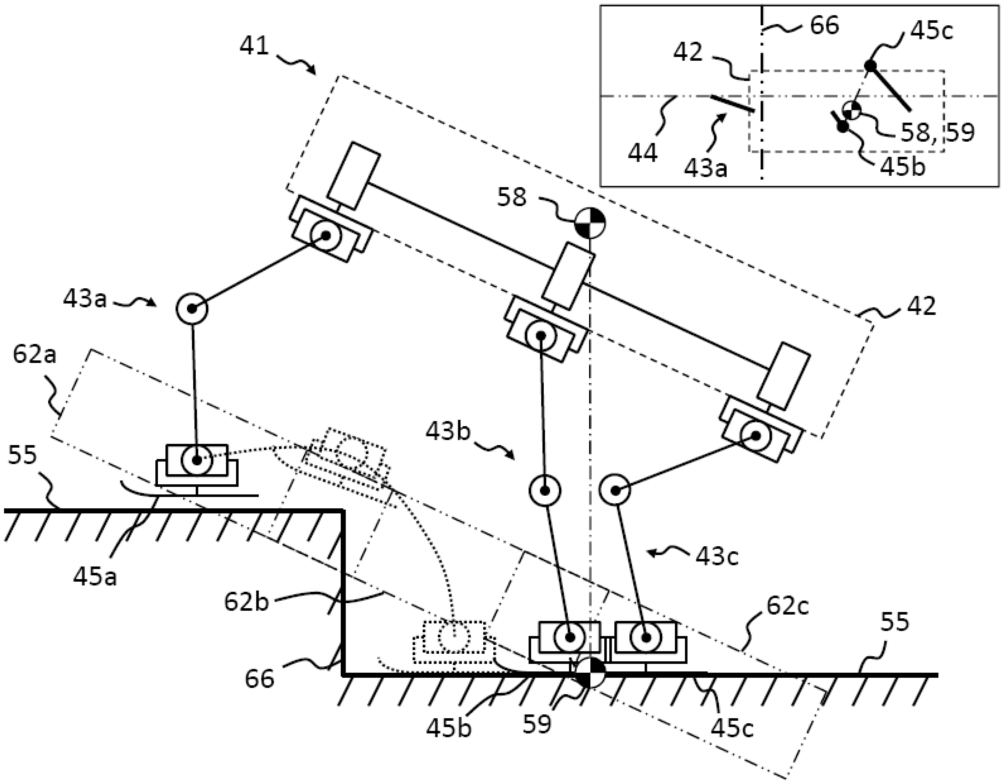


Figure 4.20. The single track tri-legged mobile robot shown shifting its body and thus its projected center of pressure on the middle and rear legs to move the body and thus the front foot forward, while maintaining balance in a biped stance.

Figure 4.21, and in particular the upper right insert, shows the ST3LMR placing the front foot on the upper ground and reestablishing the triangular three-point contact support pattern. At this time, any dynamic instability arising from, for example, measurement errors or external dynamic forces, acting on the bipedal stance are counteracted or reset by the more supportive tripod stance.

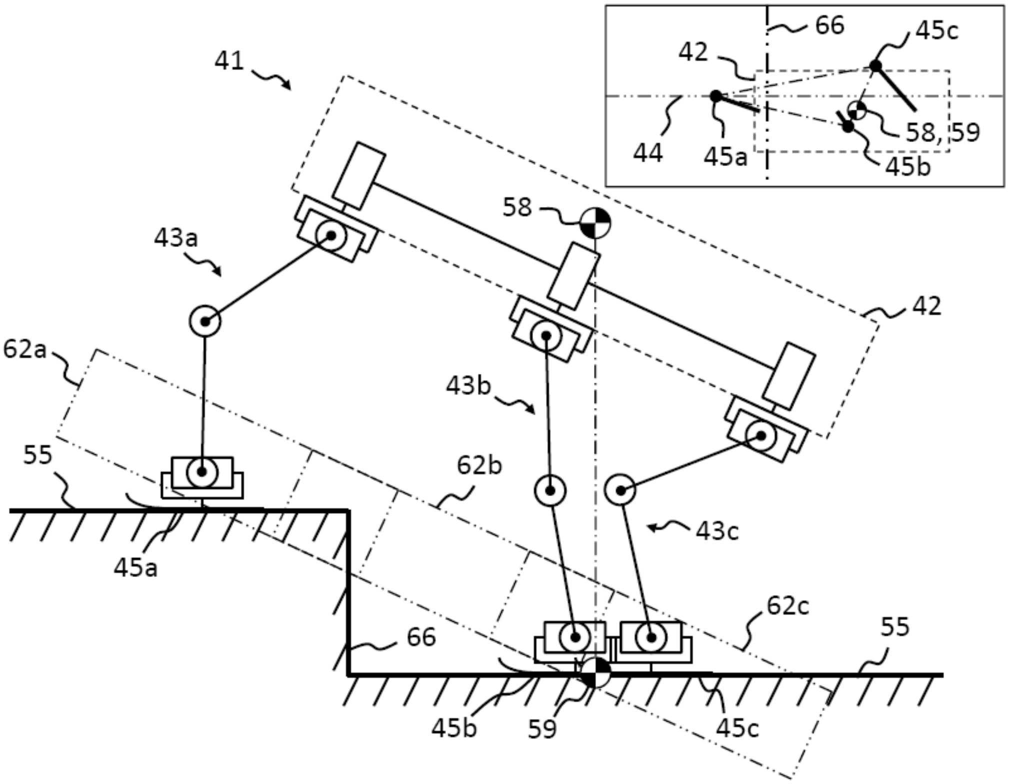


Figure 4.21. The single track tri-legged mobile robot shown (and in particular the upper right insert) placing the front foot on the upper ground and reestablishing the triangular three-point contact support pattern.

Next, Figure 4.22 shows the ST3LMR moving forward and shifting the center of pressure to along the zero moment line that bisects the centers of the front foot and rear foot to afford the robot to lift the middle foot while maintaining stability of balance by the front leg and rear leg in a biped stance. Note that in any dynamic system, the momentum imparted by moving the body forward and shifting is taken into account such that the middle leg may be lifted off the ground sooner than the static case of when the center of pressure must reach the zero moment line that bisects the centers of the front foot and rear foot to maintain static stability of balance. The middle foot is positioned above the upper ground and forward of the vertical or step gradient.

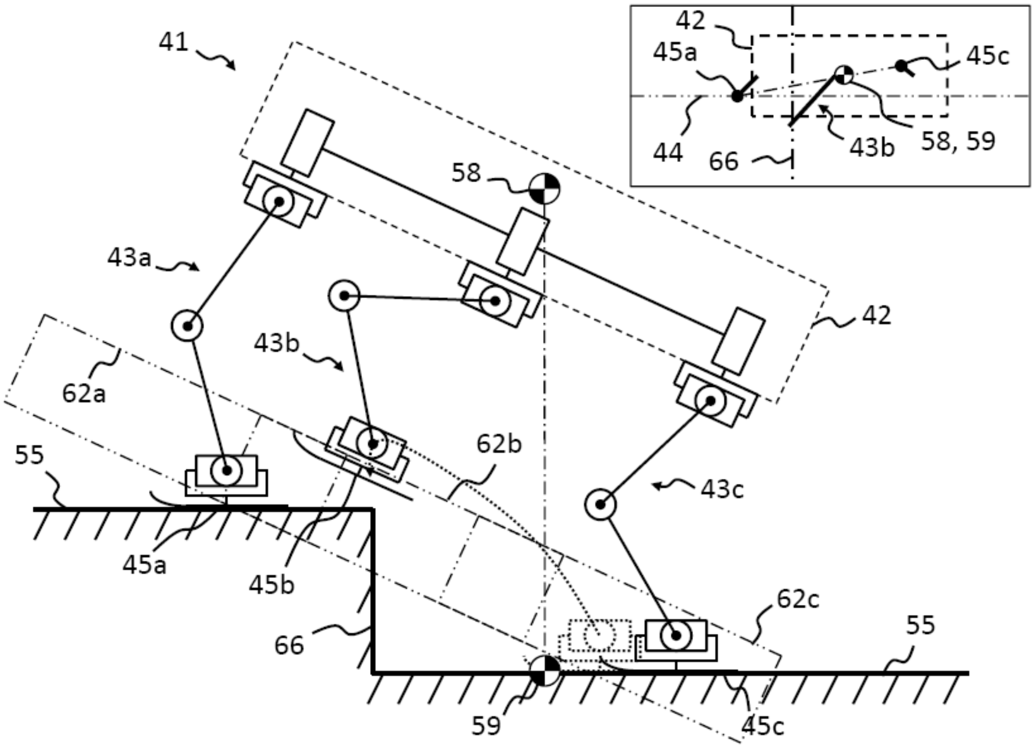


Figure 4.22. The single track or in-line legged mobile robot shown moving forward and shifting the center of pressure to along the zero moment line that bisects the centers of the front foot and rear foot to afford the legged mobile robot to lift the middle foot while maintaining stability of balance in a biped stance.

Figure 4.23 then shows the middle foot contacting the ground, and the ST3LMR reestablishing the triangular three-point contact support pattern. At this time, any dynamic instability arising from, for example, measurement errors or external dynamic forces, acting on the two-leg, bipedal stance are counteracted or reset by the more supportive tripod stance.

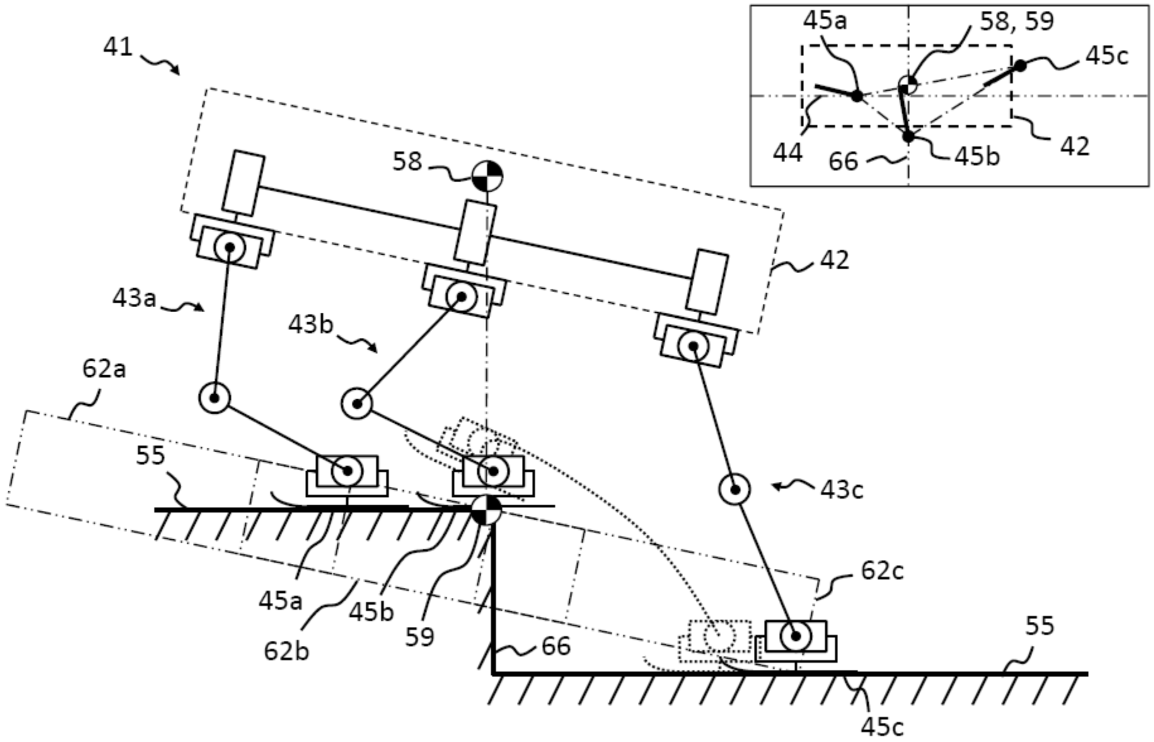


Figure 4.23. When the middle foot contacts the ground, the single track tri-legged mobile robot reestablishes the triangular three-point contact support pattern.

Next, Figure 4.24 shows the ST3LMR moving forward and shifting the center of pressure to along the zero moment line that bisects the centers of the front foot and middle foot to afford the legged mobile robot to lift the rear foot while maintaining stability of balance by the front leg and middle leg in a biped stance. The body is rotated to a level posture while the rear foot is simultaneously positioned above the upper ground and forward of the vertical or step gradient.

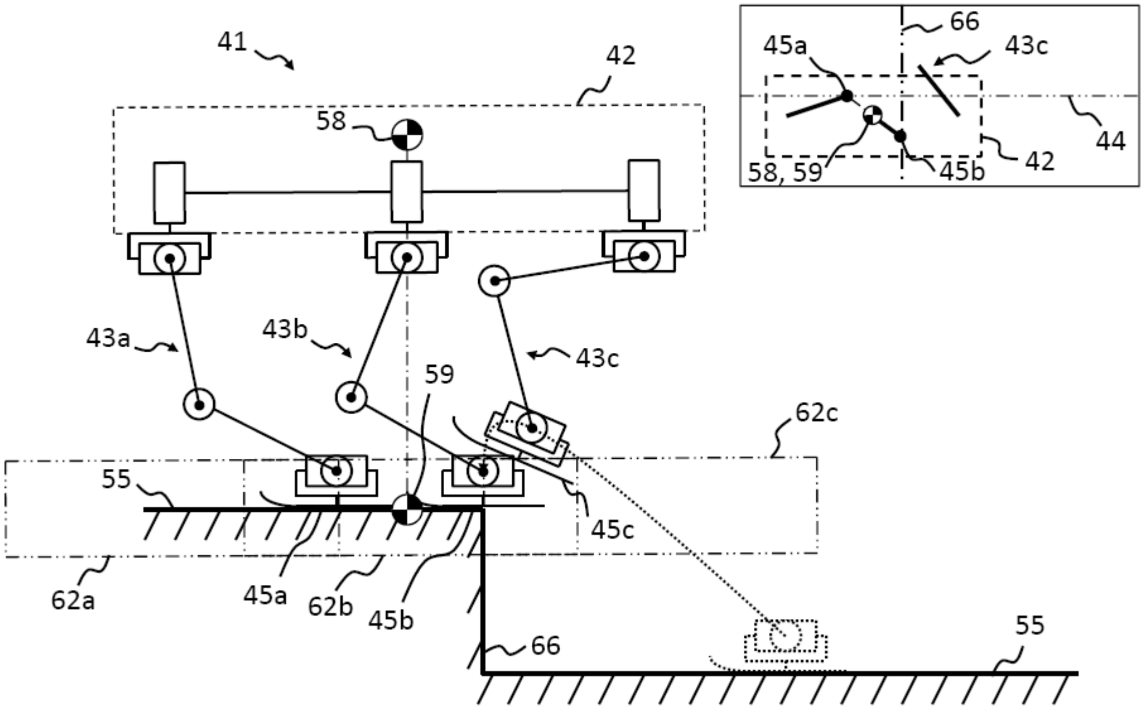


Figure 4.24. The single track tri-legged mobile robot shown moving forward and shifting the center of pressure to along the zero moment line that bisects the centers of the front foot and middle foot to afford the legged mobile robot to lift the rear foot, while maintaining stability of balance in a biped stance.

Figure 4.25 then shows the rear foot contacting the ground, and the ST3LMR reestablishing the triangular three-point contact support pattern. At this time, any dynamic instability arising from, for example, measurement errors or external dynamic forces, acting on the two-leg, bipedal stance are counteracted or reset by the more supportive tripod stance.

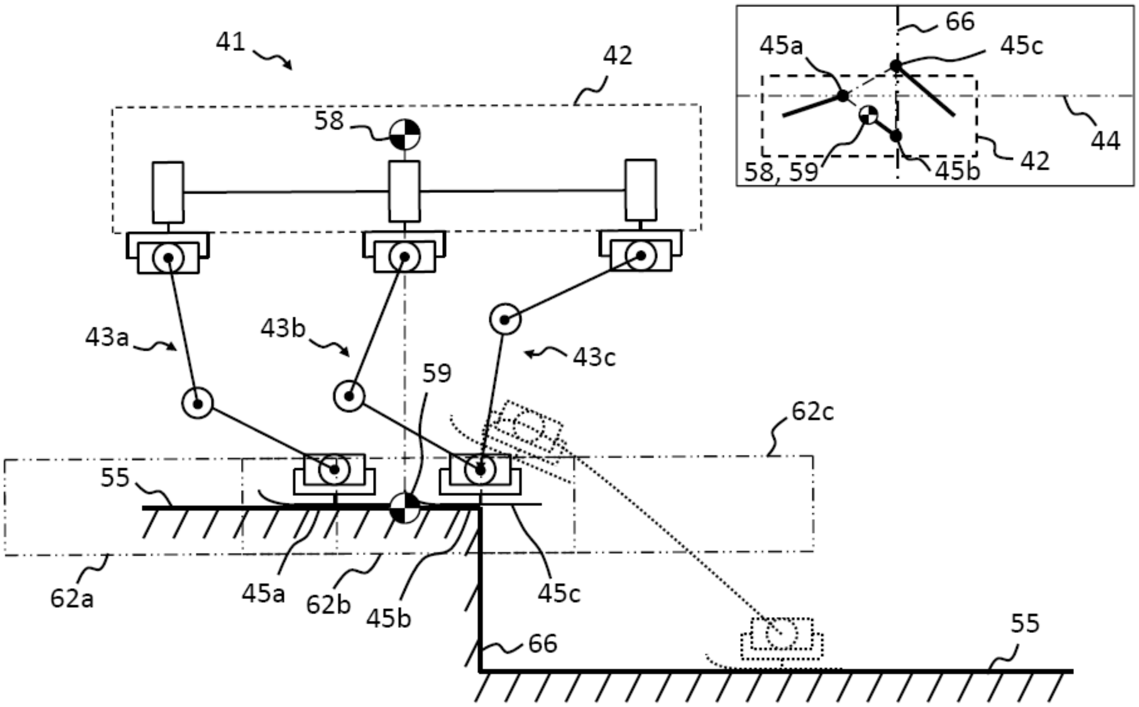


Figure 4.25. When the rear foot contacts the ground, the single track tri-legged mobile robot reestablishes the triangular three-point contact support pattern.

Finally, Figure 4.26 shows the lifting of the body and repositioning of the middle leg in preparation to begin a walking cycle on the upper ground. The body is shifted such that the center of pressure is repositioned to be along the zero moment line that bisects the centers of the front foot and rear foot to afford the legged mobile robot to lift the middle foot while maintaining stability of balance by the front leg and rear leg in a biped stance. The robot would then shift the center of pressure over to along the zero moment line that bisects the centers of the middle foot and rear foot and reposition the front foot to the forward most position while simultaneously moving the body forward. Forward motion is then maintained by any number of gaits, such as for example a backward wave gait, discussed in Section 4.1.

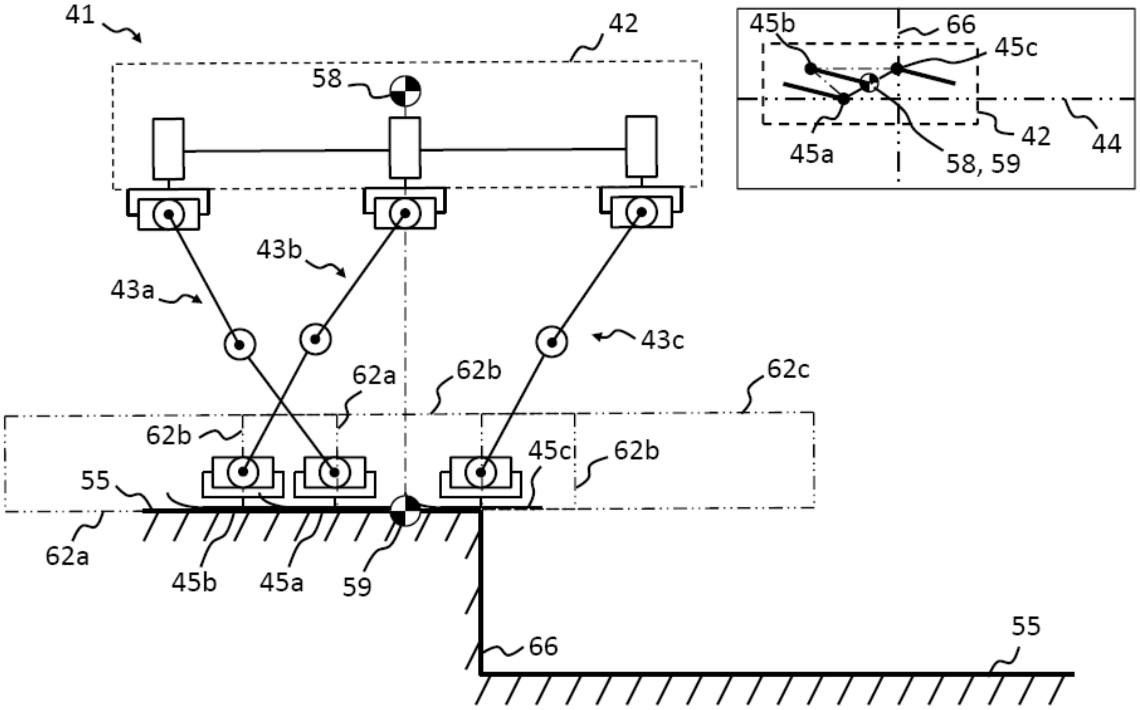


Figure 4.26. The lifting of the body and repositioning of the middle leg in preparation to begin a walking cycle.

Chapters 5 through 7 continue the discussion of foot, leg, and motion planning by first introducing a control strategy and architecture for single-track robots and then describing a probabilistic multi-hypothesis foot position planning system.

CHAPTER 5

SINGLE-TRACK MOTION PLANNING

In Chapter 3, the mechanical arrangement of the Single-Track Three Legged Mobile Robot (ST3LMR) legs is shown to achieve stability of balance without motion by positioning the three legs in a tripod stance. Recall, the term ‘single track’ shall be interpreted as referring to the general narrowness of the foot-placement patterns along a straight or curved path. In Chapter 4, a single-track gait strategy is proposed that keeps two legs on the ground at all times and decouples control in the pitch and roll axes. This chapter introduces single-track motion and foot placement planning, discussing fundamental concepts needed for a model-based Monte-Carlo predictive planning control system. To this end, the ST3LMR achieves the form and function of a motorcycle with the added benefit of legs and partial or fully automatic stability of balance (like a horse).

5.1. Inverted Pendulum Systems

Invented circa 1819, the first single-track vehicle was the two wheeled Drais Lauf-maschine or Draisine, a pedal-less precursor to the bicycle (Drais 1819 and 1832). Single-track vehicles, such as the Draisine and motorcycle, are inherently unstable and will fall over without some type of active control mechanism to contain roll accelerations. Like the Draisine, the ST3LMR must control stability of balance in the roll direction, while relying on leg coordination to keep two feet on the ground at all times. Figure 5.1 shows a robotic Draisine developed by this author and described in detail in Chapter 8.

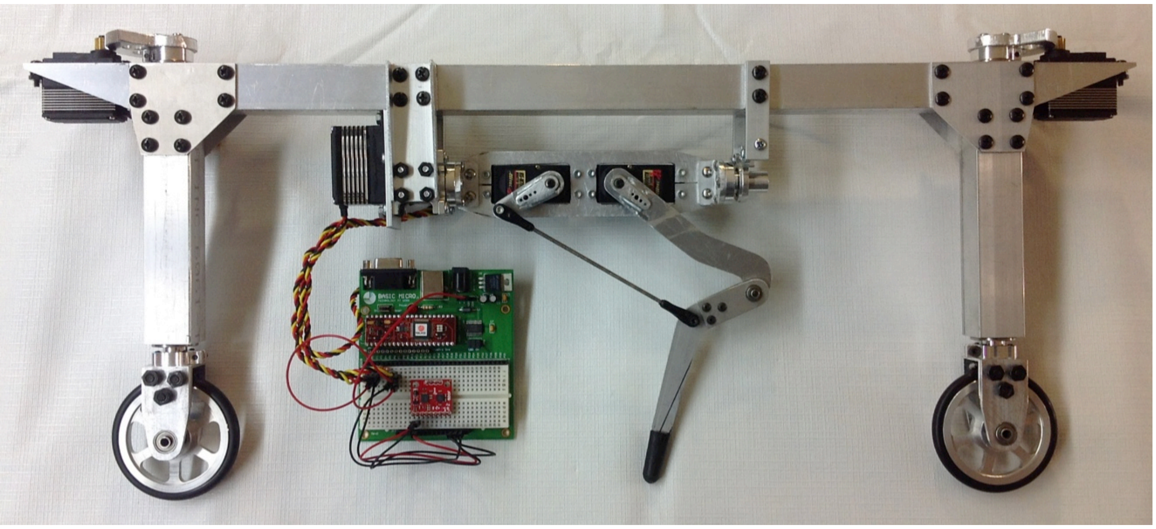


Figure 5.1. Draisine robot prototype developed to test single-track motion control theories for the ST3LMR.

About 1869, Rankine described that a bicycle is primarily righted by the lateral acceleration of the support line due to steering (Rankine 1869). Later in 1885, Harvard physicist Charles Warring compared the problem of bicycle riding to ‘broomstick balancing’ by translation of the support point back under the center of mass (Warring 1891). Unlike wheeled vehicles where the wheels are steered along a curved path to reposition the support points, a single-track legged vehicle must reposition one or more supporting feet laterally to reposition the support line. As a single-track legged vehicle moves forward (or sideways), a sequence of discontinuous support lines are developed. Because individual legs follow a stance-flight-stance cycle (discussed in Chapter 4), typically of long stance duration, instantaneous correction of the support line (like a rolling wheel) is not possible. Rather, single-track vehicle motion and specifically foot placement must be planned in advance to maintain upright balance.

In 1982, Marc Raibert studied dynamically unstable legged mobile robots, starting with a one-legged hopping robot (Raibert 1986). Raibert developed fundamental control algorithms for determining foot position (Eqn. 5.1) and hip angle (Eqn. 5.2) to control acceleration, velocity, and balance (adapted to the ST3LMR). Figure 5.2 illustrates a one-legged hopping robot modeled as an ideal inverted pendulum system with center of mass at height z, traveling from left to right, and extending a massless leg of length L to the ground in the start of a stance phase.

$$y_f = \frac{\dot{y}T_S}{2} + k_{\dot{y}}(\dot{y} - \dot{y}_d) \tag{5.1}$$

$$\gamma_d = \phi - arcsin\left({}^{y_f}/_{L}\right) \tag{5.2}$$

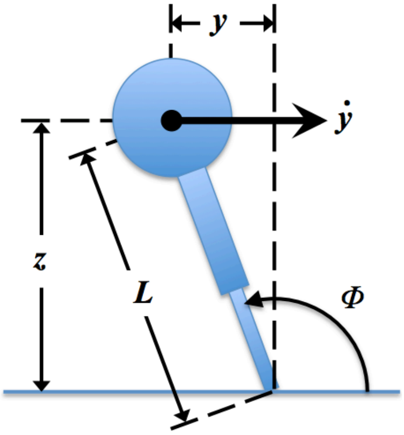


Figure 5.2. One-legged hopping robot modeled as an inverted pendulum system with center of mass at height z, traveling from left to right, and extending massless leg of length L. (Adapted from Figure 2.5 in Raibert 1986.)

In equations 5.1 and 5.2, y_f is the position of the foot, $\dot{y}$ is the lateral speed, T_s is the duration of the stance phase, $k_{\dot{y}}$ is a feedback gain, $\dot{y}_d$ is the desired lateral speed, γ_d is the angle between the leg and the body, Φ is the pitch angle of the frame, and L is the expected length of the leg during stance. Raibert used high-pressure, fast-acting servo hydraulics to quickly move the leg to a new position to regain balance (Raibert 1986). We shall return to the aforementioned control algorithms later in this chapter.

5.1.1. ST3LMR Inverted Pendulum Model

Consider the ST3LMR, shown in Figure 5.3, as an inverted pendulum system. Briefly, the ST3LMR includes a frame, wherein the frame includes a major axis corresponding to and generally parallel to a forward/backward direction of travel. Three jointed leg mechanisms attach to the frame, one behind the other, wherein each leg is attached at its proximal end at one or more discrete attachment points. The attachment points are arranged substantially parallel to the major axis of the frame and the forward/backward direction of travel. Each of the legs includes actuators attached between the legs and the frame and between adjacent leg members. The legs are actuated for movement of the distal end in three dimensions, wherein forward/backward movement of the legged vehicle is according to approximately single-track foot placement.

In Figure 5.3 left, the legged mobile robot with a rider (semi-autonomous mode) or passenger (fully autonomous mode) and center of gravity is designated 101 and 102, respectively. The rider is shown decoupled in two different subsystems. The first of these systems is an inverted pendulum system and the second is the ST3LMR. A free body diagram of a simplified model is shown on the right-hand side of Figure 5.3.

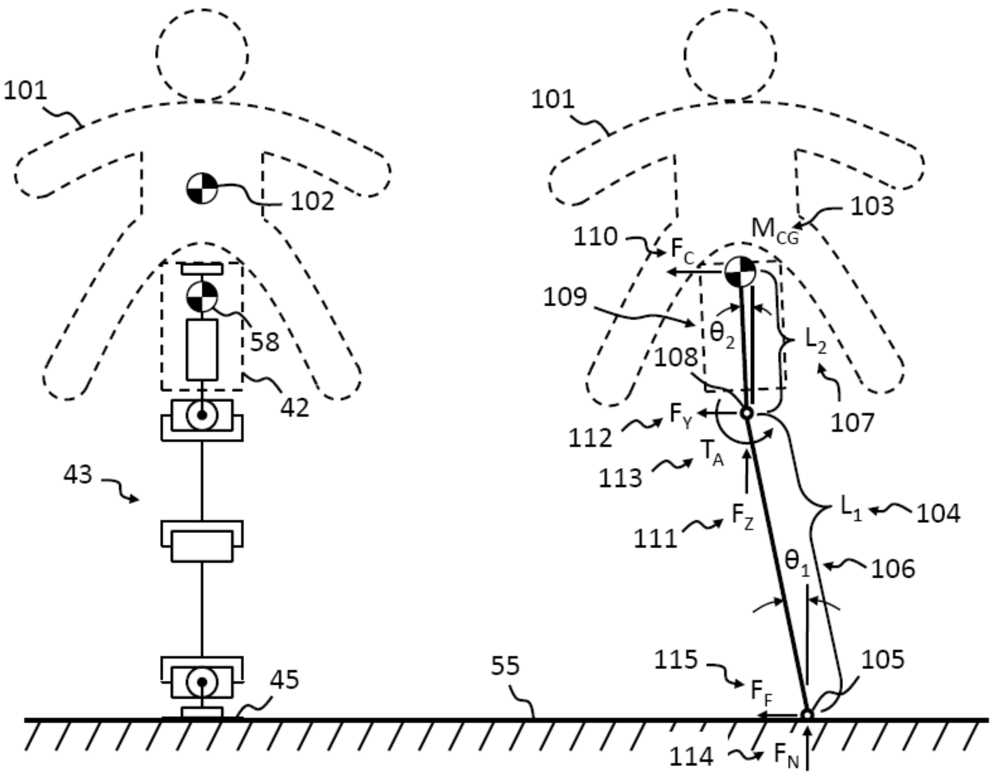


Figure 5.3. Lateral cross section view of the Single-Track Three Legged Mobile Robot where the left-hand illustration depicts the robot with rider and the right-hand illustration shows the resulting free body diagram. (From US patent 8,457,830 Fig 33.)

An inverted pendulum system is inherently an unstable system. However, an actively controlled ST3LMR is neutrally stable, and it is assumed that pitch and yaw motion is controlled separately and for different control objectives than upright equilibrium or balance. It is also assumed that the feet are always in contact with the ground and that no slip exists.

Only the latitudinal motion along the y-axis (perpendicular to the direction of travel, x-axis) and roll is considered. The latitudinal motion of the ST3LMR body is characterized by the projection of the center of gravity to the ground, called the center of pressure. (See also 55, 58, and 59 in Figure 3.1.) It is measured by the leg (proprioceptive) joint angles and by the filtered rate gyro and accelerometer system. The roll motion is characterized by the tilt angle, angle rate, and angle acceleration.

The fundamental inverted pendulum model for the ST3LMR is a point of mass, m, and three mass-less legs, coupling the mass to a point on the ground. For clarity of presentation, Figure 5.3 illustrates only one active leg. Each leg is modeled by an axially elastic, laterally rigid linear spring with a force-free length and spring constant. The inverted pendulum model is an equivalent rigid link and does not contain individual leg segments and joints, as previously described in Chapter 3. Locomotion dynamics occur in the sagittal plane with balance along the roll axis in the y-z plane.

Briefly from Chapter 4, a full stride comprises a stance phase followed by a flight phase. The stance phase for each leg begins after flight phase when the leg Touches Down (TD) to the ground and is elastically compressed so as to carry the weight of the body and vertical momentum force. The point of foot contact is modeled as a moment-free pin joint. Under normal environmental conditions, the foot remains fixed for the duration of the stance phase. The body moves forward and the leg rotates under the body. The amount of weight of the body and vertical momentum force carried by the leg is dynamic and determined by interaction of any other supporting legs and perturbations from the rider or passenger. The stance phase ends when the leg is Lifted Off (LO) the ground, and this begins the flight phase. During the flight phase, the leg is moved to reposition the foot ahead, with respect to the body, for the next stance phase.

5.1.2. Catching A Fall

Turnover of the ST3LMR can take place when the feet slide or slip sideways or laterally to a point that a normal target walking pattern cannot reestablish traction. A motion planning system must automatically sense and prevent turnover by: 1) extending at least one leg in the direction of roll to catch the fall, 2) advancing the leg in the expected direction of motion (and solving the foothold problem, discussed in Chapter 7), and 3) correcting the walking pattern in mid-step or in the next step. For partial automation, catching a fall is a highly important control action.

Figure 5.4 is a front skeletal view of the ST3LMR illustrating a leg being extended in the direction of a fall to catch the fall. The ST3LMR is shown with at least one foot in contact with the ground after the feet have started to lose traction. The IMU combined with the foot force and torque sensor measure uncontrolled foot slip. The IMU measures the rate of body roll, and the foot force and torque sensor measure traction or lack thereof. As foot slip continues to increase, traction will approach zero and the rate of roll will increase measurably. The control unit receives continuous measurement data from the sensors and determines if data inputs of slip and rate of roll are higher than achievable when feet have lateral traction with the ground. Corrective action is accomplished by controlling the joint actuations to deliberately (and quickly) shift a foot away from a single-track walking pattern towards the direction of fall, as show in Figure 5.4. Equations 5.1 and 5.2 govern the distance the foot must be shifted. A desired counteracting moment brings the ST3LMR upright.

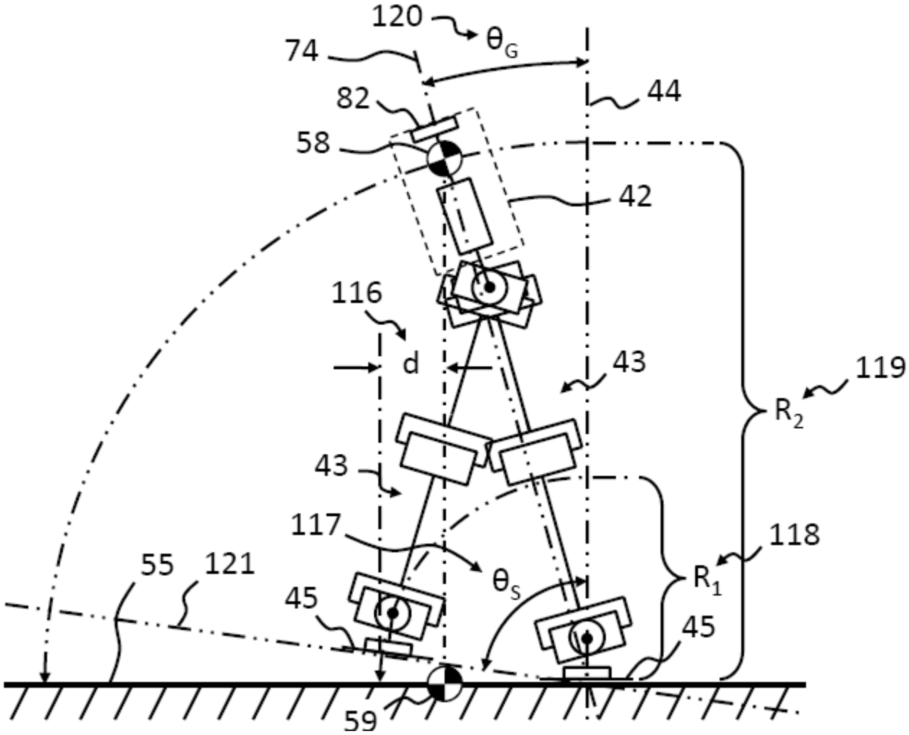


Figure 5.4. Front view of the single-track three legged mobile robot extending a leg in the direction of roll to catch its fall. (From US patent 8,457,830 Fig 34.)

In more detail and referring again to Figure 5.4, the initial condition is when the center of gravity is above the supporting point of the foot. The actuator encoders measure leg joint angles, and the IMU measures roll rate and acceleration from the vertical reference plane. The control unit calculates the criticality of roll rate by comparing the IMU measurements in a look-up table. Figure 5.4 is a simplified front view skeletal diagram illustrating the movement of the roll axis as foot traction approaches zero. In particular, the ST3LMR is shown in a tilted position. The traction of the feet may approach zero when: 1) the coefficient of friction force changes from static (high coefficient) to kinetic (low coefficient) due to a slippery roadway or loose road surface; 2) some or all of the vehicle weight is in a free-fall state; and 3) the polar moment of inertia moves to the center of mass. Once traction is lost, the roll axis moves from the foot to the center of mass. When the ST3LMR is rolling, the load transfers from the feet to the free-falling center of mass (roll axes) and the normal force on the feet decreases to zero. The ST3LMR rolls at an increased rate as the polar moment of inertia moves to the center of mass.

Regardless of foot slip, body trajectory is computed as an inverted pendulum problem and at least one leg, called the swing leg, is extended in the direction of the fall. Ideally, the swing leg is a leg already in flight or near flight phase, or it may be the leg contributing least to the expected stability of the body. Motion planning for this swing leg involves controlling two parameters. First, Figure 5.4 shows that the center of gravity trajectory is expressed as an inverted pendulum whose leg length is constant and thus defines an arc of radius R2. The center of gravity moves in a circular orbit about the supporting foot. The projected center of pressure and zero moment point shifts in the direction of the fall. Second, the expected moment of inertia of the body is calculated for the future time of when the fall would be caught, and a torque is computed to counteract the fall, which then computes the distance, d, required from the projected center of pressure and the swing leg arc of radius R1. The placement of the foot is adjusted to position the foot beyond the projected center of gravity and in the direction of the roll, using Equations 5.1 and 5.2. As a result, a counteracting moment can be induced to obtain a large attitude restoring force, to catch the fall and prevent vehicle overturn.

Simultaneously, the gate pattern may be adjusted for the other legs, such that in mid-step or in following footsteps, the walking gate is restored.

As the outstretched foot touches the ground and body roll is stopped, the IMU data and joint angle data is monitored by the motion-planning unit. While the outstretched leg and foot keeps the legged mobile robot from overturning or lying down on its side, it may be disadvantageous (to the rider) to immediately force the body into an upright position. For semi-autonomous control, the rider may regain control of the vehicle if it is held at an attitude very close to that at which control was originally lost. Perhaps after a predetermined pause to allow the rider to regain control, the legs and feet are automatically repositioned to raise the legged mobile robot to the fully upright position. It may also be advantageous, depending on the environmental circumstances (e.g., input from the IMU, joint angle sensors, and high-level rider commands), for motion planning to slow or stop all vehicle motion and transition the legs to a stable tripod stance, such that all three feet are in contact with the ground, but not necessarily with equal force.

On the other hand, if the maximum leg reach given the actuation time and time of fall is calculated to be less than required, it is not possible to obtain a righting force and the robot will fall over. In this case, it may be advantageous for motion planning to reposition the legs to a safe posture to prevent damage of the ST3LMR and/or the rider or passenger. Other than for these conditions, the robot would only signal to reposition the leg far laterally in emergency situations. Thus, a cautious rider may never lose lateral traction in which case the control system shall not intervene.

Finally, it is noted that catching a fall could be based on or adaptable to rider experience, e.g., uncontrollable situations for a novice rider or passenger may be controllable by an expert rider. This topic and general rider control and stabilization of single-track legged mobile robots is discussed in Chapter 10, the Future Work.

5.1.3. Planning Over Time

There are two general classes of motion planning problems for unstable legged robots (Raibert 1984; and Latombe 1991). The first class concerns slow body velocity problems where the kinetic energy due to robot forward velocity is of little concern. The second class must consider and/or rely on dynamic momentum of the robot and rider. While a purely reactive motion planning system would work well at slow body velocity to catch a fall, such systems would not provide stability of balance for partial or full robotic control under dynamic conditions, such as for example, unexpected forces imparted to the body by the rider or rugged and loose terrain. An improved model-based temporal planning system is needed for discrete foot placement planning to achieve stability of balance over time.

In Chapter 3, it was shown that robot motion planning uses a gait model as the basis for coordinating the movement of the various legs. The gait model describes the position of the legs and feet over time. Individual leg positions are converted to a time-based sequence of desired actuator position or rate commands (joint drive patterns). Because the ST3LMR is unstable in the roll axis (and the robot will fall over), feedback from leg joint angles, foot force and torque sensors, and roll and yaw angle and angle rates from the IMU sensor are used in motion planning. When executing a gait, the ST3LMR must balance so it does not roll over. For example, estimating the roll angle at a future foot TD time and calculating the shift of ground reaction force in the direction to which the attitude is restored, is the basis of temporal motion planning. Second, the ST3LMR must follow a desired trajectory. The trajectory may come from the rider or the path planning system.

5.2. Predictive Systems

In 1963 the concept of a predictive control system, using a high-speed model, was formulated to replace man-in-the-loop systems (Johnson 1965). A predictive system estimates the future state, such as roll angle and angle rate, given a device model and sensory input. For example, Raibert's Equations 5.1 and 5.2 provide a predictive 2D model that is used to control a hopping robot in 3D (Raibert 1984). About this same time, Boolean logic was applied to control paradigms to predict future events. The concept of a subsumption architecture for path planning and motion control of robots was developed (Brooks 1986). The subsumption architecture provides a method for structuring motion from the bottom up using layered sets of rules. Further developments in predictive path planning systems combined expert system methods with models to estimate systems in dynamic environments (Fiorini and Shiller 1993). More particularly, Fiorini and Shiller developed an algorithm whereby assigning an appropriate cost to each decision branch that a robot path may traverse, an appropriate objective function can be maximized or minimized to calculate a robot trajectory. Recently and with the advent of the high-speed digital computer, Monte-Carlo methods have become a popular means to simulate multi-hypothesis predictive systems (Elfes 1989; Borenstein and Koren 1991; Philippsen and Siegwart 2004; Spalanzani et al. 2008; Tay et al. 2008; and Fulgenzi 2009). Nevertheless, predictive path planning for unstable legged robots, and more specifically predictive foot placement planning for maneuverability, obstacle avoidance, and to maintain stability of balance, is still an open research area.

5.2.1. Side-Step Planning

Predictive foot placement planning for stabilization control requires estimating body attitude over time, and Raibert's method of determining future foot position of the one-legged hopping robot (Equations 5.1 and 5.2) is adapted to control the ST3LMR foot position in the roll axis. This device and method is highly important because a plurality of legs along a single track or in-line roll axis affords a higher probability of achieving dynamic balance in the next step, over traditional legged locomotion. Referring to Figure 5.5, if an inverted pendulum (robot) is falling clockwise to the right (at time t_0), for example, it can rebalance or upright itself by discretely (and quickly) moving a second supporting foot to a new position located to the right (at time t_1). This position is called the neutral point (Case 2). The kinetic energy of the right-falling inverted pendulum is completely consumed (or balanced) by potential energy as the inverted pendulum rises along an arc (at time t_2), given a fixed leg length. If the foot is repositioned behind the neutral point (Case 1), not all of the kinetic energy is consumed, and the inverted pendulum continues to fall to the right, but now with increased acceleration. If the foot is repositioned forward of the neutral point (Case 3), all of the kinetic energy is consumed before the inverted pendulum reaches the upright position. With the support point now to the right of the center of mass, gravity induces a torque about the foot, and the inverted pendulum now falls to the left and the inverted pendulum (robot) falls over.

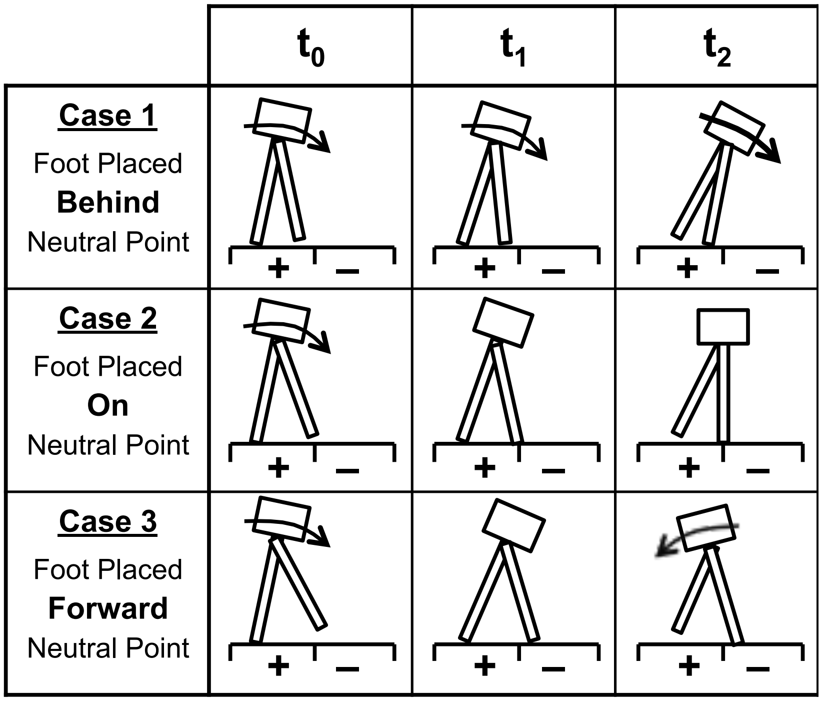


Figure 5.5. Three side-stepping cases/hypothesis for the ST3LMR, illustrated graphically in a front view. (Adopted from Figures 2.10 and 2.11 in Raibert 1986.)

Predictive foot placement planning to implement a rider's turn commands or to avoid obstacles while simultaneously maintaining (and/or recovering) balance in the roll axis generally requires planning the foot positions many steps in advance. Given the above 3-case (multi-hypothesis) framework for positioning a leg in the cross-axis, a

Monte-Carlo tree search method for predicting foot position planning is devised. Figure 5.6 illustrates a 3-case multi-hypothesis approach for the ST3LMR.

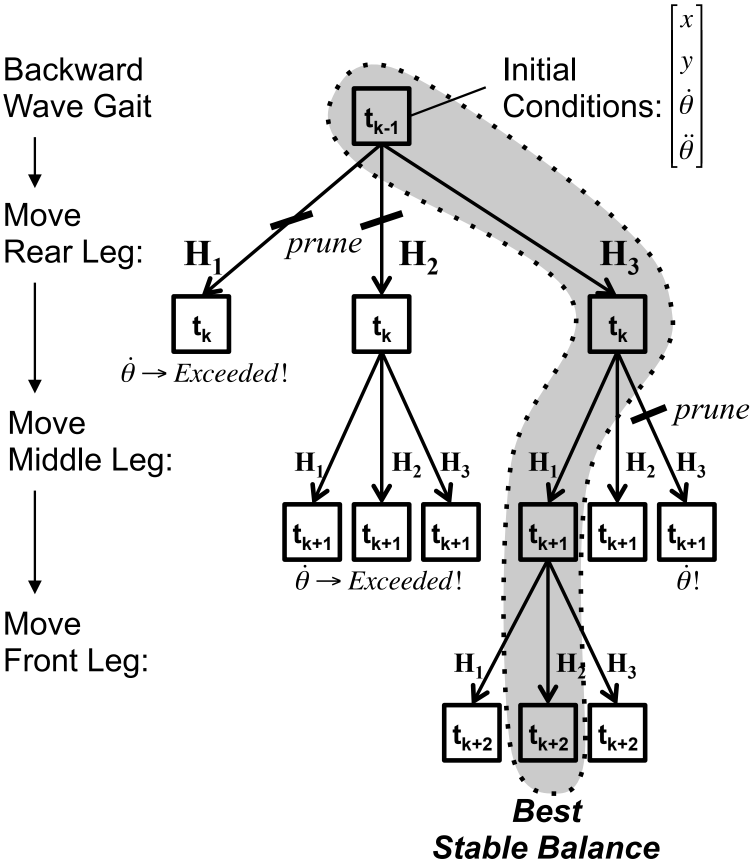

Backward Wave Gait
Move Rear Leg:
Move Middle Leg:
Move Front Leg:
Initial Conditions: $\begin{bmatrix} x \\ y \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}$
t_{k-1}
prune
H_1
H_2
H_3
t_k
$\dot{\theta} \rightarrow Exceeded!$
prune
t_{k+1}
$\dot{\theta} \rightarrow Exceeded!$
$\dot{\theta}!$
t_{k+2}
Best Stable Balance

Figure 5.6. Multi-hypothesis planning using branch-based scoring in a depth-limited search.

Referring to Figure 5.6, the top node at time t_{k-1} of the tree presents the initial conditions (positions of the feet, roll velocity, roll acceleration, and the like) of the robot. At time t_k, the robot moves the rear leg to a new position, as part of a backward wave gait. There are three options or hypothesis about positioning the leg laterally (the 3-cases: behind, on, or forward of the neutral point). If the leg/foot is positioned according to the first hypothesis, H_1, the roll velocity is increased and the robot will fall over. If the leg/foot is positioned according to the second or third hypothesis, H_2 or H_3, the robot does not fall over. The robot state is updated for time t_k (based on the initial conditions and a temporal inverted pendulum model). Then the middle leg is moved at time t_{k+1}. Given the robot state from H_2, neither of the three lateral leg positions would contain the roll, and the robot falls over. Given the robot state from H_3, the robot does not fall over. This multi-hypothesis test continues until a desired future state is reached, as depicted by the gray shaded branch paths in Figure 5.6. Note that if all hypothesis result in the robot falling over, the hypothesis are re-computed using a different feedback gain, $k_{y\text{-dot}}$. Solution refinement is an iterative process of making incremental positive or negative adjustments to the feedback gain, $k_{y\text{-dot}}$, to maximize stability of balance. When there is no gain that produces a stable future state, the robot is in an out of control situation. Recovery from an out of control situation is beyond the scope of this dissertation.

5.2.2. High Speed Model

Figure 5.7 illustrates a block diagram of the model-based control system developed in this work. Starting with the Robot box in the upper left of Figure 5.7, the initial conditions or states of the robot (e.g., leg lengths, hip angles, body roll angle, body roll angle rate, and so on) are input to the High-Speed Model (HSM). The HSM performs the above-described multi-hypothesis planning using branch-based scoring in a depth-limited search. The foot trajectories (and their timing) that result in the best stable balance are then input to the leg State Machine (lower left). The HSM then (or it waits until the next sensor time cycle) inputs new data, recomputes the trajectory that provides best stable balance, and updates the results input to the State Machine. When the State Machine determines a leg is ready to transition from one state to another (e.g., stance to flight), the State Machine reads the latest foot trajectory planning data from the HSM, and implements it. All three legs are moved, providing a new set of initial conditions to the HSM. Additionally, Gait Coordination (e.g., backward wave gait) and velocity-based timing information is input to the HSM to structure the possible leg and foot actions.

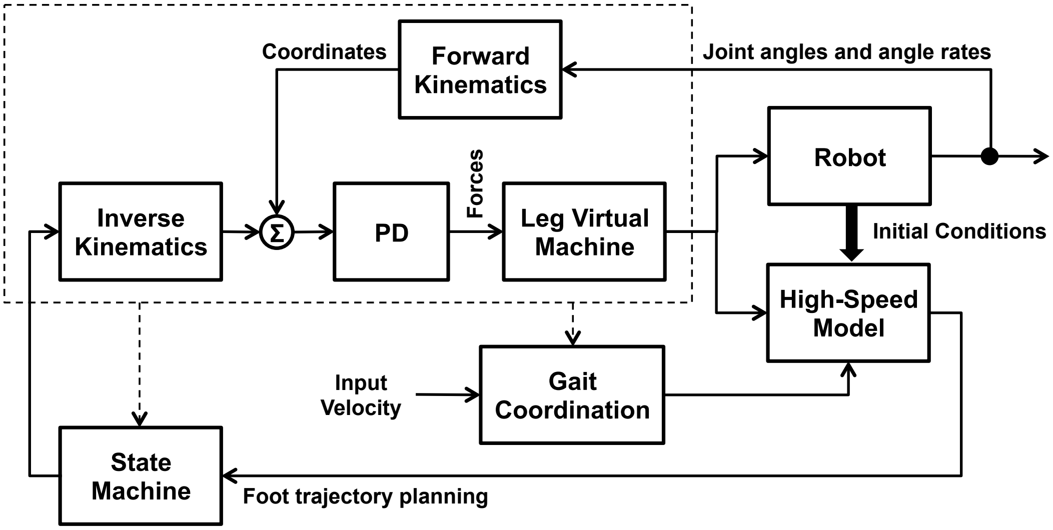


Figure 5.7. Gate coordinated leg control system using a high-speed model for foot trajectory planning.

5.3. Digital Simulation

A simple digital simulation was written in MATLAB to demonstrate feasibility of the ST3LMR design concept. The annotated MATLAB code is found in Appendix A. In this simulation, the robot (height, weight, actuator performance, etc.) was modeled after

Boston Dynamics' BigDog quadruped robot, as a first approximation (Raibert 2008; and Holste 2009). Table 5.1 summarizes the robot specifications found in the literature.

Table 5.1. Boston Dynamics' BigDog quadruped robot specifications.

Size	Length 5.17 feet (2.44m)	COG Height 2.5 feet (1.0m)
Weight	240 pounds (110kg)	
Speed	Flat road 30 mph (48kph)	Rugged terrain 4 mph (6.4kph)
Capacity	Carrying 340 pounds (150kg)	Climbing 35-degrees incline
Actuation	3,000psi @ 25gpm	(9x) Two-Stage Electro-Hydraulic Servo

5.3.1. Simulation Setup

A number of steps are implemented to control and coordinate leg/foot motion, including: a) collecting sensor data; b) associating the collected sensor data into sets of observations; c) storing and maintaining a real-time database of sensed data and robot state; d) estimating the future position, state and their covariance of tracked footholds/ obstacles based on past positions, states, and covariance; e) estimating multiple hypothesis of the future leg/foot positions, robot states and their covariance; and f) scoring, ranking, and selecting the leg/foot hypothesis with respect to the desired operator input. Figure 5.7 illustrates the control system developed in this work.

For this work, only the backward wave gait (Chapter 4) was implemented, and all environmental conditions were ideal, e.g., flat ground, perfect sensing, perfect actuation, and so on. All possible future foot positions are estimated using the branching multiple hypothesis temporal simulation that uses an inverted pendulum model for the robot. The feedback gain, $k_{y\text{-dot}}$ (Equation 5.1), is iteratively adjusted, from one to the maximum working reach of the leg, until a stable-balance solution (minimizing roll rate) is obtained. The results are ranked and a single best foot position plan is selected to maintain stability of roll. When a foot is ready to be moved, the next foot position is sent to the Leg State Machine. The individual leg joint angles are calculated using inverse kinematics, and the robot motion is simulated using individual PD control loops for each leg. For each time interval of the simulation, new initial conditions of the robot are determined using an inverted pendulum model.

A HSM estimates future robot states (foot positions, body roll angles and angle rates, etc.) and their covariance, using temporal-based equations of motion. The HSM operates at a much faster rate than the sensor data rate to perform the aforementioned multi-hypothesis planning, i.e., a two-time-scale system. In this manner, real-time data from sensors form the initial conditions for obstacle avoidance, trajectory planning, and stability of balance. At a desired path planning look-ahead time or state, the highest scoring path is selected as the optimal path (Figure 5.6).

A brief overall description of the trajectory generation follows, taking a leg and foot trajectory of the ST3LMR as an example. First, the aforesaid basic leg trajectory is generated in advance using a gait model. More specifically, the trajectory is determined using a robot centric coordinate system referenced to the body. The end point of a leg trajectory at the time of foot rise from a modeled virtual ground surface and referenced to the x-y-z coordinate system is computed. Then the next or successive stance trajectory for that leg is planned from the gait model, given high-level foot placement data, i.e., areas or regions of ground that are derived from sensed or a priori data and deemed safe to support the legged mobile robot. Based on a foot end and starting positions over time, from the gait model, a leg trajectory is calculated which connects the foot LO to foot TD. The trajectory may be modified to account for ground surface irregularities and obstacles. For example, a sensed obstacle is avoided by adding an additional ground clearance to the flight trajectory. As the body moves, the aforesaid coordinate system is displaced (translated and/or rotated).

The other functional elements of the control system are the typical Leg State Machine (support-to-flight-to-support cycle), Inverse Kinematics (converting desired foot positions to leg joint angles), Proportional-Derivative (PD) loops, and the Leg Virtual Machine (force- and compliance-based leg extension).

5.3.2. Simulation Results

Three experiments typical for determining motorcycle performance and maneuverability were performed. In the first experiment, a sideways force impulse is applied to the ST3LMR center of mass. One typical simulation result is presented in Figure 5.8. The force impulse imparts body roll with respect to the feet. The feet are repositioned to the left or right of the body to minimize roll rate, regain an upright pose, and bring the robot back to the original trajectory.

For all results, the horizontal axis is the major axis of motion, and it also corresponds to simulation time. The vertical axis is the lateral or cross-axis of motion. The stance position of the feet are large circles, while the smaller open circles represent the path of the foot during flight. The subscripted numbers denote the stance count (time) with respect to the initiation of the maneuver. The solid black line represents the center of mass. The dotted lines represent areas of support when all three legs are on the ground. Only a portion of the overall solution is shown in the graphs, for clarity.

The second experiment involved obstacle avoidance. One typical simulation result is presented in Figure 5.9. Through this experiment, it was found that repeating intervals of the tripod stance enabled controlled, sideways thrusting of the body without imparting uncontrollable body roll.

The third experiment was a controlled turn wherein the ST3LMR is leaned into the turn. Figure 5.10 shows the ST3LMR executing a single-track turn wherein the center of mass (black line) is spatially and angularly displaced from the single-track foot positions. This is called “leaning into the turn.” During this maneuver, the center of gravity and projected center of pressure is moved towards the center of curvature, thus developing a torque about the roll axis that counteracts the outward centripetal inertial force acting on the center of gravity of the body. In leaning into a turn, the feet are following a single track or in-line curve of radius about a center point normal to the ground. The top of the body thus follows a second curve of smaller radius about the projected center point normal to the ground such that the resulting plane of motion is a truncated cone. Through this experiment, it was found that the feet must follow a curved trajectory to torque the body in the yaw axis, along the single-track curve.

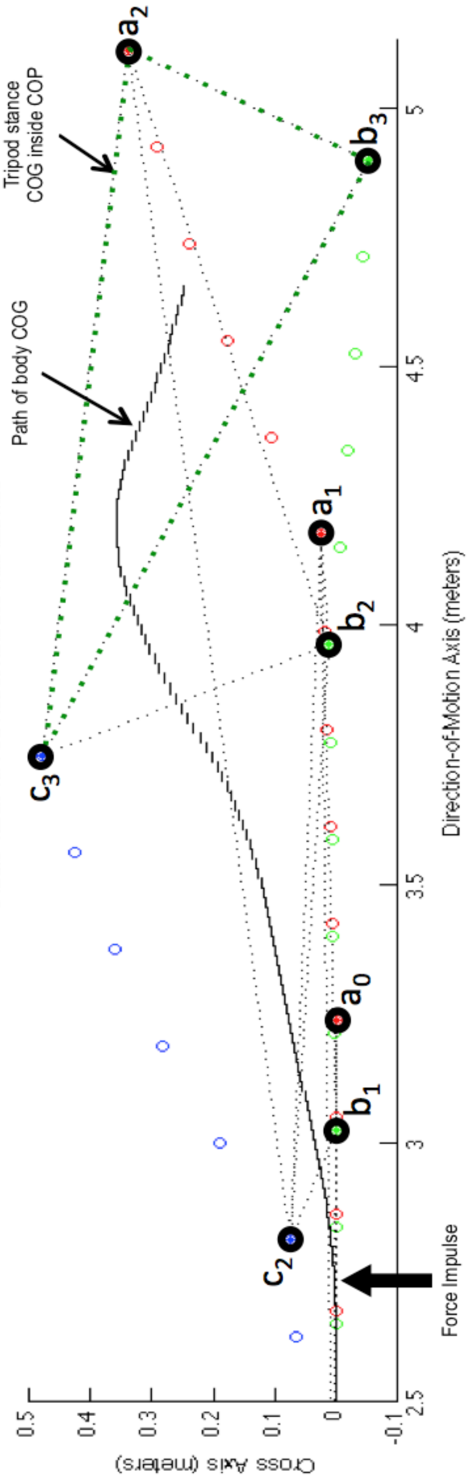


Figure 5.8. Simulated impulse response maneuver.

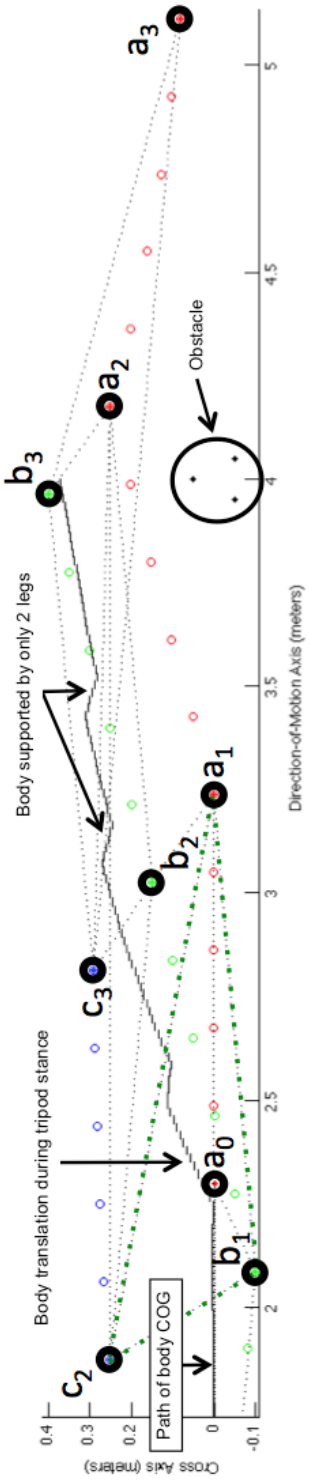


Figure 5.9. Simulated obstacle response maneuver.

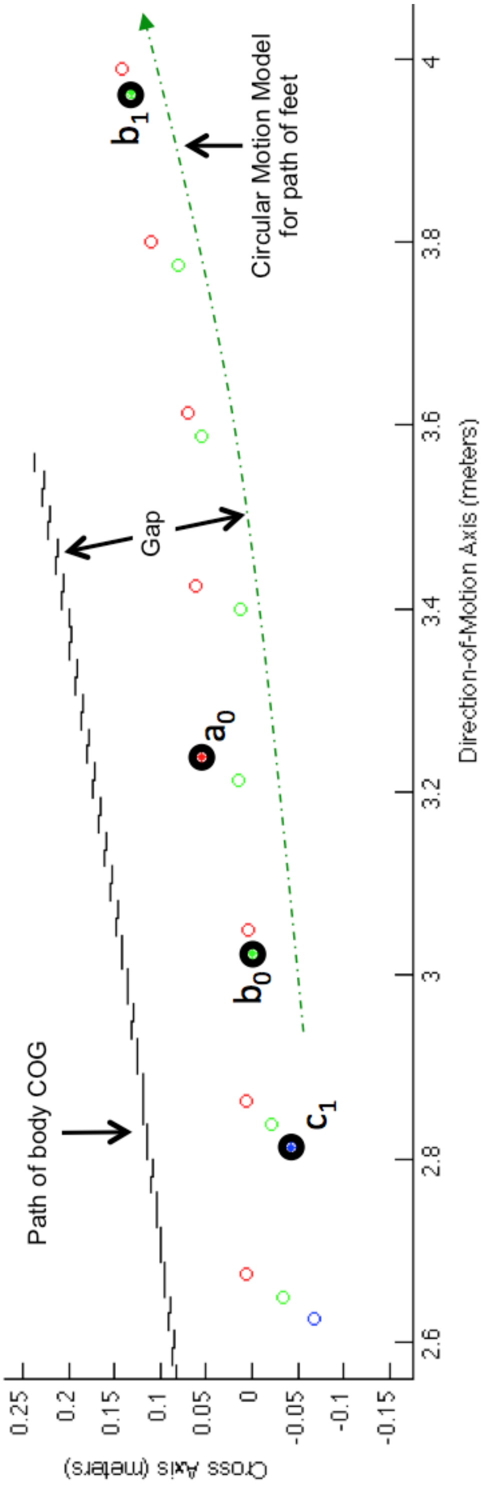


Figure 5.10. Simulated lean-into-turn maneuver.

In all cases, the foot branching temporal simulation found a solution to demonstrate the feasibility of the design. Again, this work is a proof-of-concept for the ST3LMR design concept. From the aforementioned discussion, it is necessary for the ST3LMR to have lateral dynamic balancing, controlling attitude on the basis of the detected inclinatory or roll angle, angular velocity, and angular acceleration of the body. An inherent physical property of the ST3LMR is decoupling of the leg positioning along the length of the body or major axis of motion and the leg positioning along the width of the body or normal to major axis of motion, given the appropriate gait (discussed in Chapter 4). During locomotion, the feet are dynamically placed to the left or right of the support line to maintain (regain) stability of balance. Especially in rough terrain, planning and coordinating foot placement with regard to cross-step distance, forward speed, body height, and duration of ground contact is key to balance stabilization during locomotion. The control system must also account for rider-induced perturbations (especially in the roll axis) and mechanical losses in the system. The degree to which the rider provides balance control of the roll axis (like a motorcycle) versus the robot providing stability of balance (like a horse) is the subject of future research. Chapter 6 introduces and discusses concepts of temporal control systems.

CHAPTER 6

TEMPORAL PLANNING AND CONTROL SYSTEMS

Chapter 5 introduced single-track motion and foot placement planning concepts. This chapter provides more details regarding temporal motion planning systems, including Bayesian methods for processing sensor data in real-time, Kalman filters, and temporal tracking systems, and two-time scale systems. In a temporal control system, data from real-time sensors (position, velocity, heading, joint angles, and so on) is stored and organized in a database. Sensed data provides foothold and obstacle data for the foot placement planning system.

6.1. Sensed Data

Human beings use multiple senses or cues to perceive upright orientation and maintain balance. The vestibular system (inner ear) senses angular rotation and linear velocity, the proprioceptive system (muscle stretch receptors) sense body posture, the mechanoreceptors (skin) sense pressure, and the visual system senses spatial orientation and body equilibrium (McCredie 2007). Accurate sensing of roll angles and angle rates is required to balance the ST3LMR. To date, most of the research to balance legged or inverted pendulum type mobile robots uses gyroscopes, accelerometers, tilt sensors, and potentiometer or encoder-based leg joint angle measurement (Bekey 2005). Very little research has used machine vision (optic flow, edge detection, correlation pattern matching, etc.) to balance mobile robots (Goulding 2010).

When a machine makes decisions by sensing its environment, it is called a robot (Lewis 2008). In this dissertation, controlling and coordinating robot foot placement generally involves six steps:

1) Collecting sensor data;

2) Associating the collected sensor data into sets of observations;

3) Storing and maintaining a real-time database of sensed data and robot state;

4) Estimating the future position and their covariance of tracked footholds and obstacles based on past positions, states, and covariance, which generally involves estimating the future position, state and covariance of the robot, leg, and foot based on past positions, states, and covariance;

5) Estimating multiple hypothesis of the future foot positions, robot states and their covariance, given a goal state of the leg and foot, including iterative feedback processes to refine the foot positions; and

6) Scoring, ranking, and selecting the *best* foot hypothesis with respect to multiple planning goals and objectives, including the desired operator input.

6.1.1. Representing Data As Footholds and Obstacles

For legged robot motion planning, sensor-based data may be one of two types: 1) preferred footholds and 2) obstacles. A preferred foothold, simply called a foothold, is a ground-based feature the robot may use to place a foot. In rugged terrain, a foothold may be an exposed, smooth rock, identified by a vision system through shape, color and texture. An obstacle is the opposite of a foothold. It is an area of ground that is to be avoided for foot placement. In rugged terrain, an obstacle may be a 3D volume, such as a tall post, or a 2D surface area, such as loose soil, also identified by a vision system through shape, color and texture. In the absence of either footholds or obstacles, the control system would assume the area is safe for foot placement. Generally speaking, image data is clustered or segmented into footholds or obstacles based on the minimum robot foothold size. Note that a full discussion of computer image processing is beyond the scope of this dissertation, but the topic is discussed in more detail in Chapter 10, Section 10.4, the Forward-Looking Sensing System.

Because the robot is in motion, sensor-based tracks are also in motion (with respect to a robot-centric coordinate system) and are updated using temporal equations of motion. To simulate the robot, a new track file is added to simulate the robot, and the sensor-based track files are duplicated from the core loop to enable the temporal simulation loop to introduce behaviors, to generate action hypothesis for the robot, and to provide probabilistic scoring method to optimize strings of action and state hypothesis over time.

6.1.2. Representing Data Uncertainty

To address uncertainties in predicted locations, beliefs on the level of certainty for each predicted location of the foothold can be estimated based on the current and past sensor data. These beliefs can be stored in the database and associated with their respective track files. For example, based on the most recent sensor data, the likelihood that the foothold will be located at each previously predicted location is evaluated and a weight factor assigned to each predicted location. Once the predicted locations are evaluated, each possible path that the foothold may follow (e.g., the locations through which the foothold will pass) can be given a score. The weight factor for each location in the path can be summed together to provide an overall score for the path, thereby providing an indication of whether or not the path is supported by the sensor data. If the path is supported by the sensor data, it validates the motion model used (as in a multi-hypothesis test) and aids in the classification of footholds. Once a foothold is classified, its location, state and their covariance may be predicted with more computational efficiency.

For example, when a detection-to-track file association is ambiguous (i.e., when the sampled data does not support or discredit the predicted position), decisions regarding the position can be delayed under the assumption that future data will resolve the ambiguity. This strategy is referred to as multiple hypothesis tracking, or MHT.

Dynamical systems change over time, and importance weighting the sensed or model-predicted state is key to robust control. Probabilistic methods are used to determine the relative importance of road/system preview data. Various algorithms process data organized in time-series, such as for example dynamic Bayesian networks, hidden Markov models (HMMs), and Kalman filters. In 1763, Thomas Bayes defined a conditional probability as the ratio formula:

$$P(x \mid c) = \frac{P(x, c)}{P(c)} \tag{6.1}$$

where c is called the context of the belief in x. For example, if A stands for the statement "The foothold is solid," then P(x|c) stands for the subjective belief in x given a body of knowledge c. The Bayes inversion formula:

$$P(x \mid \underline{u}) = \frac{P(\underline{u} \mid x)P(x)}{P(\underline{u})} \qquad (6.2)$$

is used to compute the belief in x given evidence u, which may include observed sensor data. P(x|u) is sometimes called the posterior probability or posterior, and P(x) is sometimes called the prior probability or prior. The denominator P($\underline{u}$) is a normalizing constant,

$$P(\underline{u}) = P(\underline{u}|x)P(x)+P(\underline{u}|{\sim}x)P({\sim}x) \qquad (6.3)$$

For example, the likelihood of the sensor evidence P($\underline{u}$|x) is multiplied by the expected target density P(x) and normalized to yield the conditional probability that "The foothold is solid given the texture-based sensor evidence."

Bayesian networks use probability theory to capture and formalize the structure of reasoning (causation, association, and relevance) through graphs, manipulated by logical propagation, to process conditional probabilities, given partial or uncertain information. Note that Bayesian networks differ from Markov networks in that a Markov network is an undirected graph with links defining symmetrical probabilistic dependencies, while the Bayesian network is a directed acyclic graph (DAG) with arrows representing causal or context-dependent probabilistic dependencies. Bayesian networks are DAGs in which the nodes represent variables, the arcs model causal relationships, and a weight or strength of the causal relationship is expressed in a conditional probability.

Because the nodes in a Bayesian network are not conditionally independent, the posterior is computed using parental evidence P(u|x) and the prior is computed using the belief in child nodes P(x|y) (Pearl 1988 and 2000), as

$$P(x \mid \underline{u}) = \eta P(\underline{u} \mid x) P(x) \tag{6.4}$$

and using λ(x) to represent the diagnostic or retrospective support from X's children and π(x) to represent the causal or predictive support from X's parents,

$$\begin{aligned} \lambda(x) &= P(\underline{u} \mid x) \\ \pi(x) &= P(x \mid \underline{y}) \end{aligned} \tag{6.5}$$

the belief in X simplifies to the normalized product of λπ messages,

$$BEL(x) = \eta \lambda(x) \pi(x) \tag{6.6}$$

Pearl's generic node update algorithm is applied to X. Note the addition of conditional probability matrices, MX|U and MU|X, with U the set of causal hypothesis and X the set of consequences or manifestations of the hypothesis.

The possible locations of a leg/foot over a period of time form a simple tree, wherein at time k-1, the foothold is at a known location. For example, the ST3LMR executes a backward wave gait. It moves the rear leg, then the middle leg, then the front leg, then the rear leg, and so on. There are three possible foot placements for each step. H_1 places the foot laterally to the left of the neutral point. H_2 places the foot on the neutral point, and H_3 places the foot to the right of the neutral point. Recall, the neutral point is the point of roll-axis equilibrium. The distance from the foot to the neutral point is determined by a gain factor. (See Chapter 5, Section 5.2.1 Side-Step Planning, for background information.)

Prior to the next sensor update, there may be uncertainty regarding the foothold/obstacle state of the next location hypothesis H_1, H_2, and H_3. During subsequent time intervals, the foothold may continue on a current track, or associate with new tracks based on various data. As a result, the temporal loop estimates that at time k the foothold is at a location with a given belief. Still before receiving the next sensor update, the temporal loop may predict the location of the foothold at the next time increment k+1. Since there is uncertainty of the location at time k, the temporal loop predicts the location at time k+1 for each of the locations H_1, H_2, and H_3. At time k, the temporal loop receives a sensor update, and then compares the sensor data to the predicted locations of the leg/foot. More particularly, each predicted location is given a score based on how close the predicted location corresponds to the sensed data. For example, locations that highly correspond to the foothold data can be scored as positive, locations that correspond to no sensor data can be scored as neutral, and locations that correspond to obstacle data can be scored as negative. In assigning a score to locations, a "cost" can be computed, wherein the association cost is Gaussian and expressed as a probability or likelihood of the goodness of fit. Equation 6.7 in Blackman & Popoli is of the form:

$$\frac{(Vc)e^{(-0.5d^2)}}{\left(\sqrt{|S|}\right)(2\pi)^{\left(M/_2\right)}} \tag{6.7}$$

where d^2 is the normalized statistical distance between the measurement and estimation, defined in terms of the measurement residual vector, y and covariance matrix S(d^2 = y'(S(-1))y), Vc is the measurement volume element, M is the measurement dimension, and |S| is the determinant of the measurement residual covariance (HPH'+R) (this is the

Blackman & Popoli goodness of fit equation mentioned above). A score of each predicted path then can be determined from the score assigned to each position that forms the path. These scores can then be used to evaluate which paths are more or less likely to be followed by the foothold, and thus to determine which paths and/or locations are purged from the database during track file maintenance.

6.1.3. Kalman Filter

Key to the correct positioning of a foot, and thus to the stability of balance of the robot, is an accurate estimate of the body angle and angle rate. Recall from Chapter 5 that the motion planning system relies on a temporal prediction of the future state of the roll angle and angle rate at the time the foot is being positioned on the ground.

In this work, a Kalman filter is used to estimate roll angle and angle rate. Kalman filters are predictive-corrective algorithms used to estimate the state of a system in the presence of measurement uncertainty and noise (Olivia 2001; and Welch 2006). It is a recursive filter that estimates the state and covariance of a linear dynamic system from a series of measurements. In the case where complex motion must be predicted (e.g., serpentine motion from a person), a behavior-based motion model may be used. Behavior-based systems input current and expected foothold position and states plus environmental attributes (e.g., distance to another foothold, distance to robot, etc.) in a multiple, hierarchical rules and equations model of the foothold (e.g., animal, person, vehicle, robot, etc.) and are capable of estimating emergent behavior.

The Kalman filter periodically samples the inputs to the (simulated) gyro and accelerometer data. After every data measurement, the Kalman filter is updated, and the state of the Kalman filter is copied and used by the HSM for motion planning. Recall from Chapter 5 that the HSM is capable of estimating hypothesis about the future state of the robot at a much faster rate than the sampled real data.

From (Hong 2008; Larsen 1998; and Raol 2010), the Kalman state and measurement vectors are defined as follows:

$$x_k = [\theta \quad \dot{\theta} \quad V_o]^T \ and \ \ z_k = [\theta \quad V]^T \tag{6.8}$$

where x_k is the kth state and z_k is the kth sensed or observed measurement (i.e., real data) from the MEMS gyro and accelerometer. State and measurement fusion is defined as follows:

$$x_k = Fx_{k-1} + w_{k-1}\,, \quad w_{k-1} \sim N(0, Q_{k-1}) \tag{6.9}$$

$$z_{k-1} = H_{k-1}x_{k-1} + v_{k-1}\,, \quad v_{k-1} \sim N(0, R_{k-1}) \tag{6.10}$$

$$F = \begin{bmatrix} 1 & dt & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & g & 1 \end{bmatrix} \tag{6.12}$$

where F is the state transition matrix, w is the process noise of covariance Q, H is the observation model which maps the true state space into the observed space, and v is the measurement noise of covariance R.

The prediction stage of the Kalman filter estimates the current state from the previous measurement or sensing time. The Kalman state prediction is defined as follows:

$$\hat{x}_k^- = F\hat{x}_{k-1} \tag{6.13}$$

$$P_k^- = FP_{k-1}F^T + Q \tag{6.14}$$

where $\hat{x}_k^-$ is the state estimate, P_k^- is the estimated covariance matrix, and Q is the process noise covariance matrix. There is no Bu_{k-1} term since there are no control inputs.

The correction stage of a Kalman filter uses the current-time (noisy, biased, and scaled) sensed data, z_k, with the predicted state, $\hat{x}_k^-$, to improve the posteriori filtered state estimate, $\hat{x}_k$, and the posteriori error covariance matrix, P_k. The Kalman measurement correction is defined as follows:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{6.15}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-) \tag{6.16}$$

$$P_k = (I - K_k H_k)P_k^- \tag{6.17}$$

where K_k is the (optimal) Kalman gain and R_k is the measurement covariance matrix.

6.1.4. Fusing Time-Delayed Measurements

A third source of sensed data may come from an optical and image processing system, called a machine vision system, which operates at considerably slower rates than the gyro and accelerometer sensors. Kalman filters have been used to fuse angle and angle rate measurements from gyros and vision systems in controlled environments using artificial features or landmarks (Dial 2005; Foxlin 2003; Hu 2004; Kyriakoulis 2009; Lobo 2003; You 2004). In (Pathak 2004), a complementary filter was used to compensate for differences in measurement frequency to obtain accurate angle information. Kalman filters have also been combined with neural networks to better model gyro sensor drift (Baerveldt 1997; and Xiyuan 2003). Also recall from Chapter 5 and previous sections of this chapter, that the motion planning system relies on a temporal prediction of the future state of foothold and obstacle data for the multiple foot placement planning. The following presents a method for fusing multi-rate data.

There is typically a delay from the time when one sensor system is sampled to the time when the angle measurement, θ, is available. In terms of the MEMS gyro sampling rate, dt, it takes N data updates (Ndt) from the time of the center of integration of a camera's image acquisition to compute a vision-based angle measurement, or $\theta_{(k-N)}$. The delayed measurements, $z_{(k-N)}$, cannot be fused using the above Kalman filter method without first correcting the state estimate, $\hat{x}_{(k-1)}$. Figure 6.1 illustrates the system timing diagram used to fuse time-delayed data from a machine vision system with faster rate MEMS gyro data.

While it is possible to re-calculate the entire N data updates with a second, parallel Kalman filter, a more computationally efficient solution is proposed by Larsen et al. (Larsen 1998). The time-delay corrected posteriori filtered state estimate equation associated with 6.16 is defined as follows:

$$z_k^* = H_{k-N} x_{k-N} + v_k^* , \quad v_k^* \sim N(0, R_k^*) \tag{6.18}$$

$$M = \prod_{i=0}^{N-1} (I - K'_{k-i} H_{k-i}) F , \quad N > 0 \tag{6.19}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-) + MK_{k-N}(z_k^* - H_{k-N}\hat{x}_{k-N}^-) \tag{6.20}$$

where z_k^* is the state-estimated measurement computed using the last simultaneous vision and MEMS gyro state, $x_{(k-N)}$, and K_{k-N} is the Kalman gain computed using the covariance from time k-N of the delayed measurement. For all other MEMS gyro-only data updates, equation 6.16 is used.

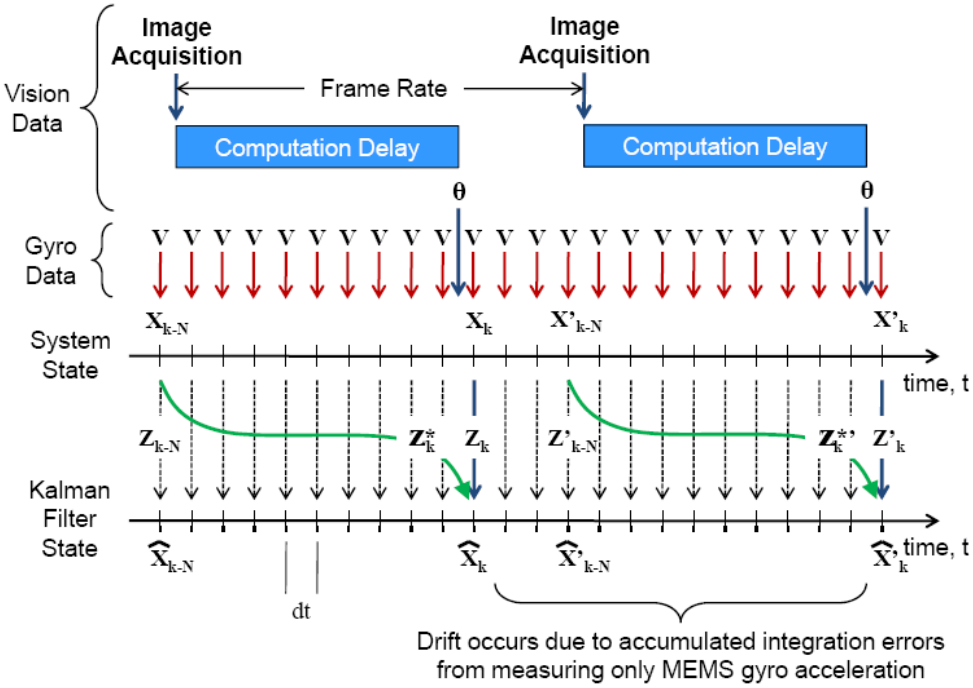


Figure 6.1. Data acquisition system timing and sensor fusion time line. (Adapted from Figure 1 in Hong 2008 and Figure 1 in Larsen 1998).

6.2. Temporal Tracking System

As the ST3LMR moves, the camera images different scenes. A temporal tracking system is used to associate sensed footholds and obstacles, called detections, with prior-observed footholds and obstacles, called tracks, and to predict (or track) the future position of the footholds and obstacles as the ST3LMR moves. Figure 6.2 shows a data flow model of a typical temporal tracking system, illustrating the core temporal loop or module (Blackman 1986). The core temporal module provides functions for predicting the future locations of footholds and obstacles, and it also computes the uncertainty or covariance that the footholds will actually occupy the predicted locations.

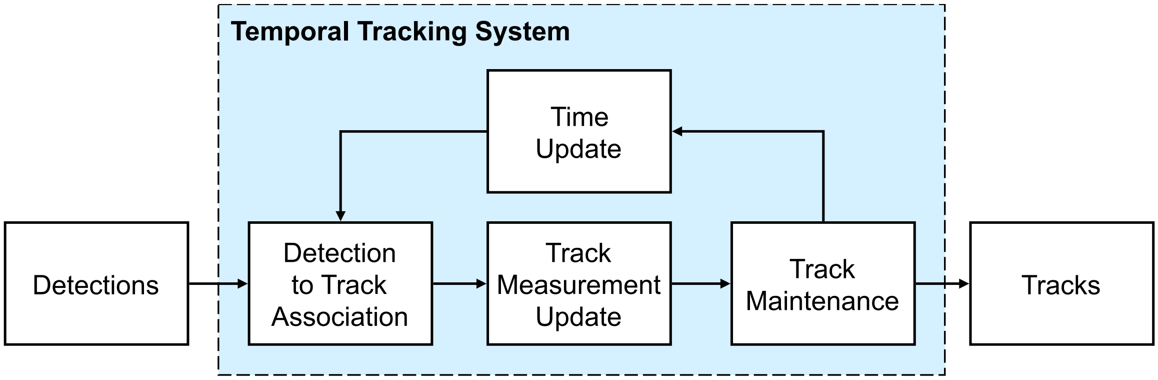


Figure 6.2. Data flow model of a temporal tracking system, illustrating the core temporal loop or module.

From left to right in Figure 6.2, the temporal tracking system receives detections from the computer image processing system. The detection may comprise position, extents, and their covariance or measurement uncertainty. Detection to track association function compares the sensed detection to previously stored and temporally propagated detections, called track files. A detection is associated to an existing track (from a previous sensor observation or a priori map data) using a goodness of fit metric(s) that compares the kinematic (e.g., position) and signal (e.g., color and texture), of the detection with the tracked data propagated to the time the new sensor data is observed.

Next, a track measurement update is performed. If a detection matches a stored track (e.g., they occupy the same position, exhibit the same characteristics, etc.), the stored track data is updated using the newly sensed data. The sensor-based measurement and covariance is used to update the estimated measurement and covariance for each associated track. For example, the current position of a foothold or obstacle, as provided by the most recent sensor data, is used to update the associated track corresponding to the foothold or obstacle. If the detection does not match any stored tracks, a new track file is created and stored. The new or updated track then is stored in a track file database.

In other words, a sensor-based track is a collection of one or more sensor observations in time, with attributes (position, velocity, heading, pose, color, temperature, behavior, etc.) typically stored in a database. The sensed data along with any known or hypothesized data (e.g., classification attributes of the detections from association with the track) about the detection is added to the track at the time the data was observed. The objective of updating the track is to add current sensory data to improve the predicted future position and state estimate and reduce the covariance or uncertainty. In the case of multiple detections, such as a newly-observed detection that was previously occluded, the detection that best matches the track may be associated and the remaining detections discarded or form new tracks. If the detection and track are only somewhat similar (e.g., at least one of but not all of position, size, texture, etc. are similar, as in most commonly detection-to-track occupancy), the detections may form new tracks or form multiple hypothesis on the existing track or both. If the detection and track are dissimilar or no track exists, a new track is started. Multiple detection tracking may be used to remove body motion and isolate to target motion, which improves filtering and prediction.

Next, maintenance is performed on all stored tracks. As will be appreciated, sensor measurement uncertainty and multi-hypothesis motion and state models associated with or applied to each predicted track can spawn multiple tracks, resulting in the number of tracks growing exponentially. To minimize memory use and reduce processing load, tracks in which the robot is unlikely to follow are eliminated. A cost-based metric can be used to score and rank the quality of tracks. For example, tracks that associate and persist over multiple sensor observations in time are maintained, while tracks that are no longer supported by sensor data or are outside the field of regard are deleted. Scoring methods can range from very simple threshold counters to complex rule and equation systems.

During the initial execution, track maintenance may be very minimal, or not performed at all (e.g., the collected and/or generated data is sufficiently small not to warrant maintenance operations). However, and as will be described in more detail in following chapters, each updated position of the robot can spawn new tracked data, and the number of track files can grow exponentially resulting in large amounts of data. Maintenance is performed on this data to eliminate predictions that are unlikely and/or simply not supported by sensor data, and as will be described later, the device and method in accordance with the present invention use scores from the probabilistic foothold-to-foothold and robot-to-foothold modeling and robot-centric simulation to aid in sensor-based track maintenance.

For example, a weighting factor can be assigned to each predicted location. Then, the likelihood that the robot will take a particular path through a location can be computed by summing the weighting factor for each location within the path. Predicted paths with scores higher than other predicted paths (and the locations that define the path) are retained in the database, while paths with scores lower than other predicted paths (and the locations that define the path) can be eliminated.

Determination of whether or not maintenance is performed can be based on various factors. For example, the amount of memory consumed by the data can be used to determine whether or not maintenance should be performed. If the required memory exceeds the predetermined threshold, then maintenance is performed (e.g., track files not supported by the data can be purged). Alternatively, maintenance may be performed during each execution cycle.

Once track file maintenance has been performed, a track file time update is implemented. More specifically, future track states and beliefs are predicted for the next sensor observation time. For example, the position and covariance of a foothold or obstacle is predicted for the time of the next sensor measurement, based on past behavior (position, velocity, acceleration, covariance, etc.). This update can be accomplished, for example, using the straight-line motion equation from physics, and is of the form:

$$x_{t+1} = x_t + \dot{x}_t \Delta t + \frac{1}{2}\ddot{x}_t \Delta t^2 \tag{6.21}$$

Where x_{t+1} is the estimated position at time t plus the incremental time Δt, x_t is the starting position, $\dot{x}_t$ is the velocity over Δt, and $\ddot{x}_t$ is the acceleration over Δt^2. As will be detailed later, the predicted future locations of the footholds and obstacles can be generated using a Kalman filter.

The temporal prediction of future positions can continue as far into the future as desired. However, the further into the future (time) the predictions are made, the less accurate or the more uncertain (higher covariance) they become. Once the track files are time-updated to the current data's sensed time, detection to track association occurs again, and the loop repeats. Note that tracked data is forward- or back-propagated (i.e., as in the position with respect to a given ST3LMR velocity over time, where velocity is measured by leg joint angles during stance phase) to the sensor collection or observation time.

6.2.1. Combined Tracking and High-Speed Model

Introduced in Chapter 5, single-track motion planning concerns mainly estimating the future foot placement position for a leg in flight, given a gait and high-level steering and velocity commands. This section introduces the concept of a high-speed model to simulate possible foot placement actions, and select the single best action to implement. This high-speed model runs asynchronously with respect to the aforementioned tracking system to simulate multiple robot motion hypothesis (multiple tracks) for obstacle avoidance, trajectory planning, and state (orientation, posture, maneuver, etc.) planning. The simulation is time-propagated using sensor data, a priori map data, and the robot in a common robot-centric representation.

Figure 6.3 illustrates the tracking system and high-speed model as a combined two-time-scale model-based predictive control system. The system includes a number of functional modules that implement the various algorithms of predictive control. As used herein, a module refers to hardware and/or software used to implement a specified function. The controller includes a sensor module 4a operatively coupled to the image processing system described above. The detection data is provided to a track file module 4b, which analyzes the data and associates it into sets of observations, called tracks, that are produced by the same detections (tracks can include data corresponding to foothold or obstacle position, velocity, acceleration, heading, etc.).

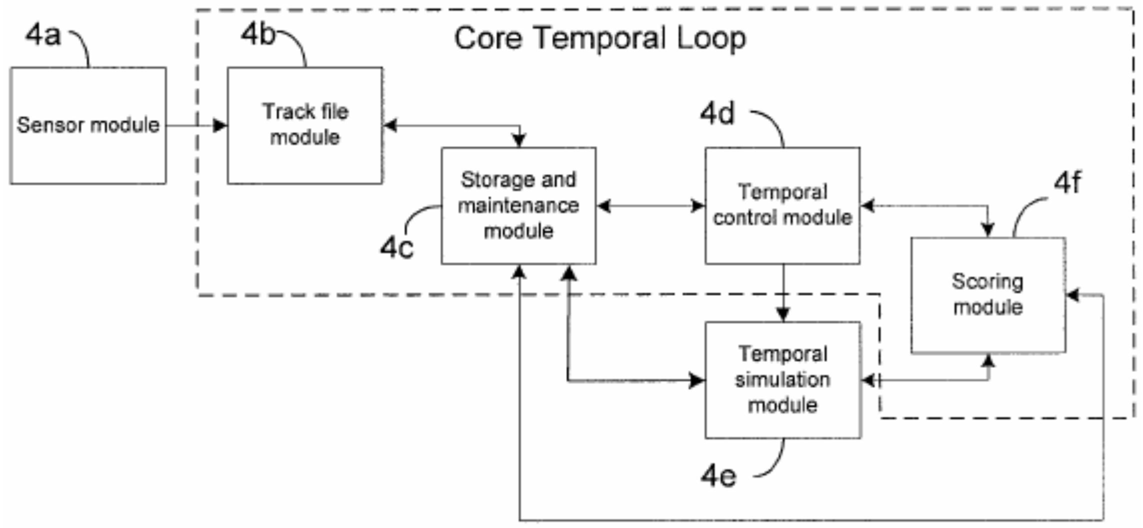


Figure 6.3. Two-time-scale model-based temporal control system combines a tracking system and high-speed model or simulation. (From US patent application publication US 2011/0231016 A1 Fig. 1.)

The core temporal loop modules, the modules illustrated inside the dashed box of Figure 6.3, processes the real-time sensor data. The temporal simulation module 4e, outside the dashed box, performs the multi-hypothesis planning, and it can operate at a much faster rate than the sensor data rate. Storage, retrieval and/or maintenance of the track files to/from the storage device is performed by a storage and maintenance module 4c. More particularly, the storage and maintenance module manages the files stored in memory of the temporal controller 4, which includes writing the data to memory, retrieving data from memory, and purging data from memory that is no longer relevant. Purging data can be based on scores associated with the data, as described in more detail below.

There are two planning modules that share the common database 4c, illustrated in Figure 6.3. The core temporal module 4d is defined as the temporal loop that operates on real-time sensed data. It is independent of the velocity or rate of change of the sensed detections. Rather, the throughput with which the loop cycles are intrinsically tied to the data collection rate, and the track time update is the time between successive data collections.

The temporal control loop is defined as the block that executes all or a portion of the simulated robot(s), as temporal string(s) of action commands. Typically, such action commands are sent to and executed in real-time by a motion control subsystem that minimizes the mean square error between the commanded action and the real-time sensed robot or robot component. The motion control subsystem may feed back the sensed robot position, state and covariance measurements to the temporal simulation loop and/or the robot or robot component (both defined as footholds) may be sensed by the temporal sensor loop.

The temporal simulation module 4e, illustrated in Figure 6.3, is defined as the high-speed model that simulates or time propagates multiple-hypothesis tracks (both sensor-based and robot) into the future. The number of cycles or the total propagation time is independent of both the sensed data and robot control temporal loops. Preferably, the temporal simulation module operates as fast as possible to simulate as far into the future as possible. The depth and complexity of the multi-hypothesis temporal simulation is constrained by the throughput of the computer hardware, required temporal length, time interval and accuracy of robot action commands, and the temporal sensor loop update rate and the sensed robot feedback rate, both of which can cause recalculation of part or all of the simulation.

Beliefs about the predicted state of the foothold or obstacles (e.g., kinematic covariance, classification, and relationships or likelihoods between footholds and their cures) can be estimated over time. In other words, a level of confidence can be associated with the each predicted robot state for each foothold (including both predicted locations of the foothold and responses of the robot based on rider commands/reactions). Such beliefs can be implemented by a scoring module, wherein each predicted leg/foot position is assigned a weighting factor corresponding to how likely the leg/foot will actually occupy the predicted location and/or how likely the rider will actually react to actions taken by the robot. Those paths having the highest score can be considered the most likely path of the robot, while those with the lowest scores can be considered to be the least likely path of the robot. The scores for each location and path can be stored in memory by the storage and maintenance module. Further, the scores can be used by the storage and maintenance module to determine which data should be purged from memory.

6.2.2. Two-Time-Scale Model

Figure 6.4 details the temporal simulation module 90. The temporal simulation module implements the high-speed inverted pendulum model from a robot-centric view (e.g., data is analyzed from the point of view of the robot) as opposed to a sensor centric view. The simulation module uses foothold and obstacle data provided by the track file storage and maintenance module 4c/56 and relevant portions of the scoring module 4f/98, as shown in Figures 6.3 and 6.4, respectively.

The temporal simulation module simulates multiple robot motion hypothesis for obstacle avoidance, trajectory planning, and stability of balance in the roll axis using the methods and equations introduced in Chapter 5. To simulate multiple robot hypotheses, the simulation is time-propagated for all tracked detections, a priori map data, and the robot in a common robot-centric representation. In particular, a new track file is added to simulate the robot, duplicate the other track files to enable a second temporal simulation loop, introduce behaviors to generate action hypothesis for the robot, and provide probabilistic scoring method to optimize strings of action hypothesis over time.

Preferably, the temporal simulation module is executed asynchronously (i.e., two-time-steps) relative to the core temporal module. In other words, the core temporal module and the temporal simulation loop may be viewed as independent systems that interact through one or more common databases. This strategy enables the temporal simulation module to execute at a much higher rate than the core module, as the simulation module is not tied to update intervals associated with data collection and/or I/O devices. For example, the core temporal module is tied to collecting real-time sensor information, which in digital systems typically occurs at fixed intervals. The simulation module is not tied to data collection and therefore can execute at higher rates. Thus, the simulation module may calculate multiple robot-centric hypothesis during a single scan of the core temporal module.

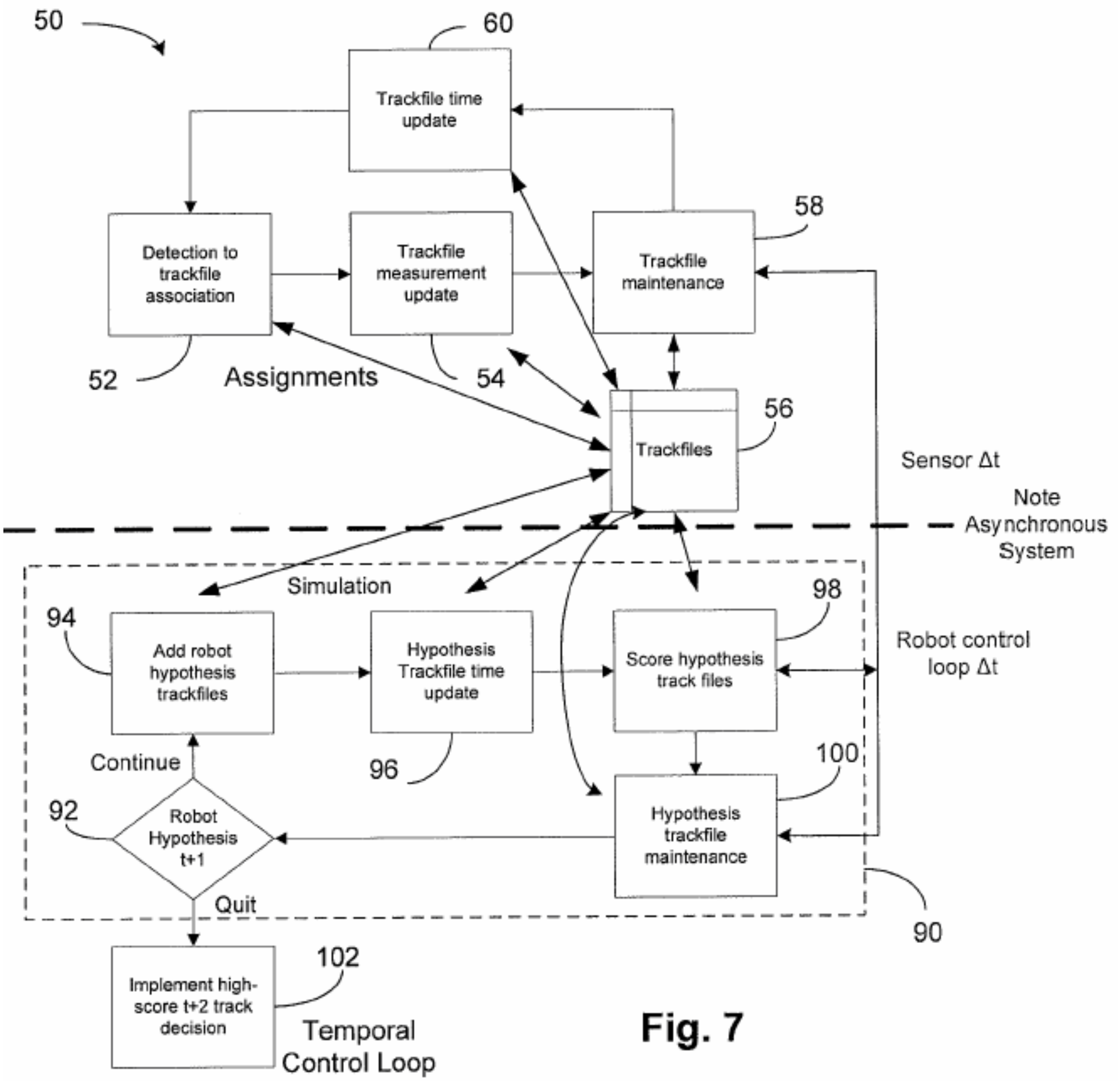

50
60
Trackfile time update
Detection to trackfile association
Trackfile measurement update
Trackfile maintenance
58
52
Assignments
54
Trackfiles
56
Sensor Δt
Note Asynchronous System
Simulation
94
Add robot hypothesis trackfiles
Hypothesis Trackfile time update
Score hypothesis track files
98
Robot control loop Δt
Continue
96
100
92
Robot Hypothesis t+1
Hypothesis trackfile maintenance
Quit
90
102
Implement high-score t+2 track decision
Temporal Control Loop
Fig. 7

Figure 6.4. Two-time-scale model-based temporal control system detail. (From US patent application publication US 2011/0231016 A1 Fig. 7.)

6.2.3. Detailed Operation

The temporal simulation module begins at the Robot Hypothesis 92, shown in Figure 6.4. As noted above, it implements a robot-centric perspective, analyzing data from the perspective of a robot-centric coordinate system, within the existing sensor-based track environment stored in the database. One or more robot track(s) is added to the database. Further, the Robot Hypothesis module may also receive positions and covariance of detection(s) and robot(s) as communicated from other control systems (e.g., another robot) and other sensing and control-related components (e.g., distributed database(s)), such that all robots may operate in common and/or coordinated fashion.

Initial hypothesis regarding how the robot can move within its space are used. For example, the robot can change parameters that affect its position, such as acceleration, velocity, heading or any other parameter that may affect the robot's position and state in the future. A priori map data (that may or may not be associated with sensed-data track(s)) can be used to determine positions that may or may not be occupied by the foothold track(s) and/or the robot track(s). For example, a wall or body of water (represented as a track) may define an area that cannot be occupied by the robot. Steep inclines may define areas that are difficult to occupy or cannot be traversed at high speeds. Further, the robot track(s), initial motion hypothesis, a priori map data, and motion models may be received from another temporal tracking control system (e.g., another robot) and/or from other distributed sensing and control-related components, such that all robots and control-related devices may operate in common and/or coordinated fashion.

Once the top-level or supervisory control strategy (also called an objective or path planning goal) is determined, the individual robot track(s) may be simulated within the robot space determined at function module 94, shown in Figure 6.4. For example, the robot track files are hypothesized using the sensor-based track files from a database of the core temporal module. More specifically, the track files corresponding to the position of footholds with respect to time (including future predicted positions), as determined in the core temporal module, are translated into the robot-centric view. In this manner, the temporal simulation module is made aware of the past, present and expected (future) positions. The track files generated by the core temporal module can be retrieved from the database, or directly from the track file maintenance function of the core temporal module, or instantiated within the database.

Next at the hypothesis time update function 96, shown in Figure 6.4, the track files are temporally updated. Because the sensor-based tracks are updated by physical measurement of the environment (i.e., truth) in the core temporal module, various hypotheses that may have developed from previous iteration(s) of the core temporal module must be time-synchronized. More specifically, in addition to computing the future position, state and their covariance of the hypothesized robot tracks, the hypothesis time update function also computes the probable robot-to-foothold/obstacle interaction and subsequent foothold/obstacle-to-foothold/obstacle interaction by spawning new foothold/obstacle hypothesis tracks, if such foothold/obstacle tracks do not already exist. In other words, a second simulation iteration (similar to the track file measurement update) is performed to predict the actions of the footholds/obstacles due to robot intervention.

A third simulation iteration (similar to the track file measurement update) is performed to predict the actions of the robot(s) due to rider reaction. The positions of footholds/obstacles (including future positions) as determined by the core temporal module is combined with various actions of the robot to simulate how the rider may react to each robot action. The predictions are based on robot position within the robot space and the expected behavior of the rider. Such new robot-to-rider and robot-to-robot hypotheses track(s) are then recorded in the track database to improve sensor detection-to-track association and aid in the classification of footholds/obstacles, as previously described. Further, such new robot-to-rider and robot-to-robot hypotheses track(s) may be communicated to other temporal control systems (e.g., another robot) and other control-related components (i.e., distributed database(s)), such that all robots and control-related devices may operate in common and/or coordinated fashion. Furthermore, the predicted future positions and states and their covariance of footholds/obstacles is also sent to the operator interface device (e.g., a display and touch screen) to show a human operator/observer.

6.2.4. Foot Placement Hypothesis Scoring

Key to determining the single best foot placement to implement is scoring each simulated hypothesis. Hypothesis track files are scored in function 98, shown in Figure 6.4. In a temporal sense, the new data is associated and scored. More specifically, each estimated future position of a foot is scored based on the likelihood that the foot will actually occupy the position, given the sensed data and its hypothesized future position. Furthermore, each predicted position of the robot comprises a possible path (a track) that inherits history from previous portions of the path taken (e.g., the score of the previous predicted position forms part of the history of the subsequent predicted position). This history forms the basis for scoring each path, wherein paths scoring the highest can be considered the most probable path of the robot. At a desired path planning look-ahead time or state, the highest scoring hypothesis is selected as the optimal path. For example, the control objective may be to both maximize stability of balance and minimize/ maximize the distance to one or more footholds/obstacles over the entire path history.

Instead of the foot placement hypothesis being data driven (i.e., based on sensor data), the scoring is hypothesis driven based on the robot-centric view. For example, instead of the predicted positions being based on track file derived from sensor data, the track files are based on actions (or inactions) made by the robot. Such actions may include the robot moving to the right (moving the line of support to the left), moving straight ahead, or moving to the left (moving the line of support to the right) to maintain stability of balance. These actions can alter the score, for example, by increasing or decreasing the distance of the foot position with respect to the body. Hypothesis scoring may also incorporate the line-of-sight (LOS) distance between the predicted foot position and a predicted foothold track. Thus, reinforcing intermediate foot positions corresponding to footholds while reducing the occurrence of intermediate foot corresponding to obstacles, all while maintaining stability of balance (in the roll axis).

Each predicted position of the robot comprises a possible path (a track) that inherits history from previous portions of the path taken (e.g., the score of the previous predicted position forms part of the history of the subsequent predicted position). This history forms the basis for scoring each path, wherein paths scoring the highest can be considered the most probable path of the robot. At a desired path planning look-ahead time or state, the high scoring path is selected as the optimal path. For the above LOS scoring example, the control objective may be to both maximize stability of balance and minimize the LOS distance to one or more footholds over the entire path history, wherein a pair-wise comparison is made. The scores are then recorded in the track database to improve sensor detection-to-track association and aid in the classification of footholds/obstacles, as previously described. Further, scores may be communicated to other temporal tracking control systems (e.g., other robots) and other control-related components (i.e., distributed database(s) 56), such that all robots and control-related devices may operate in common and/or coordinated fashion.

Maintenance of the hypothesis track files, function 100 in Figure 6.4, may be implemented as described above with respect to the track files of the core module. For example, based on the score for each path, it can be determined which path or paths are not supported by the hypothesis. Then, these paths then can be discarded, and the remaining hypothesis track files can be stored in memory. In addition, the depth of the simulation (i.e., the number of times the robot, robot-to-foothold/obstacle, and foothold/obstacle-to-foothold/obstacle hypothesis are estimated into the future) may be temporally “pruned back” based on interaction with sensor-based track maintenance to time-align the robot hypothesis for new sensor-based track data, performed in the various high-speed module function. The lines in Figure 6.4 connecting track maintenance, score hypothesis track, and hypothesis track maintenance indicate this functional relationship.

Finally, the entire foot placement high-speed simulation then moves back to the Robot Hypothesis and repeats using the latest sensor-based data. Future moves may be calculated as far into the future as possible until a newly sensed detection makes a change that causes many scenarios to be recomputed. The "pull" from the leg state machine to change its state is the factor that freezes and implements the best action. Since the tracks take into account foothold and obstacle motion and robot motion from the perspective of the robot, an accurate future map of the environment is obtained.

The best action for the robot(s) or component(s) being controlled is then sent to the motion control system. Further, the best temporal string of action(s) may be communicated to other temporal tracking control systems (e.g., other robots) and other control-related components (e.g., distributed database(s)), such that all robots and control-related devices may operate in common and/or coordinated fashion. Further, the top N temporal string of action(s) (i.e., predicted future positions and states and their covariance) for the robot(s) or component(s) being controlled is then sent to the operator interface device (e.g., a display and touch screen) to show the predicted future positions and states and their covariance of the robot(s) and/or controlled component(s) to a human operator/observer. Chapter 7 details the control architecture required to implement the above.

CHAPTER 7

THE ST3LMR CONTROL SYSTEM ARCHITECTURE

This chapter describes the fully or partially autonomous control system architecture for the Single-Track Three Legged Mobile Robot (ST3LMR). Design of the ST3LMR control system architecture begins with mechanical, control, and operational requirements. The requirements are summarized, and a basic to complex control architecture is proposed.

7.1. Requirements-Based Design

Requirements both guide and constrain the design. The ST3LMR control system architecture is developed from the requirements. Mechanical requirements are presented in Chapter 3. The control and operational requirements are summarized and presented here for clarity. Note that requirements also provide a means for determining when the digital simulation and/or physical prototype is "good enough" to stop improving.

7.1.1. Control System Requirements

An omnidirectional control system, in communication with the leg mechanisms and receiving sensed data, determines possible future states of the legged vehicle and coordinates movements of the leg mechanisms and body to enable movement of the legged vehicle in three dimensions over the ground. The control system enables movement of the legged vehicle forward, backward and sideways. The leg mechanisms, such as the actuators, are controlled by the control system to selectively induce three-dimensional forces, and roll, pitch and/or yaw torques between each foot and the ground. Controlling leg movement, torque, extension velocity and retraction, to use the mass of the leg to impart forces and torques to the frame, is discussed in the Operational Requirements, Section 7.1.2.

The control system includes an inertial measurement unit mounted on the body to sense attitude, and specifically roll angle, roll angle rate, roll angle acceleration of the body. Leg-positioning control along the length of the body is decoupled from leg-positioning control normal to the length of the body. Further, each leg mechanism includes position-measuring, force-measuring and torque-measuring components providing feedback to the control system.

The control system is used to coordinate and control the ST3LMR, and may include one or more central processing units (CPU) and one or more memory components. The memory components may include one or more memory modules, such as Random Access Memory (RAM) modules, Read Only Memory (ROM) modules, Dynamic Random Access Memory (DRAM) modules, and any other suitable memory modules. Such memory is used to store a priori data, such as for example the state gait database for the Leg State Machine, discussed in Chapter 3, and sensed data for the Motion Planning system, discussed in Chapter 4.

The control system may also include a plurality of input/output (I/O) components that may include a variety of known I/O devices, including network connections, video and graphics cards, disk drives or other computer-readable media drives, displays, or any other suitable I/0 modules. One or more data busses may operatively couple the CPU, the memory component, and the I/O component. The control system may be operatively coupled to a control component having a data display/monitor and a command/control input device (e.g. a keyboard, an audio-visual input device, handlebars, foot pegs, pressure pads, and so on).

At least one accelerometer and gyroscope, in communication with the control system, is mounted on the frame normal to the length of the body and the major axis to sense the roll condition. The control system receives sensed data from the accelerometer and gyroscope to sense velocity, acceleration, attitude, and gravitational forces. Additional sensing including the pitch, yaw, x, y, and z axis is required for omni-directional guidance, navigation and control.

Each leg mechanism includes position-measuring components providing feedback to the control system. These position-measuring components provide information regarding relative or absolute leg position to the control system. Such information provides the advantage of more-accurate leg-movement corrections based upon a comparison, within the control system, of commanded or desired leg placement in comparison to actual leg placement.

Each leg mechanism includes force-measuring components providing feedback to the control system. These components provide numerous advantages, including accurate determination of the loaded weight of the vehicle, accurate determination of the leg energy required, via one or more actuators, to perform a desired maneuver, such as straight-line walking, and accurate measurements of forces experienced at each leg and through the frame. This permits the control system to compensate according to the desired trajectory.

Each leg mechanism includes torque-measuring components providing feedback to the control system. These components provide numerous advantages, including accurate determination of the loaded weight of the vehicle, accurate determination of the leg energy required, via one or more actuators, to perform a desired maneuver, such as straight-line walking, and accurate measurements of torques experienced at each leg and through the frame. This permits the control system to compensate according to the desired trajectory.

The control unit has or is in communication with an operator interface, which is in communication with the control system, the control system receiving sensed data from the operator. This arrangement allows the advantage of remotely-directed control of the legged vehicle, and allows a rider/operator to control the legged vehicle. The operator interface of the control unit may be in wireless communication with the control system or it may be physically attached to the body. The operator interface components provide at least steering angle, throttle and braking inputs into the control system. In this fashion, the legged vehicle is controlled by an operator in a manner similar to that of controlling a motorcycle, but with the benefit of discrete foot placement.

Yaw angle; centripetal acceleration; forward motion; roll angle; steer angle, and steer torque are common to bicycles and motorcycles. Ouden provides various parameters related to regular bicycles which are adopted for this research (Ouden 2011):

1) Rotation, angular velocity, and angular acceleration in the roll and yaw directions; and

2) Velocity and acceleration/deceleration in the forward/backward and lateral (e.g., centripetal acceleration during cornering) directions.

7.1.2. Operational Requirements

The ST3LMR includes a power source connected to and driving the control system components and the plurality of actuators which drive the legs. Forward/backward movement of the legged vehicle is according to approximately single track foot placement, as described in previous chapters. Movement of each leg includes each of the four phases of motion for each leg. The first phase, called Stance, involves developing reaction forces, torques, and thrusts wherein leg/foot-to-ground interaction is transferred through the leg to stabilize the frame in the pitch, roll, and yaw axes and to propel the frame in the x, y, and z axes, respectively. The foot/distal end of the leg being generally stationary with respect to the ground during the stance phase and moving generally opposite to the major direction of frame motion. The legs/feet shall be capable of executing a monopedal stance, a bipedal stance and a tripedal stance, according to the control system.

The second phase, called Unloading, involves removing the foot-to-ground reaction forces through the leg/foot in a stance-to-flight phase wherein the foot is lifted off the ground, controlling leg velocities, according to the control system. Repositioning of the leg/foot, called the Flight phase, involves repositioning the distal end of the leg/foot in the same general direction as the frame and generally at a faster rate, relative to the ground, as the major direction of frame motion. Foot placement and leg movement is controlled to maintain an upright posture and meet foot placement constraints and desired trajectory requirements for the frame and legged vehicle, according to the control system. The fourth phase, called Loading, involves positioning the leg/foot on the ground and developing reaction forces, torques, and thrusts in a flight-to-stance phase.

The method of executing straight-line legged motion with an in-line legged vehicle includes tracking and synchronizing each phase for each leg, according to a selected predetermined gait model, with a state machine. A foot is positioned to the right or to the left of the projected center of gravity on to the ground to develop ground reaction forces that are normal to the major direction of motion, and ground reaction torques in the pitch, roll, and/or yaw axes. Leg length during a stance phase is different between feet positioned to the right or to the left of the projected center of gravity of the body on to the ground in order to level the body attitude, within the working range of the legs and their feet.

All three feet are positioned in a tripod stance to provide a period of stability in the x, y, and z axes and pitch, roll, and yaw axes. Further, any two feet may be positioned, one to the left and one to the right of the projected center of gravity of the body onto the ground in a generally bipedal stance with respect to the length of the body and major direction of motion, to provide a period of stability in the roll axis. As discussed in Chapters 4 and 6, the combined ground reaction forces developed by three feet positioned in a tripod stance, may impart a torque in the pitch, roll and/or yaw axes to aid in stabilizing the body. A yaw torque about the center of gravity is developed by the interaction of two or more legs (or a foot with a knee/hip rotation actuator) with the ground.

Retraction of the legs during flight phase is generally inwards towards the body and along the major direction of motion such that no torque is imparted to the body in the roll axis. However, a leg may be swept/swung outward during a flight phase without reducing its length to impart a torque in the pitch, roll and/or yaw axes to aid in stabilizing the body. Once kicked outward, the leg length may be induced in-flight, such as by bending the leg at the knee, and the shortened leg swept back inward towards the ground. Thus, imparting less torque in the pitch, roll and/or yaw axes than the outward kicking motion.

The method of operating the ST3LMR further includes leaning the vehicle into a desired direction of turn, wherein a projected center of gravity is laterally displaced inwardly from a point within a triangle defined by foot contact with the ground. Torque is thus developed around the roll axis in the direction of the lean. Then, one or more feet are displaced normal to and spatially distant from the projected center of gravity in the opposite direction of the lean to develop an outward torque about the roll axis to counteract the inward lean, wherein the trajectory becomes a curved line. Torque is also developed from at least one leg or a combination of two or more legs to rotate the frame along a curved trajectory. This aspect is key to the “lean into the turn maneuver” and may precede the turn, occur continuously or discretely during the turn, and/or provides a means for exiting the turn to pursue a different trajectory.

The method of operating the ST3LMR further includes sensing the roll condition from an applied external force and leaning the vehicle into the direction from which the external force is applied, wherein the roll condition is neutralized, according to the control system. Examples of external forces include, but are not limited to wind forces, impulse forces, centripetal forces, and gravitational forces (due to loss of traction causing the frame to fall over). Leaning the vehicle may be accomplished in one of several ways including but not limited to developing a foot-to-ground reaction torque about the ankle and/or placing one or more feet in the same direction as the applied external force, beyond the projected center of gravity plus a distance equal to or greater than is required to counter the estimated dynamic momentum at the time the foot is repositioned. Furthermore, the external force may be desired, to initiate a lean into a turn maneuver, for example, but ultimately the roll condition must be neutralized to maintain an upright vehicle posture.

Leaning into a turn is accomplished by balancing the centripetal forces with respect to the center of gravity, as exemplified through the displacement of the center of pressure from foot placement with an acceleration force of the legs. The method of turning the legged vehicle may include the steps of:

1) Leaning the vehicle into a desired direction of turn, wherein a projected center of gravity is laterally displaced inwardly from a point within a triangle defined by foot contact with the ground, wherein a torque is developed around the roll axis in the direction of the lean; and

2) Displacing one or more feet normal to and spatially distant from the projected center of gravity in the opposite direction of the lean to develop an outward torque about the roll axis to counteract the inward lean, wherein the trajectory becomes a curved line.

The outward and inward torques may be adjusted as necessary to conform to a desired radius of turn. Of course by leaning into the turn, an inward torque, toward the radius of the turn, is created by gravity due to the unstable positioning of the projected center of pressure. It is necessary to create a sufficient outward (centripetal) force from several leg movements to balance the inward force. The leg movements move the vehicle along the desired curved line, or around the desired radius of turn. The turn may be stopped or changed as desired through leg movements arranged to adjust the position of the projected center of pressure.

7.2. Control System Architecture

Three basic control requirements for the ST3LMR are: 1) maintain stability of balance in the roll axis; 2) implement throttle, brake, and steering commands via a rider interface; and 3) avoid sensed obstacles and use preferred footholds. An inertial measurement unit mounted on the body senses body attitude and in particular roll angle, roll angle rate, and roll angle acceleration. Joint angle, force, and torque sensors on each leg and foot determine the kinematic properties. In the previous chapter, six planning steps were discussed to control and coordinate foot motion, including:

1) Collecting sensor data;

2) Associating the collected sensor data into sets of observations;

3) Storing and maintaining a real-time database of sensed data and robot state;

4) Estimating the future position and their covariance of tracked footholds and obstacles based on past positions and covariance;

5) Estimating multiple hypothesis of the future foot positions, robot states and their covariance; and

6) Scoring, ranking, and selecting the *best* foot hypothesis with respect to multiple planning goals and objectives.

Prior research in legged locomotion has focused mainly on the case in which the projected center of mass is kept within the leg support area, i.e., the robot will not fall over. Multi-legged systems that maintain three or more legs in contact with the ground (or use large feet) at all times, such as a quadruped, provide natural stability of balance. The ST3LMR, unlike other robots, may have only two legs in contact with the ground during forward locomotion. It is, by design, an unstable robot, and it requires an active control mechanism to prevent it from falling over in the roll axis. The ST3LMR, however, also has times (both periodic and aperiodic, depending on the gate) during forward locomotion where all three legs are in contact with the ground. During the stance phase of a wave gate, for example, one of the legs may be shifted laterally to affect the line of support and thus to affect balance. This section begins with partially autonomous mode and then develops the control architecture and methods required for fully autonomous operation.

Briefly, the ST3LMR includes a frame, wherein the frame includes a major axis corresponding to and generally parallel to a forward/backward direction of travel. A plurality of jointed leg mechanisms attaches to the frame, one behind the other, wherein each leg is attached at its proximal end at one or more discrete attachment points. The attachment points are arranged substantially parallel to the major axis of the frame and the forward/backward direction of travel. Each of the legs includes actuators attached between the legs and the frame and between adjacent leg members. The legs are actuated for movement of the distal end in three dimensions. A control system in communication with the leg mechanisms and receiving sensed data to determine possible future states of the legged vehicle is used to coordinate movements of the leg mechanisms and frame, and movement of the legged vehicle in three dimensions over the ground; and a power source connected to and driving the control system components and the plurality of actuators which drive the legs, wherein forward/backward movement of the legged vehicle is according to approximately single track foot placement. The term ‘single track’ shall be interpreted as referring to the general narrowness of the foot-placement patterns along a straight or curved path.

7.2.1. Basic Architecture

In the preferred embodiment, the control system is a low-level or real-time processor primarily responsible for dynamic actions with respect to the ground reaction force, the reaction force produced from gravitational forces and inertial forces. The term "ground reaction force" is used here to mean the resultant force and moment at a point of action obtained as the vector sum of all ground reaction forces acting on individual legs. Specifically, the control system is responsible for planning and reacting to mechanical feedback, called preflex-dominated control. A preflex is defined as the zero-delay, intrinsic response of a neuromusculoskeletal system to a perturbation and is programmable via pre-selection of muscle activation (Choset, Lynch, et al. 2005). Leg preflex, for example, pulls the foot back and lifts it if a leg/foot force and torque sensor indicates it encounters an unexpected obstacle during a flight phase. Leg preflex, also for example, causes the leg to push downward if the leg/foot force and torque sensor indicates that it is not bearing adequate vertical load during touchdown (flight to support transition) or during stance (as in loose ground). Leg preflex, also for example, causes the relative leg length to be adjusted so the body remains level. These are feed-forward control processes, and the robot is stable when the forces acting on it are in dynamic equilibrium.

As introduced in Chapter 5, the most basic control system shall be capable of catching a fall and preventing the ST3LMR from falling over. Referring now to Figure 7.1, the individual ankle actuators 52 and 53 of the leg 43 of the ST3LMR are shown with a six dimensional force and torque sensor 81 of conventional design. By measuring the x, y and z force components Fx, Fy and Fz transmitted to the ST3LMR through the feet 45 and also measuring the moment components Mx, My and Mz around the three axes, the six-dimensional force and torque sensor detects whether or not the associated foot has landed and the magnitude and direction of the forces acting on the supporting leg.

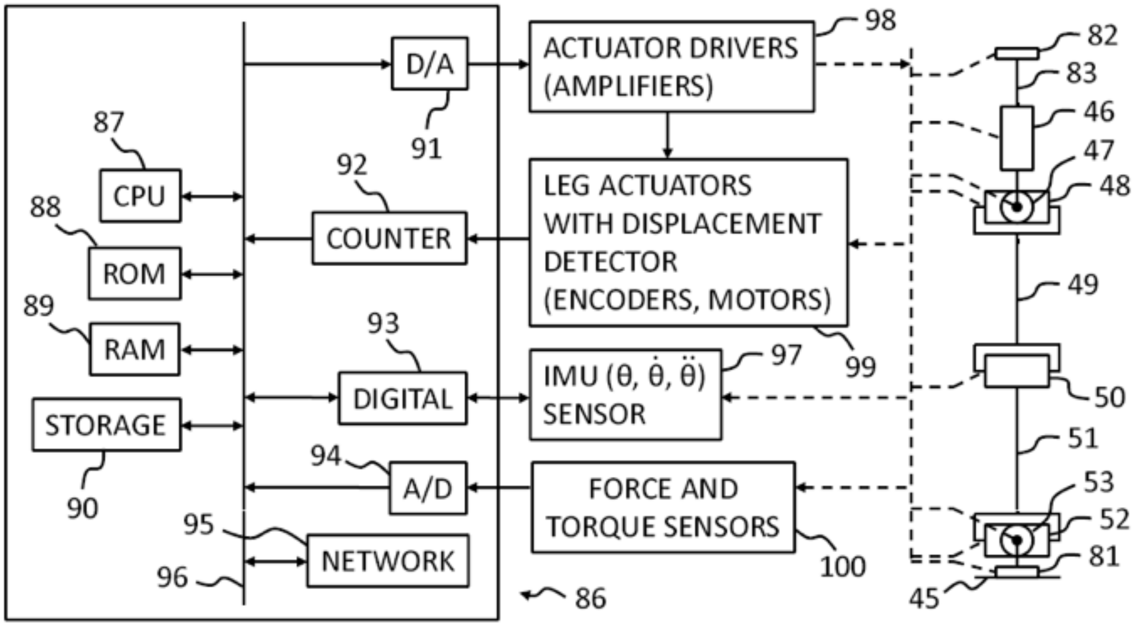


Figure 7.1. Skeletal schematic view of a single leg, illustrating the fundamental feedback and control system of the single-track three legged mobile robot. (From US patent 8,457,830 Fig 31.)

The ST3LMR body is provided with a three-dimensional inclination sensor 82, called an inertial measurement unit or IMU, rigidly connected by mount 83 that is ultimately connected to the leg mounts, not shown for clarity. The IMU is sometimes also referred to as an inertial navigation system or INS. An INS combines the IMU with complementary filters and kinematic proprioceptive information (body height, center of pressure, zero moment point, etc.) to provide more accurate dynamic information. The IMU measures the robot's three-dimensional (roll, pitch, and yaw) angles, angular velocities, and angular accelerations relative to the z-axis in the x-z reference plane, y-z plane, and ground (x-y) plane, not shown for clarity. Each actuator at the individual hip, knee, and ankle joints is provided with a rotary encoder for generating sensed kinematic data for actuation control and posture feedback.

Although not illustrated in Figure 7.1, the ST3LMR is provided with a zero reference switch, such as an oil-damped pendulum, for calibrating the output of the IMU and a limit switch for a failsafe to stop motion in the case of overturn. The outputs of the sensors are sent to the control unit 86.

The control unit is a computer comprising at least one central processing unit or CPU 87, read only memory or ROM 88, random access memory or RAM 89, data storage 90, and input/output devices including but not limited to digital-to-analog converter or D/A 91, digital counter 92, digital interface 93, such as for example a universal serial bus or USB port, analog-to-digital converter or A/D 94, and network interface 95, such as for example an Ethernet port. All aforementioned devices are connected together by at least one bus 96. The angle, angle rate or velocity, and angle acceleration 97, from the IMU is communicated to the control unit via the digital interface. The D/A output is amplified 98 to control the joint actuators with the resulting encoders providing joint angle feedback 99 converted into digital signals by the digital counter. Feedback 100 from the six dimensional force and torque sensor is input to the A/D converter. The resulting digital values are sent via the bus to RAM for storage.

7.2.2. Advanced Architecture

In a temporal predictive foot placement planning system, data from the sensors is stored and organized in a track file database. A track file, simply called a track, is a collection of one or more sensor observations with attributes, such as position, velocity, heading, pose, and so on. A temporal loop estimates future foot positions and covariance, using temporal-based equations of motion. A dynamic model, such as an inverted pendulum model is then applied to predict the robot state (specifically roll angle and angle rate) over time. A high-speed model performs multi-hypothesis planning in a Monte-Carlo simulation. It operates at a much faster rate than the sensor data rate. The temporal simulation loop is implemented from a robot-centric view (e.g., data is analyzed from the point of view of the robot) as opposed to a sensor centric view. The best path is determined as the path that has the lowest probability of inducing unrecoverable platform roll. Uncertainties in the velocity-heading distribution can be expressed as a covariance. As can be appreciated, the further out in time the position is estimated, the larger the uncertainties become.

Especially in rough terrain, sensing suitable footholds and coordinating foot placement is key to balance stabilization during locomotion. Adjusting step length, forward speed, body height, and duration of ground contact must all be planned or the robot will fall over. The legged mobile robot control system must also account for rider-induced perturbations (especially in the roll axis) and mechanical losses in the system. A predictive control system is required to estimate a future position or state.

As was introduced in Chapter 6, the High-Speed Model (HSM) estimates all possible future leg positions using a branching multiple hypothesis temporal simulation. It ranks and selects the best foot placement combination over multiple steps (time) to maintain stability of roll. When a foot is ready to be moved, the next foot position is sent to the Leg State Machine. The individual leg joint angles are calculated using inverse kinematics, and the robot motion is simulated using individual PD control loops for each leg (similar to the physical proof-of-concept prototype described in Chapter 8). For each time interval of the simulation, new initial conditions of the robot are sent to the high-speed model, and the best foot placement position to maintain stability of roll is continuously updated while the foot is in flight.

Figure 7.2 illustrates a data-flow model of the basic functional components of the Single-Track Three Legged Mobile Robot (ST3LMR). From left to right, the environment is sensed using an image processing system, the sensed data is processed and flowed to a predictive model, which receives high-level control input and low-level kinematic data from on-board sensors, resulting in foot placement trajectories that are then filtered by a leg state machine and fed to real-time servo controllers, which then actuate the legs.

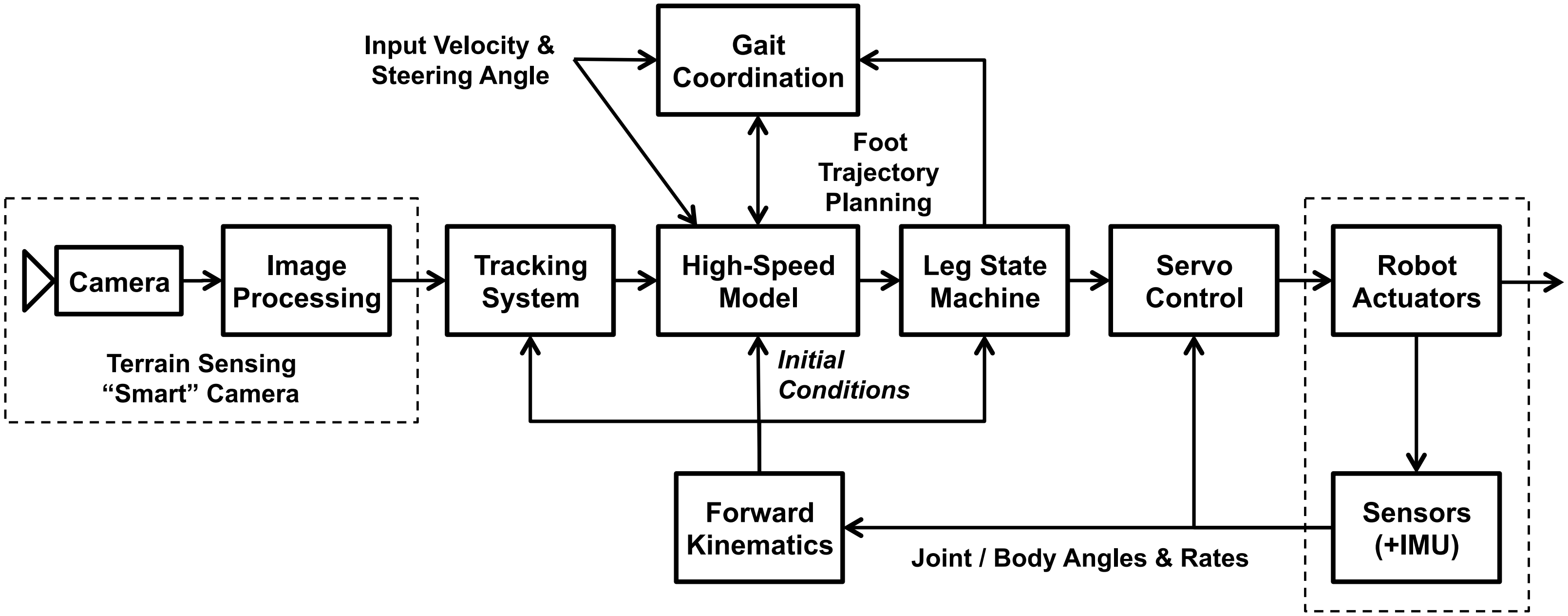


Figure 7.2. Data flow model of the robot control system, illustrating the high-speed model receiving tracked sensory data, available gaits, and kinematic data and outputting control signals to the leg state machine.

7.2.3. High-Performance Architecture

Preferably, the high-speed temporal simulation loop is executed asynchronously (i.e., two-time-steps) relative to the core sensor temporal loop. That is, sensor data processing and the predictive simulation may be viewed as independent systems that interact through one or more common memory interfaces. In the preferred embodiment, a high-performance computing (HPC) architecture provides multiple cores for sensor-rate and simulate-rate algorithms. Figure 7.3 illustrates a HPC architecture adapted from the Intel dual processor Nehalem architecture, left, shown connected with a skeletal schematic view of a single robot leg, right (Intel 1987).

The purpose of having two processors with one or more cores each is to separate time-critical functions, such as computing forward and inverse leg kinematics, from non-time-critical functions, such as the multi-hypothesis temporal simulation or high-speed model. That is, the HPC architecture enables the high-speed simulation algorithms to execute at a much higher rate than the real-time sensor algorithms, as the simulation rate is not tied to update intervals associated with data collection and/or I/O devices. For example, the core temporal loop is shown tied collecting real-time sensor information, which in digital systems typically occurs at fixed intervals. Because the simulation loop is not tied to data collection, it can therefore execute at higher rates, i.e., a two-time-scale system.

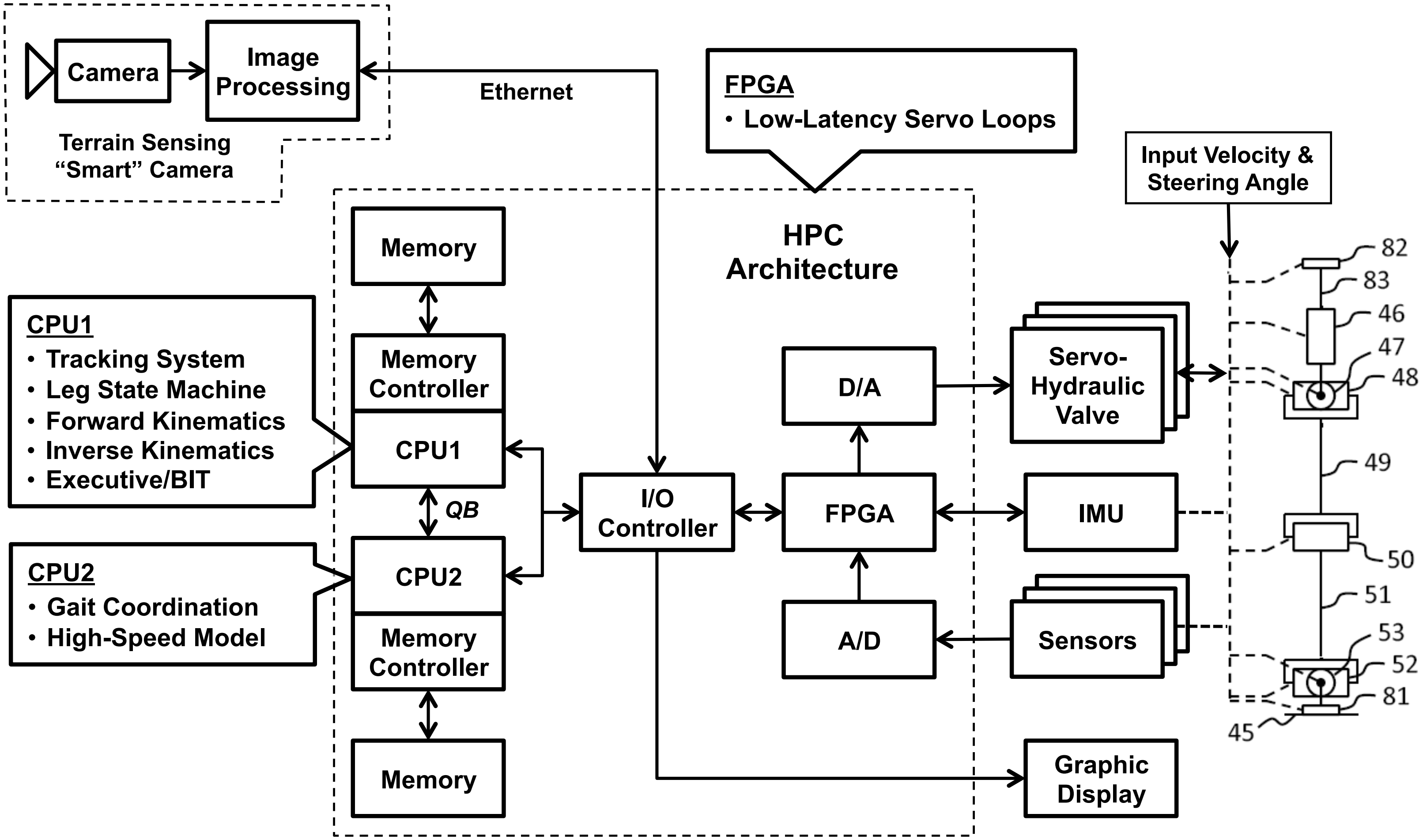


Figure 7.3. System functional block diagram for an ST3LMR High-Performance Computing (HPC) Architecture based on the Intel dual processor Nehalem architecture.

The core temporal loop and the temporal simulation loop may also be viewed as independent systems that interact through the common database (or memory). Track data is retrieved and stored across the dedicated processor-to-processor QuickPath™ interconnect bus. Thus, multiple robot-centric hypothesis planning simulations may be performed by the predictive loop during a single scan of the core real-time sensors.

A Field Programmable Gate Array (FPGA) is primarily responsible for real-time control, such as maintaining the ground reaction force and adjusting the ankles to mitigate in-stance roll disturbances. The term ‘ground reaction force’ is used here to mean the resultant force and moment at a point of action obtained as the vector sum of all ground reaction forces acting on individual legs. The control system senses terrain information through one or more sensor systems, such as a camera and image processing system. The outputs of the inertial measurement unit (IMU) and sensors are fed back to the control system.

7.2.4. Additional Architecture Considerations

It is noted that while a single central database is shown in the figures and described herein, it is contemplated that multiple databases may be used to store data, and these databases may be distributed throughout the system. For example, an exemplary control system may include multiple sub-systems (e.g., multiple distributed sensor collection systems), wherein the sub-systems each have their own database. Data corresponding to sub-system "A" may be stored in database "A", which is local to sub-system "A" (e.g., database "A" is located in the general vicinity of sub-system "A").

Further, data corresponding to sub-system "B" may be stored in database "B", which is local to sub-system "B". Data from each of database "A" and database "B" then may be used within the context of the present invention.

Initial hypothesis regarding how the robot can move and change state within its space may come from an operator interface device (e.g., a display and touch screen). Such a device would dynamically accept and respond to strings of temporal-based human operator commands. Based on robot parameters, a priori map data, communicated data, and/or the operator temporal commands, probabilistic hypothesis can be made regarding the robot and/or foothold positions and their covariance at a future time period. Such probabilistic hypotheses (as tracks) can be made as far into the future as desired, but with increasing uncertainty.

Furthermore, scores may be received by communication from other temporal control systems (e.g., another robot ahead of the current robot and measuring physical data, such as loose soil) and other control-related components (e.g., distributed database(s)), such that all robots and control-related devices may operate in common and/or coordinated fashion.

To improve throughput, only the tracks modified by sensor data need be updated. Because more foothold-to-foothold and robot-to-foothold modeling and robot-centric simulations can be performed between track maintenance intervals, this has the benefit of improving (by depth of analysis or simulation, i.e., the number of temporal iterations of simulation module) the aforementioned track rating and ranking scores used for sensor-based track deletion which then improves the quality or certainty of remaining tracks and thus computational efficiency of subsequent simulations.

This and previous chapters introduce fundamental concepts to plan and coordinate foot placement position to maintain (regain) stability of balance in the roll axis. The goal of this work is to develop a robot that achieves the form, function, and convenience of a motorcycle but with the added benefit of legs and partial or fully automatic stability of balance (like a horse). Chapter 8 presents the physical prototypes constructed to demonstrate the feasibility of the ST3LMR employing such a control system.

CHAPTER 8

THE ST3LMR PROTOTYPES

This chapter presents the proof-of-concept prototype Single-Track Three Legged Mobile Robot (ST3LMR). Due to mainly mechanical limitations of the proof-of-concept prototype, which limit performance and prevent verification and validation of the simulation, a second-generation single-track prototype – a Draisine robot – was designed and built to test principles of controlling accelerations in the roll axis (and to limit costs). The design and construction of the second-generation Draisine (wheel-leg-wheel) prototype is presented in this chapter. Coupled-drive leg experimental results and Draisine robot experimental results are presented in Appendices C and D, respectively.

The contribution of this work is a robot design that achieves the desired single-track form and function of the Draisine with the added benefit of robotic control. Through prototyping, the proof-of-concept ST3LMR demonstrates the single-track concept, providing the advantage of gait-based inherent stability along the pitch axis, and decoupling and simplifying control of balance to only the roll axis. Through simulation, the ST3LMR design is shown capable of sideways thrusting of the body for obstacle avoidance and leaning into a turn, with two legs torqueing the body in the yaw axis to follow a desired curve. This design and method is highly important so that a legged machine can achieve single-track turns over rugged terrain, not heretofore accomplished by legged mobile vehicles or robots.

8.1. Proof-of-Concept Prototype ST3LMR

A one-fourth scale model prototype of the ST3LMR was built using off-the-shelf parts from a Robotis Bioloid robotics kit, as shown in Figures 8.1 and 8.2. The parts included 15 AX-12+ Smart Actuator Dynamixel (5 per leg, with 2 axis each at the hip and ankle), CM-510 Servo Controller, GX-12 2-Axis Gyro Sensor, Composite Framing System, LBS-10 11.1v LiPo battery, and custom frame (a drilled aluminum plate). The robot weighs 4.0 pounds and the legs stand 6.5 inches tall from foot to the hip axis. The battery, controller, and accelerometer were arranged such that the center of mass was located above the middle leg hip axis. The accelerometer was oriented on the frame to measure the pitch and roll axis.

Robotis software was used to program the robot controller. Several simple control algorithms were written to:

1) Initialize the robot in a tripod stance, as shown in Figure 8.1;

2) Transition the legs out of the tripod stance and into a backward wave gait (BWG);

3) Walk forward implementing a BWG; and

4) Transition the legs out of the BWG and into a tripod stance.

An accelerometer feedback loop was written to dynamically adjust the hip and ankle actuators during (2)-(4). The robot successfully demonstrated (1)-(4) on flat level surfaces.

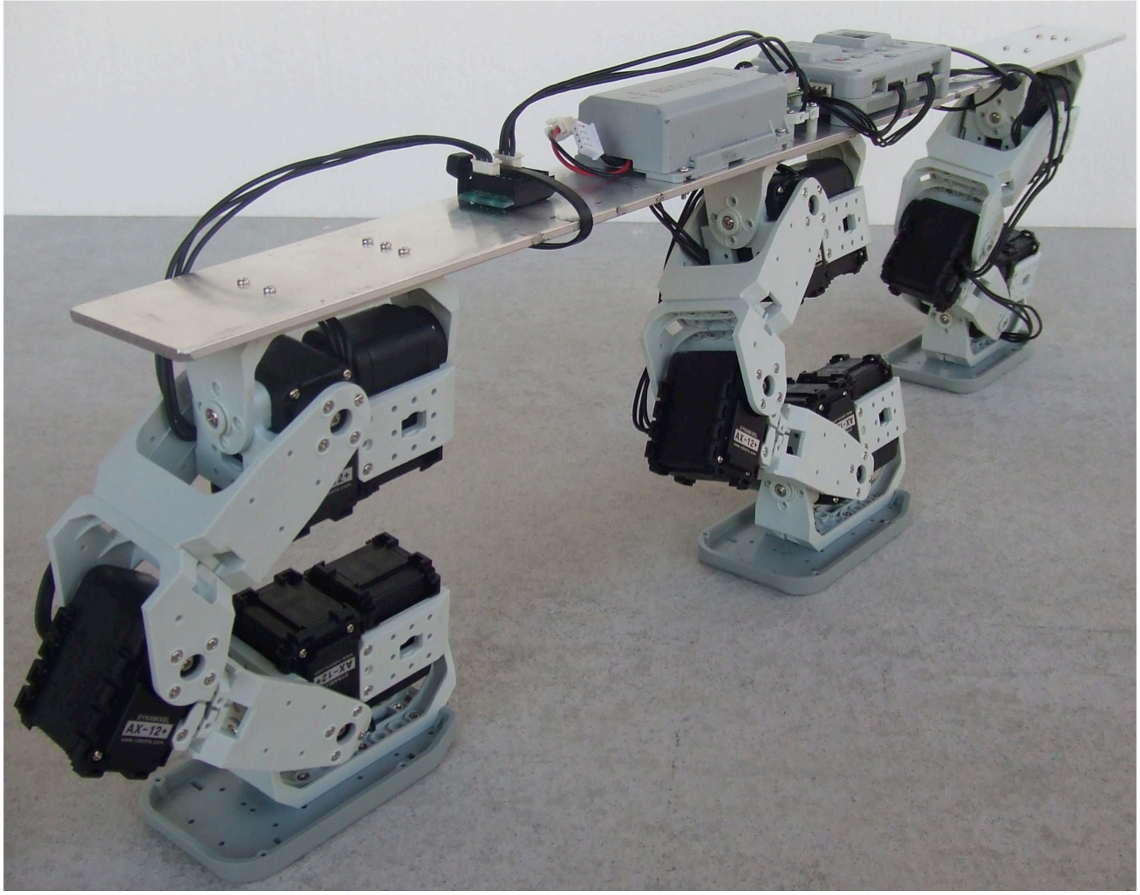


Figure 8.1. The single track three legged mobile robot prototype in perspective view (see figure 8.2 for top and side views).

It was found that having the AX-12+ actuators distributed throughout the leg at each joint imparted an unacceptable torque in the roll axis when the leg was swung to either side. Consequently, such a leg design is only suitable for straight-line motion. A second-generation leg design is discussed in the next section. The leg is a coupled drive design with actuators located at the hip axis to minimize leg mass (Takita 2000 and 2001).

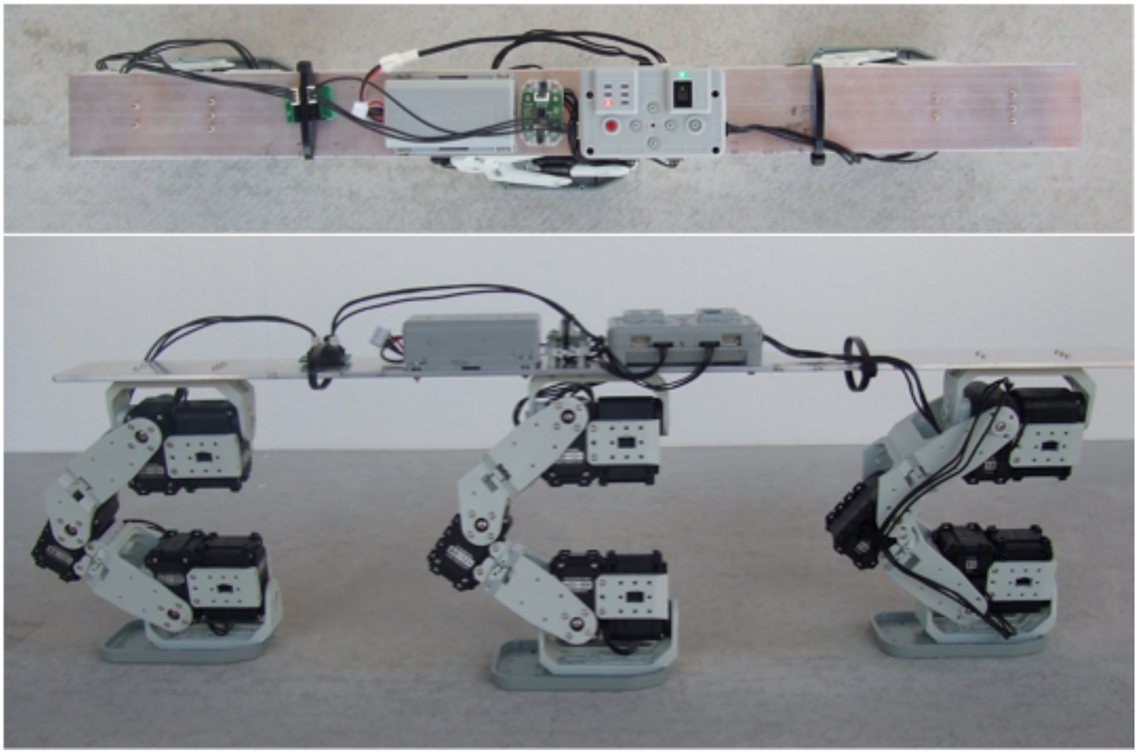


Figure 8.2. Top and side photos of the proof-of-concept prototype robot, demonstrating a stable tripod stance (no forward motion).

8.2. Prototype Draisine Robot

In the last section, the first Proof-of-Concept robot demonstrated the feasibility of straight-line motion for the in-line three-leg design, and it also highlighted the need for an improved leg design. Specifically, reducing leg mass and improving foot repositioning speed was thought to be key to stability of balance for an unstable robot design. This chapter discusses alternative leg mechanisms, details the leg selection process, and presents the design of a second robot and digital simulation. It was found that electric actuators (electric motors with gear reducers and position/angle sensors) are desired for their ready availability of components of various capacity, mechanical packaging, and ready interface to common control signals, whereas both hydraulic and pneumatic actuation systems are cost prohibitive and complex to construct.

8.2.1. Leg Selection

Chapter 2 provides a background discussion of various leg mechanisms, including the coupled-drive leg mechanism. A decision matrix was used to select the best leg type from the aforementioned jointed, coupled, prismatic, and pantograph mechanisms. Table 8.1 presents the leg type selection matrix. Such decision matrices are used in engineering design to help eliminate bias (Ullman 1992).

The vertical columns are the different types of leg designs considered: actuators on joints, bi-articulate actuators, coupled-drive, prismatic, and pantograph. The rows are the different criteria (six mechanism, four control, and four fabrication) derived from the QFD analysis in Chapter 1 and from the ST3LMR design requirements in Chapter 7. The center portion of the decision matrix is the comparative ranking of how "worse than" (-) or "better than" (+) a design alternative is based on engineering judgment. The scores range between "very much worse than" (-3) to "very much better than" (+3).

For example, the first row considers a "lightweight leg" as a mechanism design criteria. The Actuators-on-Joints leg type is the reference or baseline design, and a zero (0) is entered. For the same size leg, the Bi-Articulate Actuator leg would be heavier, and a negative one (-1) is entered. A Coupled-Drive leg, on the other hand would be much lighter in weight, and a two (+2) is entered. The analysis continues with the Prismatic and then Pantograph legs, and then repeats for the next design criteria.

Table 8.1. Leg type selection matrix.

	Weight	Actuators-on-Joints	Bi-Articulate Actuation	Coupled-Drive	Prismatic	Pantograph	Notes:
Mechanism							
Lightweight Leg	15	0	-1	1	2	1	Especially the calf
Number of Actuators	5	0	-1	0	0	0	3-Axis = 0
Actuator Force-to-Weight	5	-2	-1	1	0	-1	Size, Weight, and Power (SWAP)
Mechanical Advantage	10	1	2	1	0	1	Gearbox, kinematics, other
Imparted Force or Torque	5	-1	-1	1	0	-1	To "catch a fall"
Device Integration	5	0	-2	0	-1	-2	E.g., Support housing reuse
Control							
Runs on Simple Microcontroller	5	0	-1	0	1	1	8051, Aurdrino, BasicMicro, etc.
Number of Servo Updates Required	5	-1	-2	-1	0	0	Per x-axis unit
Math Complexity	5	0	-1	1	2	-1	Forward and inverse kinematics
Foot Trajectory Flatness	10	0	-1	1	2	2	Per x-axis unit
Fabrication							
Off-the-Shelf Availability	10	0	-2	0	-2	-3	Parts availability
Weeks to Prototype	10	0	-2	-1	-1	-3	Days would be better
Scalability	5	0	-1	0	-2	-2	Similitude, 1/X to full-size
Use Common Shop Tools	5	0	-1	0	0	-1	Band saw, drill press, hand tools
Total:	100	-3	-14	3	-1	-10	
Weighted Total:		-10	-100	35	20	-50	

Finally, the design criteria importance is weighted (scaled to 100) based on engineering judgment. The coupled-drive leg mechanism has the highest weighted score and is selected for the following eight reasons:

1) The leg is lightweight and places the knee actuator (motor and gear box) at the Hip, connecting it to the Calf via a simple linkage;

2) The Calf actuator uses a lever arm which proportionally increases the position and velocity of the Foot;

3) The Hip actuator directly connects to the thigh, wherein the actuator bearings provide leg support without additional weight;

4) The kinematic linkages enables sharing of the vertical and horizontal forces between the Hip and Calf actuators, within a working range, thus increasing the resulting forces per actuator weight;

5) Rotary actuators (motor and gear box) are size, weight, and energy efficient and commonly found in many sizes from small RC servos to high-torque hydraulic and very high speed pneumatic motors;

6) Non-back-drivable gear boxes are commonly available for rotary motors, and hydraulic and pneumatic motors are naturally non-back-drivable;

7) RC servos and serially controlled actuators are readily available off-the-shelf; and

8) Many microcontrollers incorporate math co-processors necessary for computing the trigonometric functions in forward and inverse kinematic equations.

8.2.2. Coupled-Drive Leg Development

This section presents details on the leg kinematics and parameter optimization for the second prototype robot. A cardboard mock-up of the coupled drive leg was constructed based on (Takita 2000) as a proof-of-concept. Figure 8.3 shows the leg mockup, illustrating two leg positions. The mockup also tested two Hitec HS-M7990TH servos (not shown in the figure). Figure 8.4 illustrates the components of the leg. Link AB is the Thigh, and it directly connects to the first actuator (M) located at point A. Link CD and DE couples the Calf (denoted BEF) to the second actuator (M) located at point C. The Foot is point F. The x-axis is horizontal and positive right, the z-axis is vertical and positive down, and the y-axis is in the page and positive out of the page.

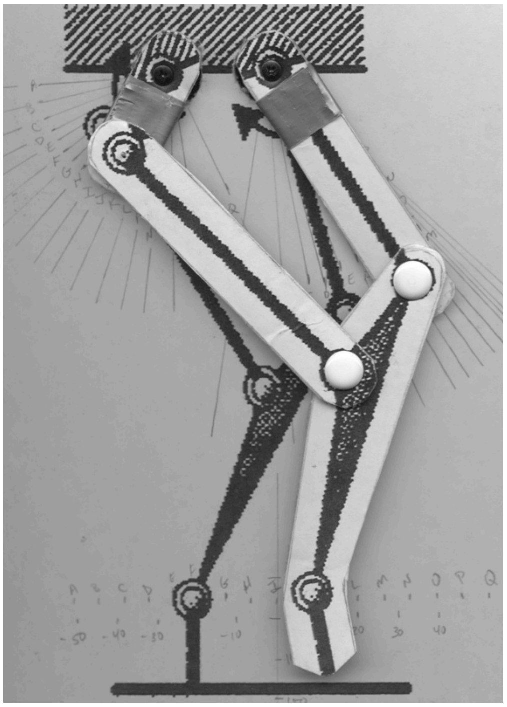


Figure 8.3. Simple cardboard mockup of the coupled-drive leg.

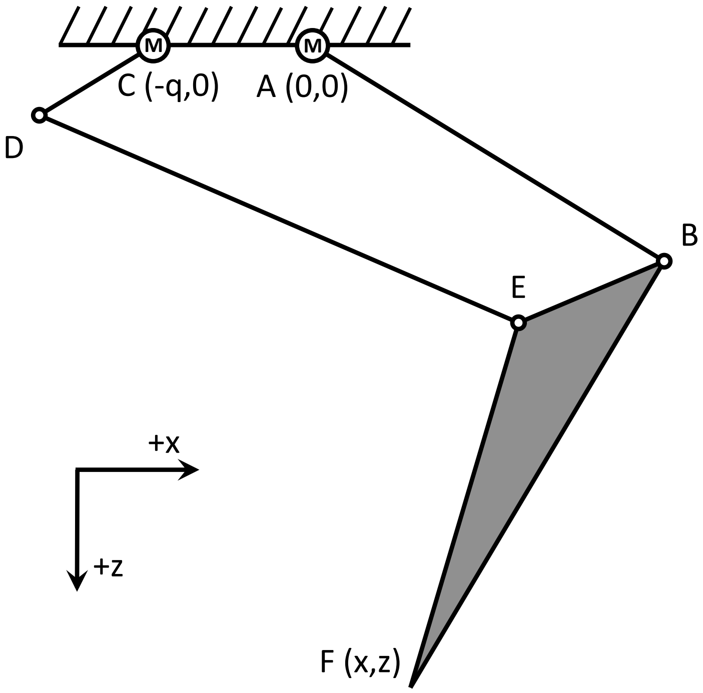


Figure 8.4. Schematic of the coupled-drive leg mockup.

Through the forward kinematics equations, the position of the foot may be calculated from the measured actuator or motor angles. Figure 8.5 lists MATLAB pseudo code for computing the position of the Foot, where a single letter denotes a position, two letters denote a distance, and three letters denote an angle. For example, angle BA0 denotes the angle at point A from joint B to the horizontal axis, 0 (zero). For convenience, the center of rotation of the first motor is made the origin for the leg system. The second motor is negatively displaced along the horizontal x-axis. The third axis of rotation is about the x-axis, but it is not shown for clarity.

```
% Constant angle EBF
EBF = acos(-(EF^2 - BF^2 - BE^2)/(2*BF*BE))
------------
% Compute points B and D
B = A + AB*[cos(BA0) sin(BA0)]’
D = C + CD*[cos(DC0) sin(DC0)]’

% Compute referenced angle FB0
BD = sqrt((Bx - Dx)^2 + (Bz - Dz)^2)
DBE = acos(-(DE^2 - BD^2 - BE^2)/(2*BD*BE))
BD0 = asin((Bz - Dz)/BD)
EB0 = pi - DBE + BD0
FB0 = EB0 - EBF

% Compute point F
F = B + BF*[cos(FB0) sin(FB0)]’
```

Figure 8.5. Coupled-drive forward kinematic MATLAB pseudo code.

Through the inverse kinematics equations, the motor angles may be calculated from the desired position of the Foot. Figure 8.6 lists MATLAB pseudo code for computing the motor angles BA0 and DC0 given the position of the Foot, F. Again, a single letter denotes a position, two letters denote a distance, and three letters denote an angle.

In computing the forward kinematics, the position of joints B and D are computed first, using the trigonometric sine and cosine functions. Then, referenced angle FB0 is computed from the inscribed angle DBE, referenced angle BD0, and constant angle EBF. The position of the Foot F is then computed using the sine and cosine functions.

```
% Constant angle BFE
BFE = acos((BF^2 + EF^2 - BE^2)/(2.0*BF*EF))
------------
% Compute point B
AF = sqrt(Fx^2 + Fy^2)
ABF = acos((AF^2 + BF^2 - AB^2)/(2.0*AF*BF))
BF0 = pi - acos(abs(Fx)/AF) - ABF
B = F + BF*[cos(BF0) sin(BF0)]’

% Compute angle BA0
BA0 = atan(By/Bx)

% Compute point E
EF0 = BF0 + BFE
E = F + EF*[cos(EF0) sin(EF0)]’

% Compute angle DC0
CE = sqrt((Ex-Cx)^2 + (Ey-Cy)^2)
DCE = acos((CE^2 + CD^2 - DE^2)/(2.0*CE*CD))
DC0 = DCE + asin((Ey-Cy)/CE)
```

Figure 8.6. Coupled-drive inverse kinematic MATLAB pseudo code.

In computing the inverse kinematics, the desired position of the Foot, F, in the x- and z-axis is first used to compute the position of joint B, which then resolves the motor angle BA0. From the position of joint B, the position of joint E is determined, from which the inscribed angle DCE is computed. Motor angle DC0 is then computed from the position of joint E plus the inscribed angle DCE.

8.2.3. Coupled-Drive Leg Parameter Optimization

One advantage of the prismatic and pantograph leg mechanisms is that a single actuator determines the position of the Foot in either the vertical or horizontal position. For a desired constant velocity, the actuator is run at a constant velocity. This property is not the case for jointed legs. Figure 8.7 illustrates the non-linear trajectory of the coupled-drive mockup Foot, given constant velocity of the hip and calf actuators. The Foot exhibits a 0.14-inch peak-to-peak change over a +/-1.25-inch working range. To achieve a flat trajectory, the Hip and Calf actuators must change velocity over the working range.

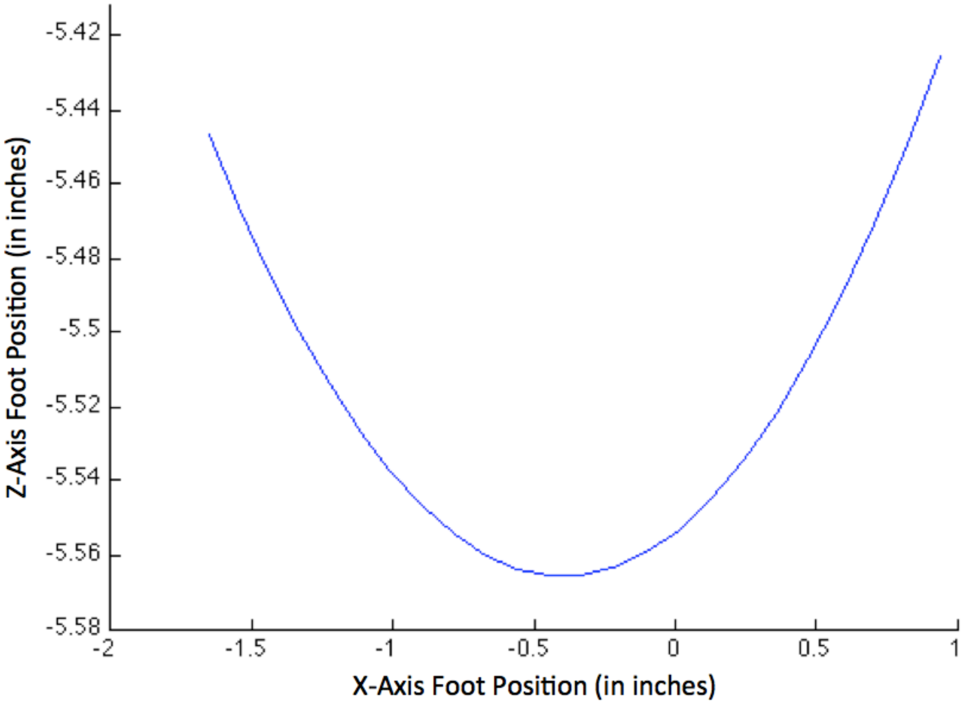


Figure 8.7. Computed trajectory of the coupled-drive mockup foot.

To achieve a flat trajectory using a digital control system, the position of the Foot and the velocity of the actuators must be computed over many discrete points to approximate straight-line motion of the Foot. The number of points that can be computed is a function of the digital computer throughput and how many other tasks (e.g., reading sensors, computing roll velocity, managing the leg states, positioning multiple legs, and so on) that it must concurrently perform. So, optimizing the leg for the flattest trajectory possible is desired to minimize the number of piece-wise calculations that must be performed.

A Monte Carlo optimization routine was added to the forward kinematic equations to change the design parameters of the leg to minimize Foot travel in the vertical axis. The leg design parameters include the location of the Calf actuator along the x-axis, the leg lengths (AB and BF, as illustrated in Figure 8.4), the position of the Calf joint (E) with respect to the knee joint (B) and foot (F), and lengths of the two kinematic couplings (CD and DE).

Figure 8.8 illustrates two different families of curves produced through Monte Carlo optimization, given constant velocity of the Hip and Calf actuators, over a horizontal and vertical working range. The curves are generally parabolic and change the position of the peak along the horizontal for different vertical lengths. As the peak moves to the lower left, the trajectory takes on a sigmoid shape. Through brute-force optimization, a family of curves was discovered that yielded the flattest vertical trajectory, given constant actuator velocity.

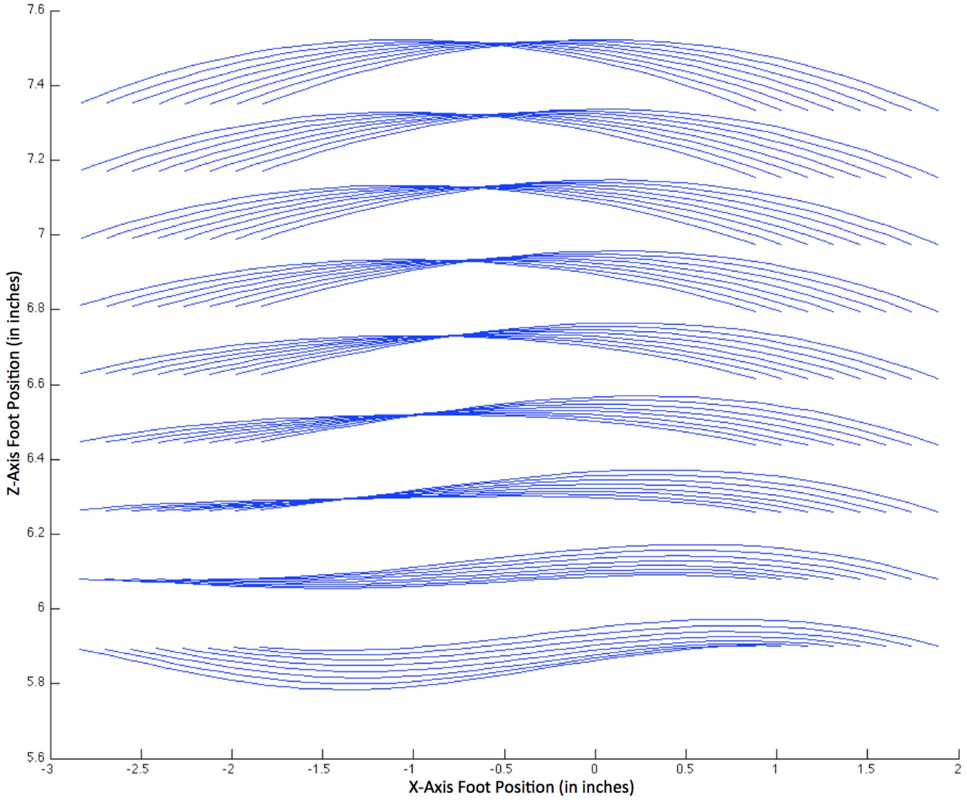


Figure 8.8. Example Monte Carlo trajectories of the coupled-drive foot.

Figure 8.9 shows the trajectory of the coupled-drive leg using optimal parameters (left) and a schematic of the leg (right) illustrating a piece-wise flatness of +/-0.024-inches per 0.28-inches of horizontal travel. A total of 31 position updates are computed using the inverse kinematic equations for 6.5-inches of horizontal travel (+/-3.25-inches from zero x-axis), given a 1/3-scale, 8-inch tall leg. The algorithms for the coupled-drive leg mechanism were incorporated into the 3-leg simulation, described in Chapter 5. Figure 8.10 illustrates the ST3LMR with coupled-drive legs, in a simulated backward wave gait, sequenced top to bottom (1 to 7), and showing continuous lines of two-leg support (dash-dot lines).

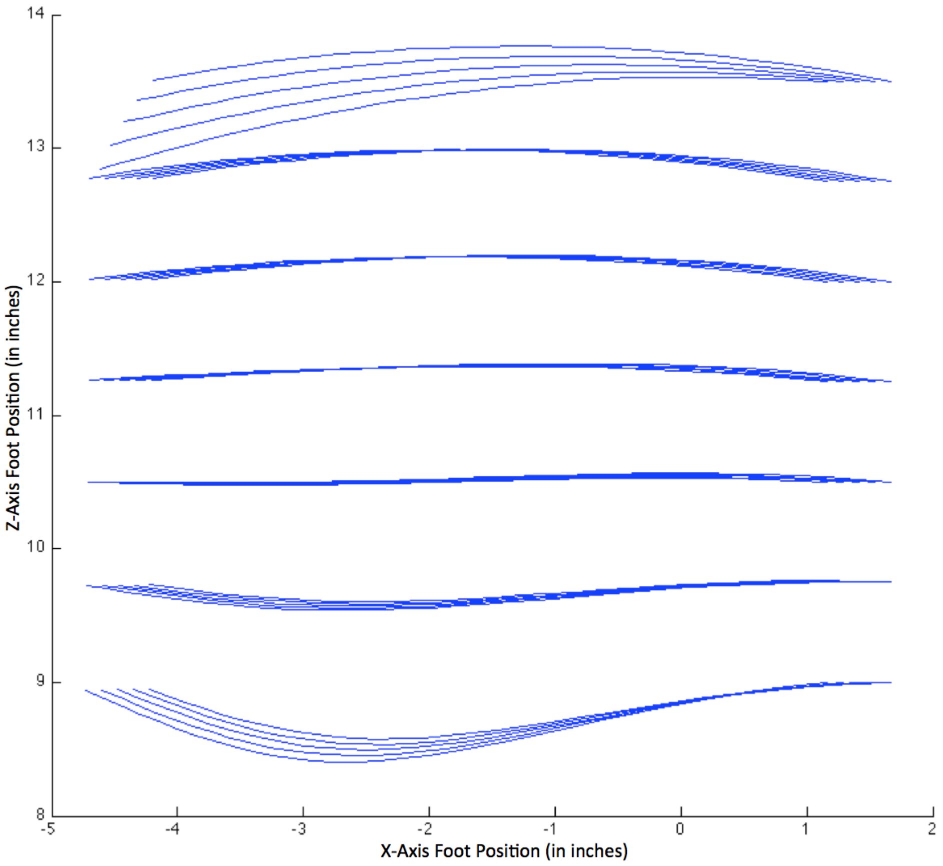


Figure 8.9. Trajectory of the coupled-drive foot after parameter design.

The leg design was most sensitive to the rotation angle range of the Calf actuator and its extension arm (CD, as illustrated in Figure 8.4), at the minimum and maximum limits of vertical travel. To this end, the Calf actuator was limited from traveling more than 175-degrees of rotation, to prevent interference between the linkage (DE) and the center of rotation of the Calf actuator. This is thought to be a scalable design. In summary, the finished robot stands 7.0-inches (177.8-mm) tall at the hip and weighs 4-lbs 3.55-oz (1.915-kg) with a center of mass 6.45-inches (163.8-mm) from the ground, which includes the “training wheels”.

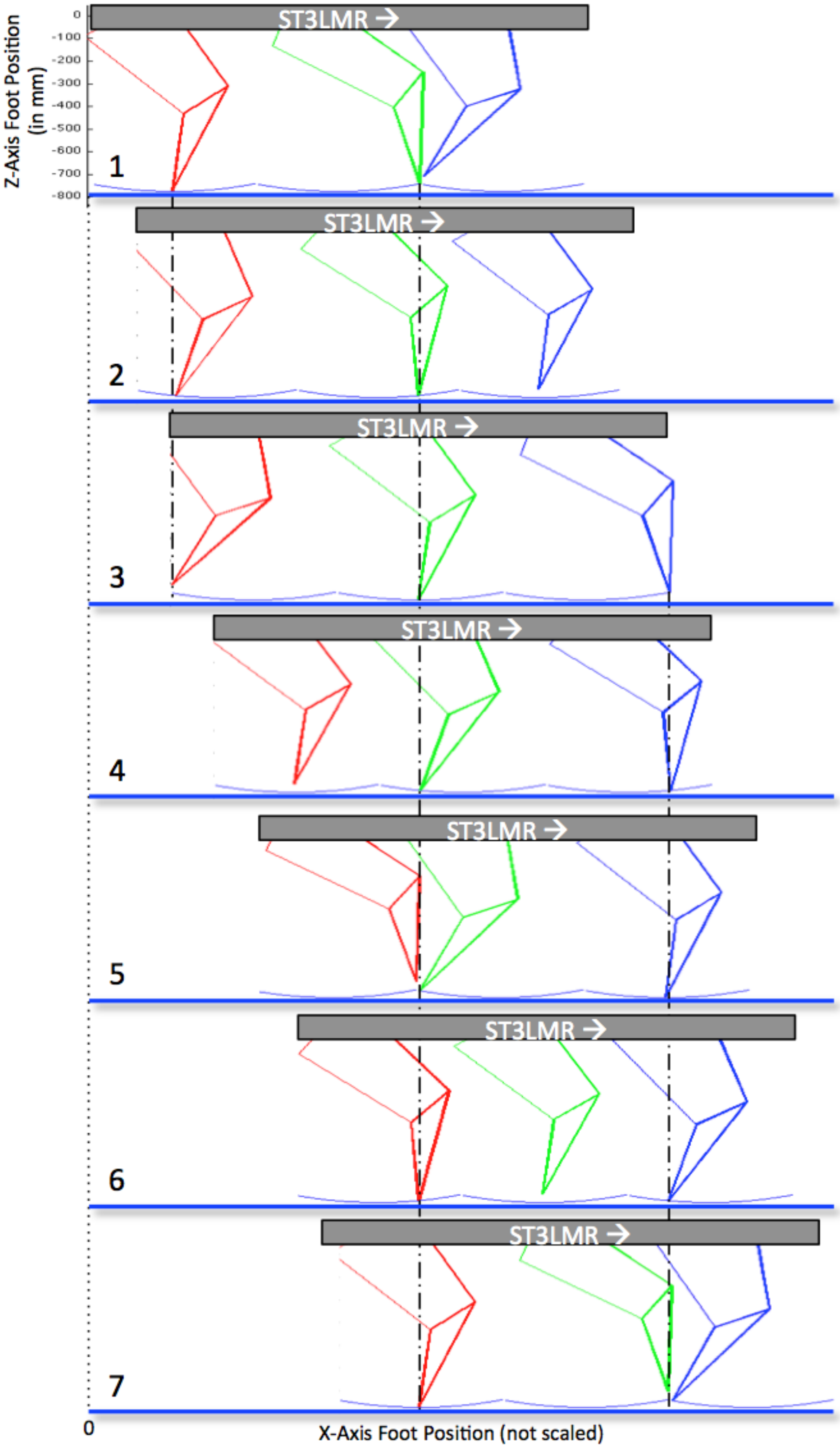

ST3LMR →
Z-Axis Foot Position (in mm)
0
-100
-200
-300
-400
-500
-600
-700
-800
1
2
3
4
5
6
7
0
X-Axis Foot Position (not scaled)

Figure 8.10. MATLAB simulation of the coupled-drive leg ST3LMR, illustrating a backwards wave gait, sequenced top to bottom.

8.2.4. Motor-Controller Selection

Whereas the proof-of-concept ST3LMR legs were constructed from off-the-shelf AX-12A+ Dynamixel servos and parts from the Bioloid Robotis kit, the Draisine robot leg is scaled to maximize performance of the off-the-shelf servos. Figure 8.11 charts the wide range of speed versus maximum stall torque for the Bioloid Dynamixel family of robotic servos. Stall torque relates to the robot's ability to support its weight, whereas speed relates to the ability to (quickly) reposition the leg.

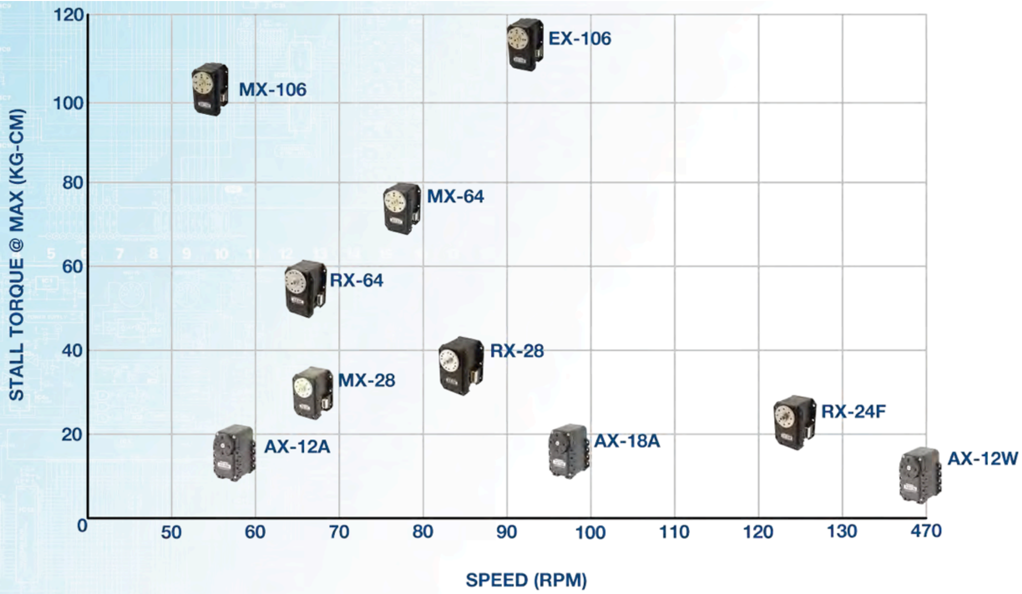


Figure 8.11. Speed versus maximum stall torque for Bioloid Dynamixel servos. (Image reproduced curtsey Trossen Robotics web site.)

Another popular type of off-the-shelf actuator is the Radio Controlled or RC servo. Whereas the Dynamixel servos input a serial command to control position and velocity, and several models employ a tunable, internal PID (Proportional-Integral-Differentiable) control algorithm, the RC servos input a pulse-width modulated (PWM) signal and operate open loop using proportional control. The RC servo control method typically requires changing the PWM signal slow enough for the servo to "follow" the control signal. It is assumed that the servo has been properly sized (torque and speed) to keep up with the control signal.

The highest performing Hi-Tech HS-M7990TH RC servo actuator is compact in size at 1.72 x 0.88 x 1.57-inches (43.8 x 22.4 x 40.0-mm), low in weight at 2.76-oz (78.2-grams), has a stall torque of 611-oz-inch (44-kg-cm), and has a no load operating speed of 58.8-rpm at 7.4-vdc. Given the aforementioned leg, this RC servo could reposition the Calf 175-degrees to translate the Foot 6.5-inches in roughly 1/3-second. The comparable RX-28 Dynamixel servo is slightly larger at 1.99 x 1.40 x 1.40-inches (50.6 x 35.6 x 35.5-mm), slightly lower in weight at 2.54-oz (72.0-grams), has a stall torque of 523.55-oz-inch (37.7-kg-cm), and has a no load operating speed of 84.4-rpm at 16-vdc. Given the aforementioned leg, this Dynamixel servo could reposition the Calf 175-degrees to translate the Foot 6.5-inches in roughly 1/4-second. Both servos cost about the same.

A further consideration is the selection of a microcontroller with a math co-processor for accurately computing trigonometric functions. The prototype ST3LMR robot, used the Dynamixel CM-2+ controller, which is based on the Atmel ATmega 128 microcontroller. The ATmega is a modified Harvard architecture 8-bit RISC microcontroller with a variety of built-in features, such as 8-bit and 16-bit timers for PWM, 12-bit analog-to-digital converter, and serial port controllers, but it does not have a math coprocessor. Third-party controllers designed for the Dynamixel's serial communication port are also available, and use the ATmega or PIC microcontrollers.

None were found to have math coprocessors. An alternative to an on-board microcontroller is using a laptop computer with a radio link for serial communication. Such a design would offer maximum computational horsepower, but it would not be in keeping with the spirit of a (stand-alone) robot.

On the other hand, various microcontrollers exist for controlling RC servos. Many are based on the ATmega and PIC microcontrollers. The BasicATOM Pro 40-M controller is based on the Renesas (Hitachi) HD64F3687GFPV 32-bit microcontroller operating at 20-MHz. It has a variety of built-in features, such as an RS232 port, 8-bit analog-to-digital converter (which can multi-sample to 12-bit), and 32 servo background controllers. Most important, it has a 32-bit integer math and 32-bit floating-point math coprocessor. It is for this last feature that the BasicATOM Pro 40-M and Hi-Tech HS-M7990TH were chosen for the second prototype robot. Furthermore, off-the-shelf high-voltage, high-current RC servo controllers exist that could be paired with liquid cooled motors, similar to how the University of Tokyo JSK Lab's battery-capacitor leg is constructed, which provides a path to build a full-scale prototype.

8.2.5. Robot System Fabrication

The Draisine (wheel-leg-wheel) robot design was conceived of as a compromise due to fabrication costs. Only one leg is needed to test certain principles of dynamic stability of the Single-Track Three Legged Mobile Robot. Further, the degree to which a rider provides balance control of the roll axis (like a motorcycle) versus the robot providing stability of balance (like a horse) is the subject of future research. Observing such fundamental rider-based principles using a three-leg platform would only complicate testing and slow research. Figure 8.12 shows a perspective view of the one third-scale (8-inch leg) second-generation Draisine prototype robot, with adjustable "outrigger-style training wheels" to prevent fall-over. Two steering actuators with gear reducers, visible at either end of the top of the frame, are used for turning the wheels.

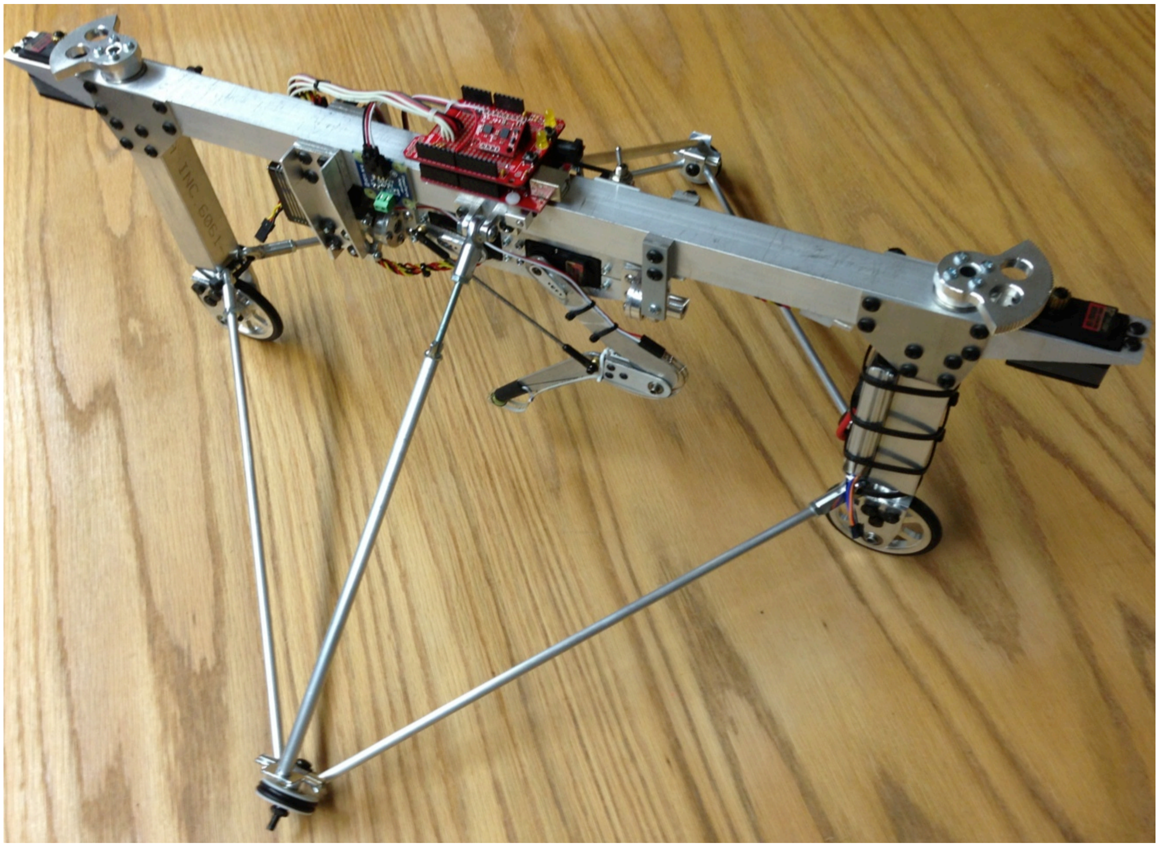


Figure 8.12. The single-track Draisine (wheel-leg-wheel) mobile robot prototype in perspective view.

Based on the stall torque of the servos, the nominal leg height was calculated (scaled) to be 7.0-inches (177.8-mm) to support a weight of 4-lbs (1.814-kg) at an inclination of 30-degrees. Figure 8.11 shows the finished coupled-drive leg of the second prototype robot. The leg is fabricated from aluminum stock, with the two high-performance Hitec HS-M7990TH servos directly connected at the hip and indirectly through a pushrod coupled to the calf. The two servos position the leg along the length (x-axis) and in height (z-axis). A third servo, seen in the upper left of Figure 8.13, rotates the Hip assembly to position the foot outward (y-axis).

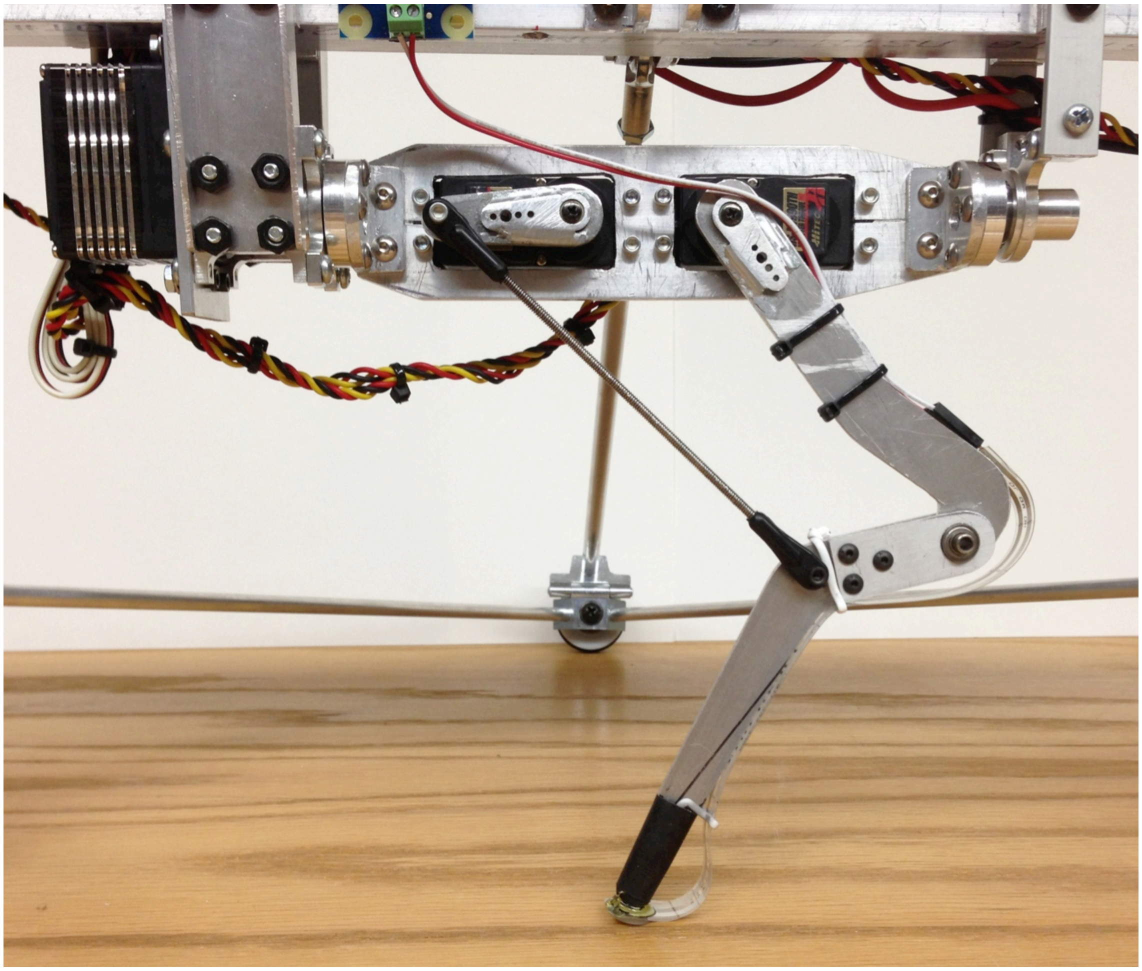


Figure 8.13. Finished coupled-drive leg of the Draisine prototype robot.

The BasicMICRO "Mad Hatter" microcontroller was selected for its two built-in serial ports, independent servo/PWM controllers, 8-bit analog-to-digital converters

(ADCs), and built-in 32 bit floating point math coprocessor (to perform trigonometric functions). The robot design also includes the selection of several off-the-shelf sensors. A FlexiForce 0-1 pound piezoresistive pressure sensor is attached to the bottom of the foot, and uses a FlexiForce adapter to interface to the microcontroller. A SparkFun Atomic 6 degrees of freedom IMU (three-axis Freescale MMA7361L MEMS accelerometer with 1.5-g sensitivity and ST Microelectronics LISY300AL MEMS gyro with 300-degrees/second sensitivity) are mounted on top of the microcontroller to sense the roll axis. Figure 8.14 provides additional views and details of the prototype single-track Draisine (wheel-leg-wheel) robot.

The Draisine prototype robot is tested through a series of increasing complexity experiments. The algorithms, software, test methods and data may be found in Appendices B through D. Specifically, Appendix B presents the development and experiments pertaining to the coupled-drive leg, and Appendix C presents the development and experiments pertaining to the Draisine robot. Verification and validation of the design is done in two ways: 1) through direct measurement of the physical model and 2) through comparison with a digital simulation.

In summary, the shortcomings of the first prototype – namely the heavy legs – are overcome with a new design using the coupled-drive leg mechanism. Electric servomotors were chosen because they are readily available, low cost, easier to work with than pneumatics, and less dangerous than high-pressure hydraulics. Electric motors can be scaled from very small to very large sizes. Small servos are available off-the-shelf, and can be controlled with a wide variety of existing microcontroller to computer systems.

Further, the University of Tokyo JSK Lab's work in developing a liquid-cooled, high-torque, high-speed leg actuator using a battery-capacitor control system lends credence to and provides an upgrade path to scale the prototype to a full-size (22-inch leg) design. Recall, the goal of this work is to develop a robot that achieves the form, function, and convenience of a motorcycle but with the added benefit of legs and partial or fully automatic stability of balance (like a horse).

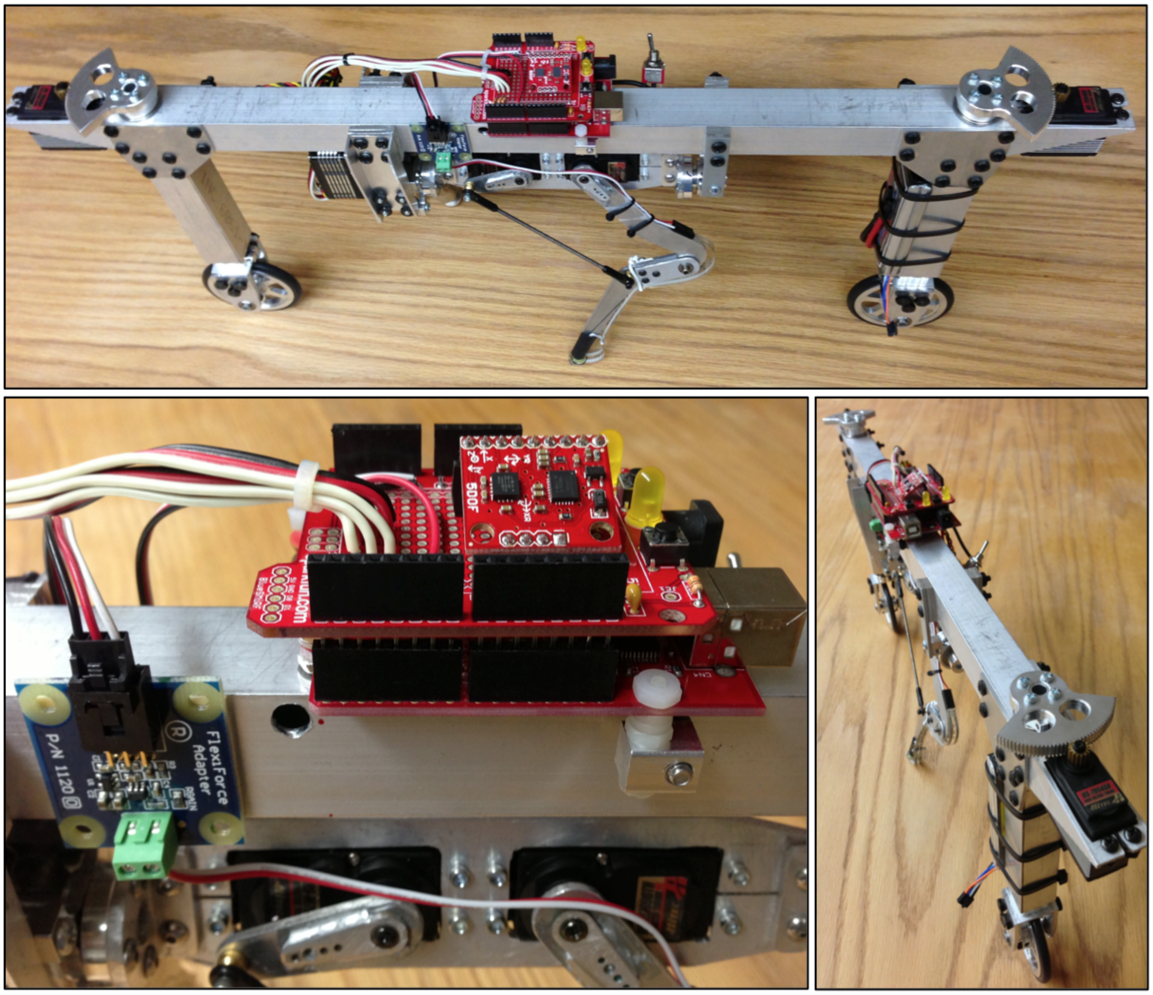


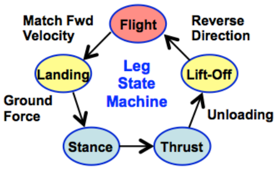
Figure 8.14. Additional views of the prototype single-track Draisine (wheel-leg-wheel) mobile robot.

CHAPTER 9

RESEARCH SUMMARY

This research and dissertation contributes a novel Single-Track Three Legged Mobile Robot (ST3LMR) and method of operation to advance the art of legged robot design. As such, it opens up new research opportunities in robotics and human-robot interface design. While this research did not realize a full-scale ST3LMR that people ride, it nonetheless provides an incremental step towards a ST3LMR that people would ride like a motorcycle. This chapter provides a summary of this dissertation and a vision statement. Table 9.1 summarizes the research.

Table 9.1. Research Summary.

Feature	Description	Research	Requirements
Basic Legged Locomotion (Proof-of-Concept Prototype)	Gate coordination repeats intervals of the bipedal and tripod stance for stability of balance	 Match Fwd Velocity Flight Reverse Direction Landing Leg State Machine Lift-Off Ground Force Unloading Stance Thrust	• Legged robot design • Forward / inverse kinematics • State machines • Proof-of-Concept Prototype
Self-Recovery from Slips and Falls (Draisine Prototype)	Place a leg in the direction of fall to regain support	Foot position / hip angle: $y_f = \frac{\dot{y}T_S}{2} + k_{\dot{y}}\left(\dot{y} - \dot{y}_d\right)$ $\gamma_d = \phi - \arcsin\left(y_f / L\right)$	• Dynamic systems • Inverse pendulum model • 2 DOF Simulation (x, roll)
Stability of Balance (Simulation and Prototypes)	Control foot (x, y), velocity, gait aperiodic tripod stance	Model-based predictive control & gait coordination	• Impulse & S response • Adaptive gain, $K_{\text{roll-rate}}$
Single-Track Turn (Simulation and Draisine Prototype)	Lean body into turn, develop a torque about the roll axis, balance centripetal force	Multiple motion models (e.g., straight and curved)	• Hardware Prototype and Simulation (x, y, roll, yaw)

9.1. ST3LMR Proof-of-Concept Summary

This dissertation developed a three-legged robot having the legs arranged in a minimally narrow profile to place successive footfalls in a predominately single-track or in-line fashion, similar in form and function to motorcycles and bicycles. In off-road environments, such as parks and wilderness areas, single-track vehicles, such as motorcycles and bicycles, exhibit superior maneuverability and deployment performance in comparison to double-track vehicles, such as automobiles and tanks. It is generally held that motorcycles are typically lighter in weight, have fewer mechanical components, increased reliability, higher energy efficiency, and faster acceleration and deceleration than four-wheeled vehicles. By this analogy, it is reasoned that an in line three-legged vehicle would share all of the benefits and advantages afforded to motorcycles over that of the quadruped (e.g., four legs arranged in a rectangular configuration) double-track legged vehicle.

This dissertation also developed a control system architecture that uses sensed data to predict the future state of the robot (as in body roll and leg/foot placement). When a person rides a motorcycle, the rider looks ahead for changes in the road, such as for example a curve or turn in the road, the rider plans the appropriate control strategy before the motorcycle reaches the turn, and the rider leans the motorcycle into the turn before the road begins to curve. It is this type of anticipative control strategy that a legged robot should perform if the legged robot is to operate and be ridden like a motorcycle. Furthermore, this dissertation develops an automatic system for sensing and preventing turnover of single-track legged mobile robots while enabling normal riding techniques in all but out of control situations. An out of control situation, for example, is when the robot is falling over and can no longer be uprighted by repositioning the feet, or when the heading of the robot cannot be changed without the robot falling over.

9.2. Contribution Statement

This research realized a novel three-leg robot vehicle and control system design, called the Single Track Three-Legged Mobile Robot or ST3LMR. A proof-of-concept was demonstrated through digital simulation and experimentation with physical prototypes. In summary, the author has contributed the following:

1) Invented a new type of legged robot with an in-line or single track leg configuration that achieves the desired form and function of a motorcycle but with the added benefit of legs and full or partial robotic control;

2) Invented new gaits for three in-line or single-track leg locomotion whereby repeated intervals of a bipedal and/or tripod stance provide for roll and pitch control;

3) Decoupled roll and pitch control using the three-legged gaits to simplify the control requirements of legged robots through separate algorithms for forward motion (gaits) and stability of balance (in the roll axis);

4) Architected a two-time-scale Monte Carlo type model-based temporal robot control system to plan the footholds and provide autonomous attitude stabilization control;

5) Demonstrated maneuverability and control through experimental results from physical prototypes and a simple digital simulation of an impulse response, avoidance maneuver, and lean-into-the-turn cornering;

6) Demonstrated stability of balance during locomotion, by repeating intervals of dynamic momentum (including the monopod stance) followed by the roll-stable bipedal and/or the fully stable tripod stance; and

7) Demonstrated a lean-into-a-turn maneuver (with respect to the body and primary direction of motion) for a legged mobile robot wherein stability of balance is controlled in the roll direction and not in the pitch direction and torque is developed about any two or three legs during the stance phase to control the direction of motion in the yaw axis.

The two-time-scale Monte Carlo type model-based temporal control system demonstrates mobile robot balance and path planning. The architecture includes a sensor module for receiving road preview data corresponding to the spatial locations of foothold areas located ahead of the robot. A temporal control module coordinates the legs/feet motion based on predicted future locations of foothold areas, predicted future robot balance or stability state, and the desired or control command. A temporal simulation module is operatively coupled to the temporal control module.

The temporal simulation module is configured to use the predicted future locations of at least one foothold area to simulate multiple robot motion hypothesis for leg/foot placement and trajectory planning. The temporal simulation module operates faster than real-time to enable robot state preview, and in this fashion it can correct for real-world (i.e., sensed) differences between the predicted and actual states. The control algorithm uses probabilistic modeling and simulation to produce temporal-based action commands to control and coordinate the legs, all in the presence of missing data, latency, translational bias, and/or sensing error.

The ST3LMR design and control method is highly important so that an inherently unstable legged robot can achieve stability of balance and maneuver over rugged terrain. Recall that as terrain becomes more rugged, continuous paths of support become less frequent to the point where only discrete, unevenly spaced areas of support exist in the most rugged terrain (Holste and Ciccimaro 2009). As such, it opens up new research opportunities in robotics and human-robot interface design that is potentially viable on several fronts – technical, commercial, and social. Further, this research provides an incremental step towards a ST3LMR that people could ride like a motorcycle.

9.3. Vision Statement

The ST3LMR is a robot that achieves the form and function of a motorcycle but with the added benefit of legs and partial or fully automatic stability of balance (like a horse), as depicted in Figures 9.1 and 9.2. The ST3LMR comprises a body and three articulated legs arranged one behind the other in a narrow profile to walk and maneuver along narrow trails and paths, by placing successive footfalls in a generally single-track or in-line fashion. The term 'single track' refers to the general narrowness of the foot-placement patterns developed on the ground when moving along a straight-line or curved path. The ST3LMR achieves stability of balance without motion by positioning its three legs in a tripod stance, as depicted in Figures 9.1 and 9.2. During locomotion, the feet are dynamically placed to the left or right of the support line to maintain (regain) stability of balance. Like the motorcycle, the ST3LMR executes a single-track turn by leaning the body into the turn, using gravitational forces to counteract the outward centripetal force. Unlike a motorcycle, the feet of the ST3LMR are stationary with respect to footholds during the support period, thus eliminating the drawback of wheels, which require continuous support (especially in rugged terrain). While not part of this research, it is envisioned that hopping, bounding, leaping, and jumping would enable the ST3LMR to traverse terrain that is too difficult for comparable wheeled and tracked vehicles.

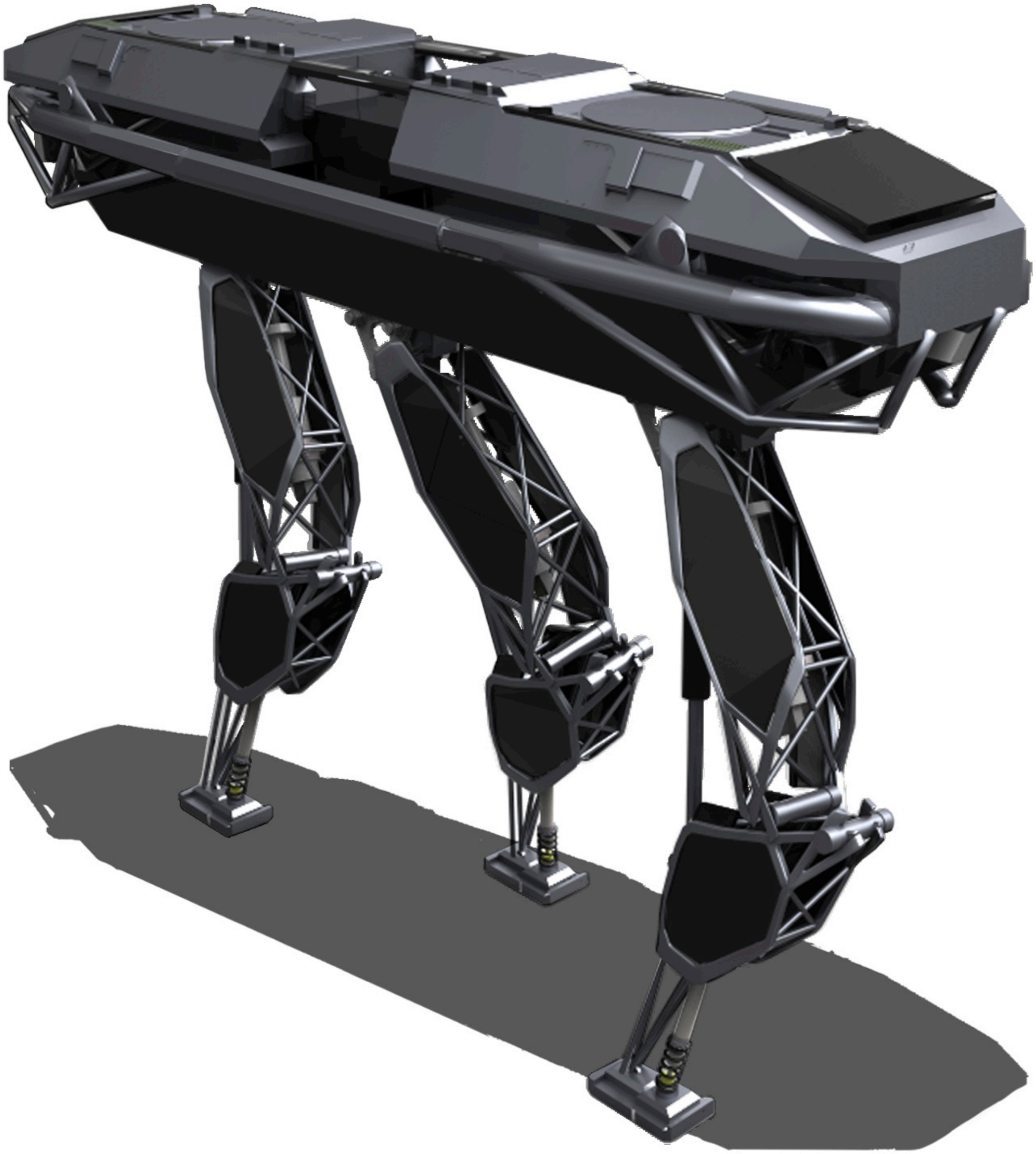


Figure 9.1. Conceptual drawing of the Single-Track Three Legged Mobile Robot.

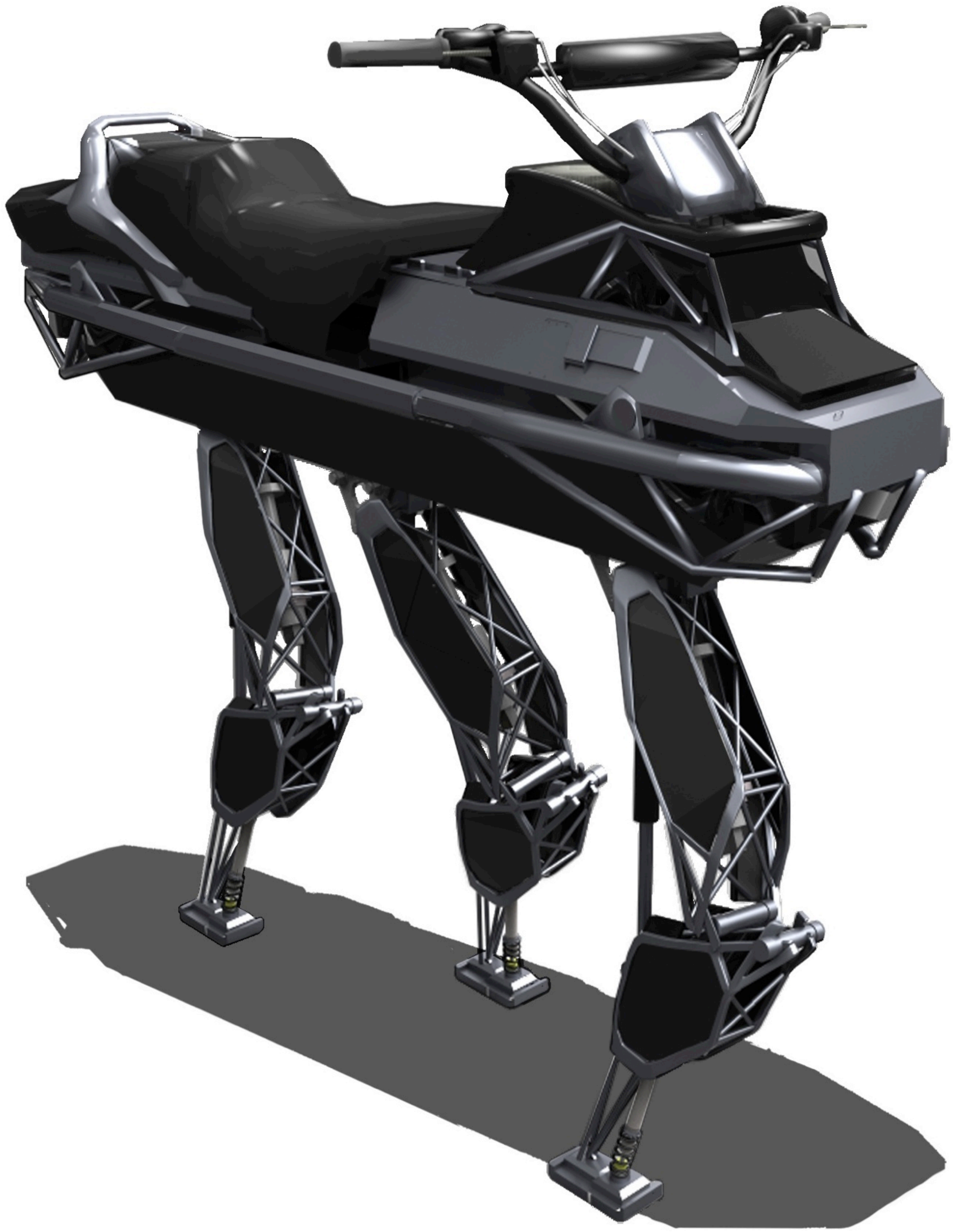


Figure 9.2. Conceptual drawing of the Single-Track Three Legged Mobile Robot, configured with seat and handlebars for riders.

9.3.1. Operational and System Context

This dissertation concerned the proof-of-concept of the ST3LMR. However, the ST3LMR is envisioned as a vehicle that people would ride like a motorcycle. (Note that a discussion of liability and other legal and political issues is beyond the scope of this dissertation.) The ST3LMR is envisioned to have the following key operational features, functions, and benefits:

- Carries one rider, an optional passenger, and/or cargo – could be used as a “pack mule” to carry heavy loads and accompany hikers;
- Costs less to manufacture – the ST3LMR uses 3/4 fewer parts (i.e., three legs instead of four legs), and would thus have higher reliability;
- High-speed legged locomotion – Unlike biped and quadruped designs, the tri-leg gate repeats intervals of the tripod stance for stability of balance;
- High-speed turns – Like two-wheeled motorcycles, the ST3LMR executes a single-track turn by leaning the body into the turn to develop a torque about the roll axis to counteract the outward centripetal force;
- Improved human health and emotional development by bringing back the thrill of trail riding;
- Improved ride quality – Single-track legs reduce body motion compared to biped and quadruped designs and wheels in rough terrain (Note: this characteristic is especially well suited to the comfort of a rider, passenger, and/or fragile cargo);

- Inherently stable gait – Three legs enables the ST3LMR to maintain (regain) stability of balance during locomotion, by repeating intervals of dynamic momentum followed by a bipedal or tripod stance;
- Lighter weight, increased reliability, higher energy efficiency, and faster acceleration, due to reduced size and weight as compared to four-legged quadruped robots;
- Long-range capable – Primary power from an internal combustion engine (driving a hydraulic pump/actuators) or a fuel cell (for electric actuators), both having refillable fuel tanks;
- Multiple terrain capability – Interchangeable feet for different conditions, e.g., paved roads, snow (skis), beaches (sand), and so on;
- Operator interface – Receives operator commands (e.g., steering, throttle, and brake) and communicates status through an operator user interface;
- Robotic control system – Relies on riders for high-level control and stability, but also allows semi-autonomous behavior, such as self-guided, GPS-based tours;
- Self-recovery from slips and falls – Able to place a leg in the direction of fall to regain support during operation/motion;
- Simpler control requirements – Unlike biped or two-legged mobile robots that must simultaneously maintain balance in both the pitch and roll directions, the ST3LMR balance is controlled in the roll direction;

• Statically stable stance – Unlike two-wheeled motorcycles, three legs may be positioned in a tripod stance to enable the ST3LMR to achieve stability of balance without motion;

• Traverses man-made obstacles such as curbs, stairs, and narrow passageways – legged vehicles choose optimal foot placement by varying the length of the leg;

• Traverses rugged terrain – Bounding, leaping, or jumping over areas of ground that do not have a continuous path of support or closely spaced footholds; and

• Walk and maneuver along narrow trails and paths, such as for example horse trails found in parks, wilderness area, and 3rd-world countries.

9.3.2. Quality Function Deployment (QFD) of the ST3LMR

Quality Function Deployment (QFD) was performed to identify the most important requirements necessary to support the operation of the robot and to focus this research. A research and development (R&D) scenario is assumed with an initial target market being the young adult age 20-35 with a technical education in software or engineering and a passion for riding motorcycles and ATVs. These are the same individuals who develop Linux, Android, and iPhone applications, the Willow Garage PR2 robot, and the DARPA Grand Challenge robots. They belong to the technology enthusiast and innovator market group, and they are readily reachable through university and technical communities.

The proposed marketing goal is to make the ST3LMR compelling enough as a whole product offering that the target market group will aggressively adopt, develop and refine key product capabilities. As the ST3LMR matures, their collaboration will drive purchases in the early adopters and visionary market group, who seek to define a new subculture and lifestyle around the ST3LMR. Ultimately, ST3LMR will mature into a reliable form of outdoor recreational transportation suitable for marketing to the more pragmatic majority market group, occupying a space in the existing 200 million unit motorcycle and ATV market, worldwide.

The single most compelling near-term application for the ST3LMR technology enthusiast is envisioned to be the first to ride a robot. Challenges such as track racing, land speed records, mobility, freestyle, and scrambling are exciting and would advance the state of the art. Most important to the “stickiness” of the product concept is to identify key individuals who share the product vision and can instill a “Tom Sawyer” effect among fellow innovators and early adopters (Heath and Heath 2007). As the product matures, the customer and end user will organically define the most popular applications. In other words, it is important to avoid overly ambitious expectations before the product and general markets are mature.

The ST3LMR QFD matrix is illustrated in Table 9.2. The left side lists the (weighted) internal and external customer wants and needs, the right side benchmarks the competition, the top defines the design features, the middle analyzes the requirement relationships, and the bottom sets the engineering targets and benchmarks. The center of the matrix quantifies the relative relationships (1 = weak to 9 = strong) between the engineering actions (vertical) against the wants and needs (horizontal). The right-hand benchmark matrix quantifies (1 = does not meet to 5 = fulfills the requirement completely) the comparable alternatives (vertical) against the wants and needs (horizontal). Values of 999 represent large numbers.

Through the QFD process, the important wants and needs for directed research were identified as: 1) being easy to ride, 2) easy to maintain/repair, 3) having agility over rough terrain, 4) not interfering with the spatial constraints of the rider, 5) having a high-quality simulation tool, 6) access to core algorithms, 7) a developer's community, and 8) a knowledgeable software application support team. The subjective want of being easy to ride is captured through the number of steps to operate, steps to dismount, maximum speed, maximum grade, maximum traverse, turning radius, overturn angle, and mean time between failure or MTBF. Core algorithms refer to the various algorithms required to control the robot (described in more detail in Chapters 4 and 5). For an R&D scenario, 3rd-party subject matter experts typically develop such "core" algorithms. Figure 9.3 depicts these direct and indirect customer wants and needs, as a total product offering.

Table 9.2. The ST3LMR Quality Function Deployment or QFD matrix.

Stramlemor QFD			Weighting (total 100)	Engineering Requirements														Bench-marks		
				Steps to operate	Steps to dismount	Maximum speed	Maximum grade	Maximum traverse	Turning radius	Overturn angle	Vehicle weight	# actuators	# of major parts	MTBF	MTTR	Robot OS	# library functions	Horse	Motorcycle	BigDog Robot
Functional performance		Low environmental impact	2			3	3	9	6		3							5	2	5
	Operation	Easy to mount/dismount	2	9	9					6	3							2	5	?
		Easy to ride	5	6	3	6	3	3	9	9				3				3	3	?
		Easy to fuel	2	1	1												1	2	5	5
		Easy to maintain/repair	5										9		9	3	6	1	5	2
		Easy to dispose	1											9	3			1	5	5
	Ride quality	Smooth ride	4				3	9	9	1						3	3	3	4	?
		Not wobble	4					6		6		9				3	6	4	4	4
		Agility over rough terrain	8							6	9	6				6	9	5	2	5
		Have long life	3				1	1	1	1		9	9	9	1	1	1	4	5	2
		Lightweight	5			3	1	1	1	1	9							2	5	4
		Not unpredictable behavior	2			1	1	1	1	3		9						3	5	?
Spatial constraints	Fit	Most riders	3	1	1						1	9						4	5	?
	Not interfere	With rider	4			1	1	9		3	1	6						4	4	?
		With passenger	2					9		3	1	6						4	4	?
		With cargo	2							3	1	6						4	4	4
		With accessories	2			3	3	3		3	1							3	4	3
		Narrow trails	2			3	3	3	6	9	1	9						4	4	3
		Doorways	1			1			6	1		9						4	4	2
		Trailers	1				6	6	6	1	3	9						4	4	4
3rd-Party Development		High-quality simulation	7	1	1	1	1	1	1	1		3				9	9	-	-	5
		Teach/replay capability	2													9	3	-	-	5
		Field programmable	4	3	3	1	1	1	1	1		1	1			9	1	-	-	4
		Access to core algorithms	10	1	1	1	1	1	1	1		1	1			3	9	-	-	-
		Training/manuals/FAQs	4													6	9	-	-	-
		Developer's community	6	1	1	1	1	1	1	1		1	1			9	9	-	3	-
		Software team knowledge	5	1	1	1	1	1	1	1		1	1			9	9	-	-	-
		24/7 software support	2	1												3	9	-	1	-
Promo		Competitions	-															4	4	-
		Prizes	-															3	3	-
Style		Streamlined	-															-	5	4
		Popular color	-															-	5	5
Time		9 month's development	-															-	5	1
		Marketable in 12 months	-															-	5	1
Cost		Minimum capital <$35M	-															-	-	1
		Manufacturing <$18k each	-															-	-	1
Mfg		~50/year for 5 years	-															-	-	3
		Use conventional facilities	-															-	-	4
		Units		#	#	mph	%	ft	ft	deg	lb	#	#	hr	hr	#	#			
		Targets		2	2	35	10	2.0	0	15	200	18	28	500	3	3	999			
		Horse		2	2	35	20	2.5	0	5	600	-	-	-	-	-	-			
		Motorcycle		2	3	120	35	0.2	20	10	350	1	8	999	3	-	-			
		BigDog Quadruped Robot		?	?	14.5	10	1.5	0	?	300	24	36	100	8	-	-			

Requirements: 1 = does not meet to 5 = fulfills the requirement completely
Relationships: 1 = weak relationship to 9 = strong relationship

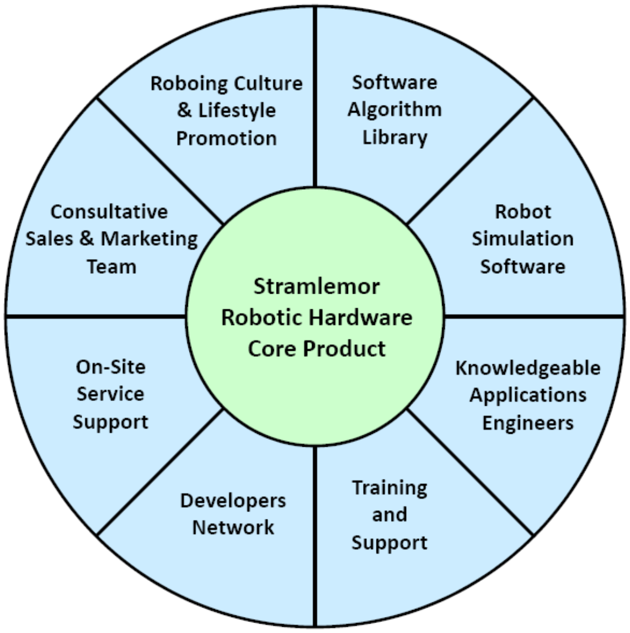

Roboing Culture & Lifestyle Promotion
Software Algorithm Library
Consultative Sales & Marketing Team
Robot Simulation Software
Stramlemor Robotic Hardware Core Product
On-Site Service Support
Knowledgeable Applications Engineers
Developers Network
Training and Support

Figure 9.3. The ST3LMR total product offering, drawn as a core hardware product surrounded by a circle of direct and indirect customer wants and needs.

Based on the current understanding of the end-user/customer, the external “why” is answered by analyzing the functional performance, spatial constraints, and 3rd-party development wants and needs. Specifically, being easy to ride, easy to maintain/repair, having agility over rough terrain, not interfering with the spatial constraints of the rider, having a high-quality simulation tool, access to core algorithms, a developer’s community, and a knowledgeable software application support team are most important to a successful R&D effort.

Benchmarking the competition (left and bottom) and analyzing the relationships (middle) reveal how the ST3LMR is innovative. Focusing on 3rd-party development, using a robotic operating system, providing a software library of functions, reducing the number of actuators requiring control, reducing the vehicle weight, and increasing the maximum overturn angle are key innovation areas. Engineering requirements and targets are specified to drive this research.

9.3.3. Value Proposition

Compared to a quadruped robot, the present invention uses 4 fewer actuators – reducing the number of legs from four to three, the minimum number of required actuators (three per BigDog leg) is reduced from 12 to 8, wherein the ST3LMR does not require a middle leg hip roll actuator. For comparison, the Boston Dynamics Company received US $10 million in US Defense Advanced Research Projects Agency (DARPA) funding to build an enhanced version of the Legged Squad Support System (LS3 a.k.a. the BigDog quadruped robot) over two years (Keller 2013). The current cost to build one BigDog robot, but not including software or sensor costs, is estimated at $45,000 and roughly $1,875 per actuator. By reducing the number of legs from four to three, the ST3LMR design would reduce the estimated hardware cost by 17% to $37,500 per unit. The ST3LMR would be positioned at about the same price as a high-end motorcycle or mid-size automobile, making the hardware design viable for the mass market.

Despite considerable interest in robotics, no company has yet produced a biped or quadruped robot that people ride (possibly due to the immaturity of legged robot technology and liability costs). The current product alternatives are the horse and the motorcycle. A comparative product offering (and one that may overcome liability cost issues) could be a ST3LMR that is directly controlled by the human rider. That is, the ST3LMR would use a motorcycle-like rider interface (e.g., handlebars for steering, and throttle and brake levers for speed control), and implement a follow-the-leader gait. The human-machine interface is discussed further in Chapter 10, Future Work.

Since Boston Dynamics received its initial DARPA research funding, the general state of technology has advanced considerably. Microprocessors are more powerful, batteries and fuel cells have greater capacity, and high-pressure precision hydraulic servo actuator technology is more reliable and affordable (thanks in-part to companies like Boston Dynamics). Software development, on the other hand, remains the long pole in the development tent. The open source development method is attractive because it can reduce the overall investment required to develop a robot. Further, the ST3LMR is exactly the type of product programmers in the target market group (see Chapter 1, section 4) are attracted to. To this end, a primary engineering strategy is to leverage the hardware to provide an open development platform, similar to existing robot kits available on the market today. Design collaboration and knowledge sharing would use existing open-source communities, tools, and resources. A complementary business strategy is to grow organizational knowledge in design for manufacture, product application support, and marketing and sales methods.

As a first-to-market leader, the ST3LMR would have a dramatic initial competitive advantage. The key to generating cash flow is to win over the early adopter and visionary market group (Moore 1991; and Christensen 1997). Depending on the success of the ST3LMR, direct competition from existing quadruped robots, such as from Boston Dynamic’s BigDog and AlphaDog quadruped robots, will naturally occur. In the long-run, a certain percentage of the market will become disenchanted and return to the motorcycle, an indirect form of competition. Thus an evolutionary product life cycle must be adopted to continuously improve the product and provide a compelling reason for the market to choose the ST3LMR. For example, providing incremental improvements to the user interface could provide a foundation for upgrades and new purchases, by appealing to technical and non-technical users alike.

CHAPTER 10

FUTURE WORK

The ultimate goal of this research is to build a Single-Track Three Legged Mobile Robot (ST3LMR) that people ride like a motorcycle or horse. To accomplish this vision, the research documented in this dissertation indicates the need for several improvements necessary to realize a full-size ST3LMR, including:

1) Development of an interactive, simulated ST3LMR with the goal of developing and testing a Human-Machine Interface (HMI) for legged locomotion;

2) Improved leg actuators capable of supporting and propelling a full-size robot;

3) A multi-segmented frame to provide additional flexibility in traversing irregular terrain and to provide a moving mass for stability of balance; and

4) A forward-looking sensor and control system to maintain stability of balance while executing a desired trajectory.

Improvements topics (1) and (2) are key to advancing the ST3LMR. Moreover, it is thought that the degree to which the rider provides balance control over the roll axis (like a motorcycle) versus the robot providing stability of balance (like a horse) is a fundamental engineering and business decision point. As an open research topic, much can be learned regarding human-robot interaction and the degree of partial- to fully-autonomous control used in conjunction with a rider. This chapter provides a discussion of proposed future work, which includes improvements to the leg actuators, a human-machine interface, a multi-segmented frame, a forward-looking sensing system, and a biologically inspired control system.

10.1. Human-Machine Interface

The joy of motorcycle riding occurs when balance, throttling, braking, and steering become instinctive, freeing the rider's mind for higher-level thinking. The method and degree to which the rider provides balance control of the ST3LMR roll axis (like a motorcycle) versus the robot providing stability of balance (like a horse) is a prime candidate for future research. Further, it is intriguing to take the rider as a starting point for robotic research. Figure 10.1 illustrates the concept of a Human-Machine Interface (HMI) interactive simulation. A simulation is preferred over a real ST3LMR because it not only reduces the risk of a human being hurt during experimentation but it provides a more controlled alpha-testing environment.

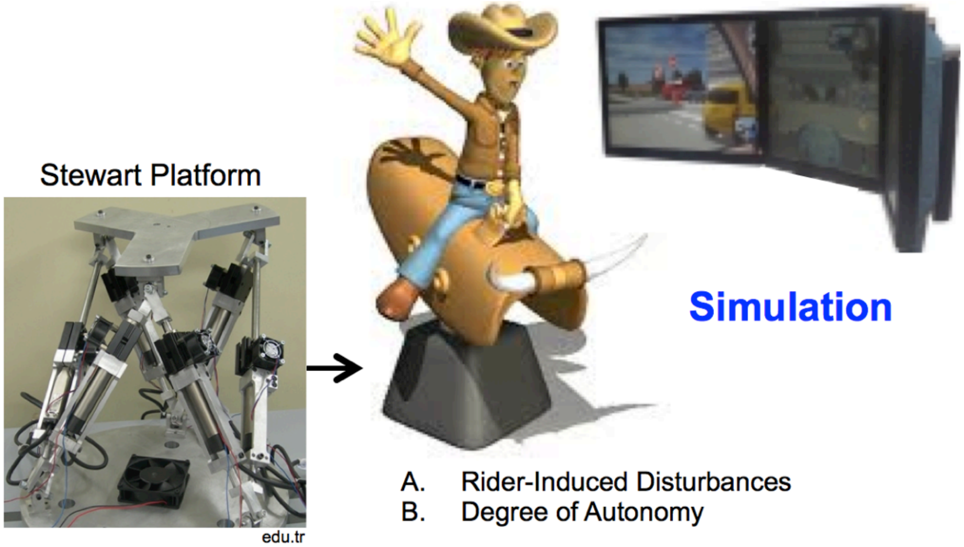


Figure 10.1. Human-Robot Interaction (HRI) interactive simulation concept.

The simulation would comprise the HMI interface and seat, similar to a motorcycle seat, mounted to a Stewart platform. The Stewart platform is a 6-DOF mechanical stage capable of moving the seat in three-dimensions to mimic actual ST3LMR riding. The Stewart platform is controlled by a computer running a simulated ST3LMR. The rider is presented with a display showing the road ahead. Through such a simulation, A) rider-induced disturbances and B) the degree of autonomy could be studied.

The simulated ST3LMR would have three general modes of operation for use with a rider: 1) fully autonomous mode (with or without a passenger, and especially useful for handicapped people); 2) partially autonomous mode with remotely controlled actions through an operator control unit; and 3) partially autonomous mode with a human operator riding on the robot. The objectives of this future research are described in the following three research questions:

1) Does model-based predictive control of a future state improve the stability and performance of the single-track legged robot for the lean-into-a-turn problem;

2) Can the results of a temporal state prediction benefit human-machine interaction; and

3) Can sensing a rider's body language anticipate changes in the road and thus reduce the complexity/cost of the robot control system?

Because the rider is an important aspect of the ST3LMR operation, stabilization of the upright equilibrium or balance in the presence of interaction between rider and the ST3LMR must be considered. For example, should the stability of balance be maintained after an external force is applied, regardless of rider commands? Maintaining stability of balance can be a complicated task. It involves counteracting pulling or pushing forces acting on the body by repositioning one or more legs while simultaneously adjusting robot body posture. If the disturbing force is severe, the ST3LMR can fall over due to the disturbance, regardless of its control response. When the ST3LMR changes leg positions or falls over, it can potentially hurt or kill a rider. Understanding safe methods for falling to avoid operator injury is important.

Nevertheless, it is anticipated that the operator would want to command the ST3LMR to perform basic operations, such as turn on or off, stand up, squat down, walk, trot, or jog. A remote or on-board user interface could provide the operator with operational and engineering data. The operator may provide high-level control input, such as pointing to a destination on a map, leaving the ST3LMR on-board control system to plan the route, operate the legs, provide stability on rough terrain and reflex responses to external disturbances.

As with riding a bicycle or motorcycle, it is also anticipated that the rider, as a control system, could contribute to the robot's stability of balance. Like a motorcycle, the rider may torque handlebars and operate a throttle and brake to provide high-level steering and speed input to guide the ST3LMR along its path and to control the speed of travel. If the ST3LMR executes a follow-the-leader leg gait, the operator may be capable of partial (i.e., assisted) or perhaps full control over stability of balance.

Further, a rider could also pull and twist on handlebars, change lateral posture, change on-axis posture, and provide momentary impulses of posture change (e.g., back-and-forth motion to initiate go, and backward leaning to initiate stop). When the rider leans backward/forward in preparation for a downhill/uphill slope, it would cause the legged mobile robot to lower the body height of the rear/front legs and thus optimize the projected center of pressure with respect to the downhill/uphill slope. On a slope or in advance of a slope, a rider may lean sideways to indicate uprightness, and the ST3LMR would respond by adjusting footfall placement to compensate for orientation of the body and virtual ground plane relative to the gravity vector. On level ground, sideways leaning by the rider would initiate ST3LMR body motion and foot placement for a turn.

To some extent the rider can be used to provide cues regarding full or partial balance. For example, the rider may command the ST3LMR through a physical interface, through body language of the rider as measured by the force and torque sensors and IMU, through an external control device, such as for example a radio control unit, voice commands, or visual commands or gestures, or through any combination of such devices and sensing. The rider may, for example, pull on one side of a steering bar and shift her center of gravity in advance or anticipation of a turn maneuver. Thus, cueing the trajectory planning system to begin the method of leaning into the turn and modifying the single track path or trajectory from a straight line to a curve. The rider also provides high-level control regarding direction of motion and velocity.

For example, when a person rides a motorcycle, the rider looks ahead for changes in the road, such as for example a curve or turn in the road, the rider plans the appropriate control strategy before the motorcycle reaches the turn, and the rider leans the motorcycle into the turn before the road begins to curve. It is this type of anticipative control strategy that a legged robot should perform if the legged robot is to operate and be ridden like a motorcycle. This research concerns the development of a two-time-scale, model-based predictive control system for a single-track or in-line legged robot performing a lean-into-a-turn maneuver.

Other methods for commanding the robot, such as through body language (e.g., shifting weight), verbal commands, and via a control interface (e.g., handle bars and hand-grip controls) could also provide high-level steering angle, throttle, and brake commands. Body language or behavioral cues are any actions or signals that indicate a rider's intent, such as for example body language, verbal command, and look or gaze direction. Such cues may provide meta-level information as to the actions riders desire the robot to do in the future. Such cues may be unconscious, such as for example when people unconsciously turn their head up to 25 degrees about 200 ms before turning (Imai 2001; and Chueh 2008). Cues may also be conscious actions with explicit instructions (Saito 2007). Cues may take the form of stereotyped behavior to signal a future event/desired action (Bien 2005). Cues may identify an emotional (affective) state, such as stress or fatigue, which could predict more general behavior patterns or habits to estimate future movement (Prendinger 2007).

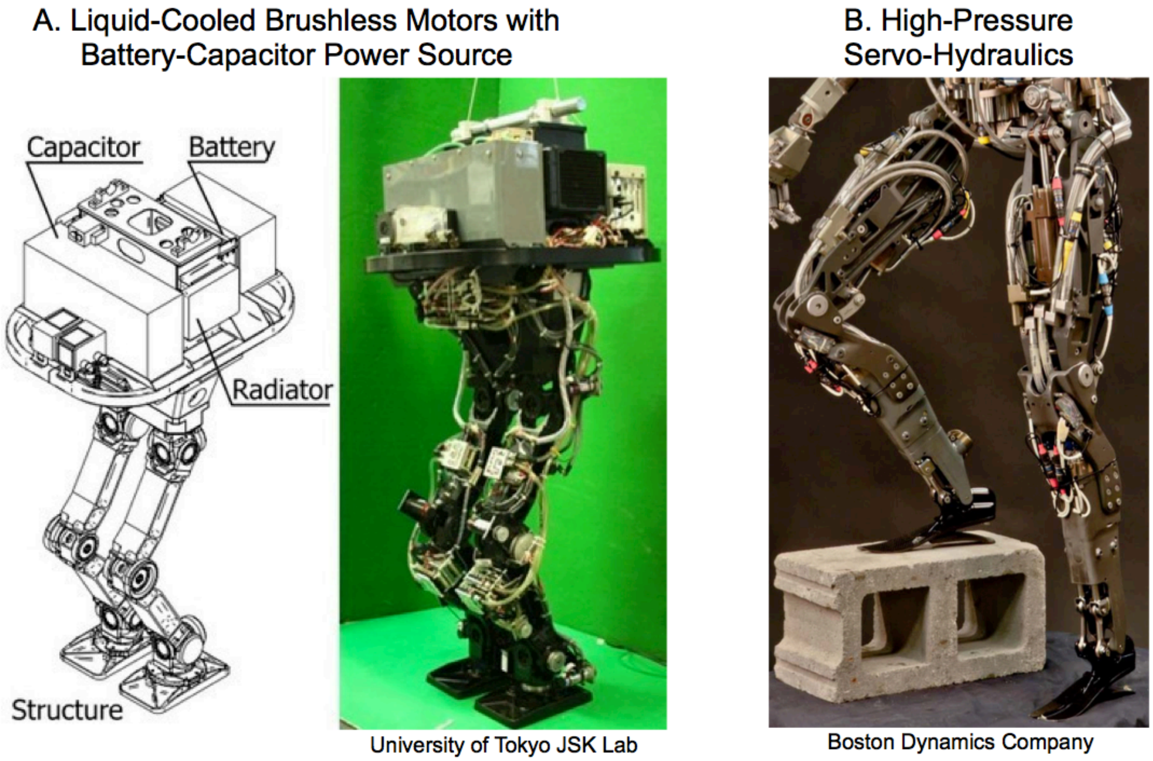
Note that the ST3LMR is an omnidirectional vehicle, but it is designed to have a major axis of motion in the forward/backward direction. That is, each leg has a greater range of motion in the forward/backward direction. Having legs enables the ST3LMR to emulate four different types of wheeled vehicle motion: both-wheels steering, parallel steering, combination steering, and rear-wheel steering. Developing a HMI for omnidirectional control of a legged vehicle is an open research topic, in itself.

For final consideration, it is important that the human rider interact with the legged robot in ways that do not require specific technical expertise. This is particularly true of dangerous or stressful environments in which robots may be used. A robot would be easily rejected if it were difficult to use. An example is using a walking robot in a rugged narrow trail environment where horses would otherwise perform this task well.

10.2. Improved Leg Actuators

A second and perhaps concurrent step to realizing a full-scale ST3LMR is improving the leg actuator. For example, leg joint angular velocities of 70-100 degrees per second are observed in people while walking and actively balancing (Osada 2011). Raibert's one-legged pneumatic hopping machine achieved a joint angular velocity of 114 degrees per second for a maximum forward velocity of 4.5 mph (Raibert 1986). The University of Tokyo JSK Lab is constructing a 1.2 meter tall jumping robot using a 13.5-Farad battery-capacitor controller and leg platform with liquid cooled motors at each joint, with an objective leg joint angular velocity of up to 300 degrees per second (Guizzo 2012). Boston Dynamics' Big Dog quadruped, Pet Man, and Cheetah robots use high-pressure hydraulics (up to 6,000 psi) combined with small-diameter hydraulic cylinders that are placed close to the leg joints. The Cheetah robot achieves a combined (knee, hip, and spine) joint angular velocity of over 430 degrees per second to sprint at up to 28.3 mph, which is faster than the fastest human sprinter, Mr. Usain Bolt at 27.79 mph (Boston Dynamics 2012).

All of the aforementioned actuators are custom designed. The choice of electric, pneumatic, hydraulic, or a combination thereof depends mainly on budget. For maximum capability, as demonstrated in the literature, hydraulic actuators are recommended. Seeking a development partnership with an industry leader in this field would be desired.


A. Liquid-Cooled Brushless Motors with Battery-Capacitor Power Source

B. High-Pressure Servo-Hydraulics

University of Tokyo JSK Lab

Boston Dynamics Company

Figure 10.2. Electric (left) and servo-hydraulic (right) actuated robot legs.

10.3. Multi-Segmented Frame

A third area for improvement is developing a multi-segmented frame to better traverse irregular terrain to improve stability of balance and to reduce energy consumption. The single-piece frame used in this research simplified the walking gaits and control thereof. However, the ST3LMR could comprise a frame that is jointed to include two or more segments, each segment having a major axis corresponding to and generally parallel to a forward/backward direction of travel, and be joined by a plurality of actuators, position-sensors and elastic components. The articulated frame would provide an advantage in flexibility, which extends the ranges of motion of each of the legs, particularly the front-most and rear-most legs. Such an arrangement would provide superior flexibility and adaptability to a wide range of terrains, and would enable the legged vehicle to traverse a wide range of terrains quickly.

A jointed frame would also enable the feet to have greater range of movement in length, width and height to provide more overlap of the working envelopes of at least two legs at any one time. It would improve the ability to place the legged vehicle into a stable bipedal or tripedal stance. A conformably flexible body would be better able to track the curvature of a single-track turn maneuver. Such an arrangement would further enable the minimally narrow profile relative to the major axis of motion, whereby the legged vehicle would better follow a narrow trail or pass (dynamically) through a doorway. With proper coordination of such dynamically repositionable masses (i.e., frame segments), a much faster, more natural gait could be developed to quickly traverse even the most challenging terrain.

If the jointed ST3LMR frame includes elastic energy storage and release components between segments, wherein the elastic components operate in at least one axis to store and release kinetic energy for transfer between adjacent frame segments and adjacent legs, the elastic members may be used to accept and release energy in a predictable manner. Such energy storage may be used by the control system to supplement the leg and/or body actuators in placing legs/feet at desired footholds more rapidly and accurately. It would also improve the capability (e.g., dissipating shock loads) of the ST3LMR when hopping, bounding, leaping, and jumping. Finally, elastic energy storage and release components would help reduce peak power requirements and reduce overall energy consumption through passive energy storage.

Furthermore, a jointed frame may afford a more aggressive gate, such as for example the leg crossover motion used by ice skaters to provide traction on lose ground or slippery surfaces. Independent rotation of frame segments would not only develop an outward torque in the direction of slip that would further counteract the outward centripetal inertial force acting on the center of gravity of the body but would incrementally push the body in the desired direction of motion. If the bottoms of each of the feet were ice skating blades, a piece-wise curve would develop force in the forward direction to propel the ST3LMR in the forward direction.

10.4. Forward-Looking Sensing System

Giving the ST3LMR the ability to sense its environment is fundamental to autonomously maintaining stability of balance while executing a desired trajectory. Table 10.1 lists five common sensor types and their sensing features. Key to such autonomous control in unstructured terrain is the detection, probabilistic scoring, ranking, and selection of potential footholds. Because vision is the predominant means by which people explore the world beyond their reach (Colby and Goldberg 1999), a computer image processing or machine vision sensor system is recommended.

Table 10.1. Sensing options summary for ST3LMR autonomy.

	Prop-reception	**IMU**	**Camera**	**LADAR**	**System**
Measurement	Force, Position, Velocity, Acceleration, Attitude	Velocity, Acceleration	Obstacles, Ground Slope, Terrain	Obstacles, Ground Slope, Terrain	Status, Load, Safety Margin
Type	Passive			Active	
Device	Potentiometer Load Cell Current	Linear Rate Angle Angle Rate	Monocular Stereo	Image + Range	Temp, Flow, Pressure
Method	Joint Angle Cosines	Gyro + Accelerometer	Optic Flow Sines	Time-of-Flight	Engine Actuators
Minuses	Slip	Drift	Motion Req’d	Snapshot	Lag
Observation-to-Observation Correlation – Temporal Tracking (as platform moves)					

Described in Chapter 6, a number of forward-looking steps are implemented to control and coordinate leg/foot motion, including: a) sensing terrain; b) associating the collected sensor data into sets of observations; c) predicting the future position, state and their covariance of tracked footholds based on past positions, states, and covariance; d) predicting multiple hypothesis of the future positions, states and their covariance of footholds using goal- or constraint-oriented and behavioral models; e) estimating beliefs about the state of the footholds; and f) comparing the predicted states with the sensory data to aid in the classification of footholds and path planning. Figure 10.3 illustrates the fundamental architecture of a computer image processing system for sensing terrain, i.e., item (a) above.

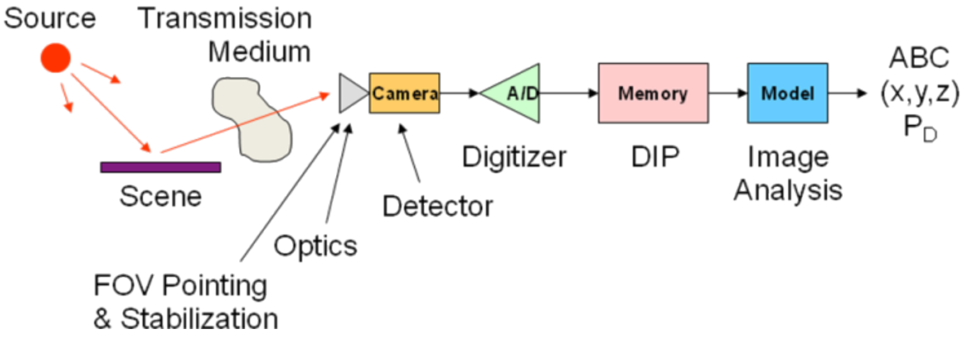


Figure 10.3. Fundamental computer image processing architecture.

Figure 10.3 illustrates a data-flow model of the basic functional components of the Single-Track Three Legged Mobile Robot (ST3LMR). From left to right, the environment is sensed using an image processing system, the sensed data is processed and flowed to a predictive model, which receives high-level control input and low-level kinematic data from on-board sensors, resulting in foot placement trajectories that are then filtered by a leg state machine and fed to real-time servo controllers, which then actuate the legs.

From left to right in Figure 10.3, light from a source illuminates a scene. Reflected or scattered light then passes through the transmission medium, such as air, to the camera. A typical camera detector comprises a raster array (millions) of cells, called pixels, that optically maps to a unique solid angle or field of view (FOV) of the scene. Each pixel integrates photons over a small period of time as an electrical charge. An analog to digital converter (A/D) samples the electrical charge and maps it to a numerical value of luminance. The value is stored in computer memory for each pixel. The pixel charge is then zeroed, and the integration process starts over.

Once in computer memory, digital image processing (DIP) and image analysis algorithms convert the numerical image to a meaningful representation. The four basic DIP operations are detection (presence or absence decision), location (position, rotation, and scale), measurement or gauging, and quality (statistical, geometric, texture). Image analysis typically involves an internal model or representation of component objects found in the scene. The output of computer image processing is typically object recognition, identification, or classification, with position and probability of data given the internal model.

The goal of image processing is to reduce the image size at every step of the process. Figure 10.4 conceptualizes image processing as an inverted pyramid or funnel. Millions of pixels are segmented into smaller regions of interest (ROIs), modeled, and finally represented by simple text, location, and probability.

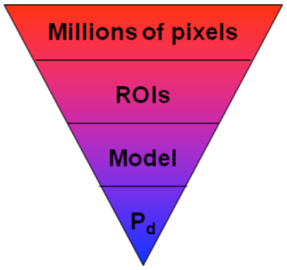


Figure 10.4. The computer image processing “funnel.”

There are two general types of DIP and image analysis algorithms. The first type operates on individual pixels and typically involves image enhancement, and includes such algorithms as image arithmetic (averaging, subtracting, weighting, rotation, scale), filtering and contrast (gamma, histogram equalization, thresholding), and statistical image quality. The second type operates on several pixels at a time, and includes algorithms such as morphology (erosion, dilation), frequency conversion (Fourier, wavelet), and spatial operators (low/band/high-pass, 1st and 2nd derivatives, edges, shape/structure, texture, and correlation).

Spatial operators are important to image processing because they provide a mechanism for applying a piece-wise algorithm over a much larger image. Equation (10.1) is the convolution algorithm of the spatial operator or template, T, with input image, f, yielding output image, g.

$$g(r,c) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} T(i,j) f(r+i, c+j) \tag{10.1}$$

Variables r and c are row and column image pixels coordinates, (-a, a) and (-b, b) are template row and column image pixel coordinates. Figure 10.5 illustrates equation 10.1 wherein a 3x3 spatial operator is applied over an image, f(r, c).

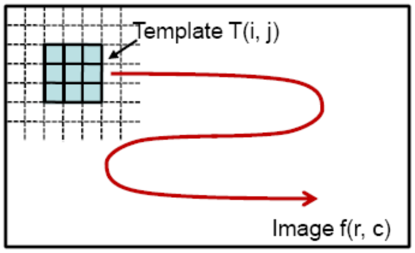


Figure 10.5. Illustrating a 3x3 spatial operator used in computer image processing.

For this research, and specifically for digital simulation, the recommended camera is a 2D flash laser radar (LADAR) sensor. The LADAR sensor measures the time of flight an impulse of light takes to travel from the robot to an object and back again. The time of flight is converted to a 2D distance map of the ground surface and objects thereon. The spatial operator is used to find contiguous ground surface regions suitable for foot placement.

In addition to sensing the terrain, the vision system may augment the inertial measurement unit or IMU. The IMU measures angle and angular velocity and acceleration, commonly using a microelectromechanical systems or MEMS sensor. Found in games, cell phones, cameras, model airplanes and helicopters, and other consumer and industrial products, the MEMS sensors are low-cost and reliable. A MEMS IMU typically provides very high sampling rates and fast response times, but its output contains high non-Gaussian noise and time-varying bias and/or scale factor (e.g., from temperature and vibration). A machine vision system may be used to measure absolute angle data from statistical image processing of horizontal and vertical features in unstructured environments. The machine vision angle data is combined with a MEMS gyro through a Kalman Filter, and the accuracy of the combined measurement exceeds that of either sensor used alone (Goulding CNS 2010; and Goulding MFI 2010).

Furthermore, a machine vision system may use other image processing algorithms (e.g., Superpixels segmentation, optical flow, contextual reasoning, and so on) to provide additional behavior-predicting information about the environment. Chapters 6 and 7 discuss the temporal core loop and temporal simulation loop used to predict future states.

Using the predicted motion of footholds, simulations are run to predict actions taken by the rider in response to robot actions (e.g., will the rider change his/her control input due to the robot action to maintain stability?) that also aids in the classification of footholds. The simulation loop is not tied to data collection and therefore can execute at higher rates, i.e., a two-time-scale system. Thus, multiple robot-centric hypothesis planning may be performed by the simulation loop between sensor collection and processing by the machine vision system.

In this recommendation for future work, a behavior-based motion model may be used in place of the Kalman filter. A behavior-based motion model can better predict the path of chaotic and erratic behavior, given for example, texture based information about a road surface. Behavior-based systems (e.g., subsumption architectures used to control unmanned ground vehicles) go beyond conventional motion models to enable emergent behavior. A hybrid implementation, such as an independent multiple model (IMM) or multiple hypothesis tracking (MHT) system, would enable the behavior algorithm to "select" the motion model to use (straight, curved, stop-go, etc.). Behavior, unlike a velocity vector and its covariance, takes additional factors into account. For example, a motorcycle or bicycle cannot move sideways and, therefore, a piecewise-linear algorithm may be used to model its behavior. The ST3LMR, on the other hand, can move sideways and a probabilistic model would incorporate some probability distribution to this end.

Moreover, behavior refers to other controllable aspects, features, or sub-components of robots, such as kinetic and potential energy, leg joint angles, system center of gravity/pressure, gravitational and inertial forces, temperature (e.g., hot/cold), color, mode of operation (e.g., walking, running, trotting, pacing, hopping, etc.), pose (e.g., orientation/heading, one body segment with respect to another, e.g., camera pointing), posture (e.g., relaxed, recoiled, walking stance, reaching arm, etc.), and so on. Behavior models also apply to secondary systems and effects and local environmental conditions and effects, such as wind and noise (that may affect rider control/response).

Behavior-based motion models would enable complex, emergent patterns. Behavior based models would also enable probabilistic prediction of goal-oriented motion and aid in the classification of footholds. Given a priori goal map data, a behavior-based hypothesis would compute a goal-based trajectory for analysis. Behavior/Goal oriented motion can be used to determine a likely path that the robot will take. Thus, there is less uncertainty of the robot moving sideways, for example, and this has a positive benefit on deleting paths with low scores, following paths with higher scores, and in correlating the simulated robot-to-rider and robot-to-robot tracks with sensed data to aid in the classification of footholds.

10.5. Biologically-Inspired Control Systems

Static rule based systems could provide Bayesian frameworks for reasoning about such features, but robust learning from new data would remain problematic. Figure 10.6 illustrates a visual-behavior-based controller concept for controlling the ST3LMR, presented by this author in October 2009 (as a potential Dissertation topic). Called the “Elegant Stepping Simulation,” it was adapted from the RatSLAM hippocampal model (Milford 2008). It uses experiential learning of road preview images and proprioceptive feedback (e.g., body height, roll angle and rate, and forward velocity) to plan (map) forward velocity, body attitude, and especially for running, leaping, and bounding.

As illustrated in the upper left of Figure 10.6, Local View Cells (LV), i.e., the machine vision system, process the color camera data to provide allothetic cues, used to estimate body attitude. The Pose Cells (PC), a competitive attractor network (i.e., one winner / one state) receive proprioceptive feedback. The Experience Map (EM) contains experiential information about hopping height, body attitude, and forward velocity given the LV and PC states. The RatSLAM algorithm learns associations between its internal state and sensory data. The EM generates action hypothesis for the ST3LMR control, interpreting behavior-predicting information about the environment to adjust the translation of the legs (with hip and/or ankle rotation) one or more steps in advance of the current state. It is hypothesized that such a biologically inspired, experientially taught system would better predict nonlinear behavior and better manage leg motion in out-of-control situations to regain stability of balance.

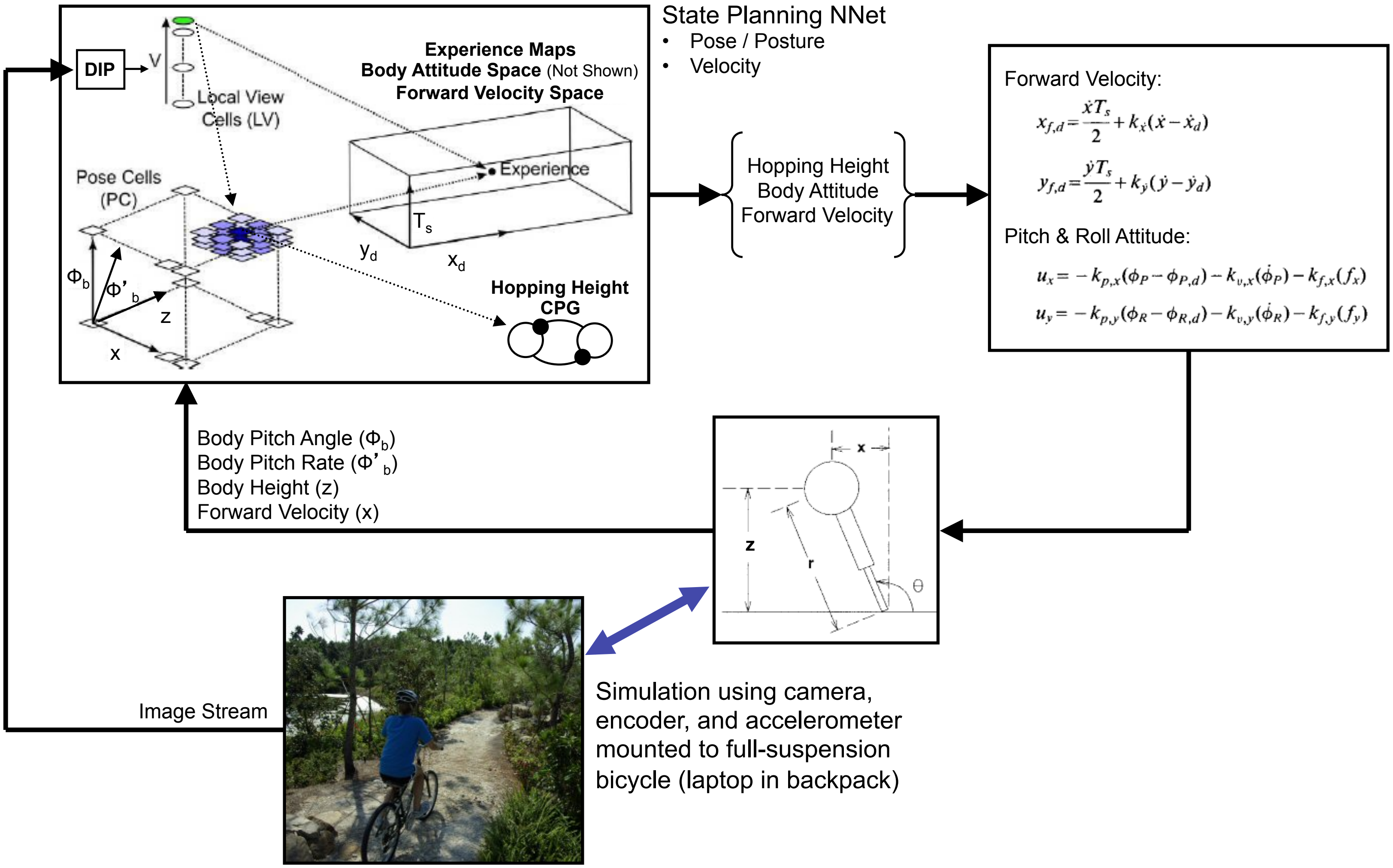

DIP
V
Local View Cells (LV)
Experience Maps
Body Attitude Space (Not Shown)
Forward Velocity Space
Experience
Pose Cells (PC)
T_s
y_d
x_d
Φ_b
Φ'_b
z
x
Hopping Height CPG
State Planning NNet
• Pose / Posture
• Velocity
Hopping Height
Body Attitude
Forward Velocity
Forward Velocity:
$x_{f,d} = \frac{\dot{x}T_s}{2} + k_{\dot{x}}(\dot{x} - \dot{x}_d)$
$y_{f,d} = \frac{\dot{y}T_s}{2} + k_{\dot{y}}(\dot{y} - \dot{y}_d)$
Pitch & Roll Attitude:
$u_x = -k_{p,x}(\phi_P - \phi_{P,d}) - k_{v,x}(\dot{\phi}_P) - k_{f,x}(f_x)$
$u_y = -k_{p,y}(\phi_R - \phi_{R,d}) - k_{v,y}(\dot{\phi}_R) - k_{f,y}(f_y)$
Body Pitch Angle (Φ_b)
Body Pitch Rate (Φ'_b)
Body Height (z)
Forward Velocity (x)
x
z
r
θ
Image Stream
Simulation using camera, encoder, and accelerometer mounted to full-suspension bicycle (laptop in backpack)

Figure 10.6. Elegant Stepping Simulation using a hippocampus-inspired, experiential learning, behavior-based motion model to control the ST3LMR, especially for running, leaping, and bounding.

In summary, 1) developing a ST3LMR/HMI simulator with the goal of testing rider-based legged locomotion, 2) improving the leg actuators, 3) constructing a multi-segmented frame to provide additional flexibility in traversing irregular terrain and to provide a moving mass for stability of balance, and 4) adding a forward-looking sensor and control system to maintain stability of balance while executing a desired trajectory are all possible research topics for future work. The ST3LMR/HMI simulator is recommended as the best short-term approach to realizing a full-scale ST3LMR that people would ride.

APPENDIX A

ST3LMR DIGITAL SIMULATION CODE

A simple digital simulation was written in MATLAB as a proof-of-concept for the Single-Track Three Legged Mobile Robot (ST3LMR). In this simulation, the ST3LMR is modeled using the coupled-drive or Hirose legs (discussed in Chapter 2, Section 2.2.2), with real-world parameter values from experimental results (presented in Appendices B and C). A 2-dimension (2D) inverted pendulum model is used in conjunction with a Backwards Wave Gait (BWG), illustrated in Figure A.1 (and discussed in Chapter 4, Section 4.2). The ground plane is the x- and y-axis, where the x-axis is in the direction of travel and the y-axis is perpendicular. The z-axis is the height axis. The pendulum base is defined by two supporting feet of the BWG, which allows the ST3LMR to free-fall in the 2D x- and y-axis. Positive roll follows the right-hand rule. The transition between a supporting foot and a flight foot is modeled as an instantaneous process.

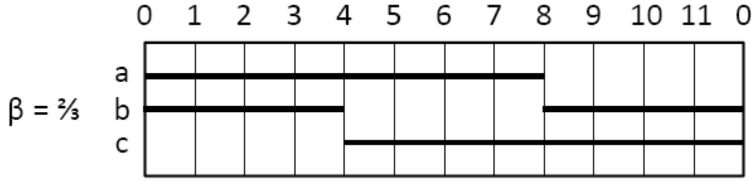


Figure A.1. Example of an ideal backward wave gait with 2/3 stance and 1/3 flight phasing.

Figure A.1 is an example of a Gait Diagram illustrating a 2/3 support phase (illustrated as a black horizontal line, also called beta) and 1/3 swing phase (illustrated as no line) BWG along the horizontal axis for each of the three legs (a, b, and c) shown on the vertical axis. It is called a BWG because the flight phase of the legs progresses from the rear leg c to the middle leg b to the front leg a. In this example, the three flight phases are evenly dispersed over the period. For clarity of understanding, Figure A.2 illustrates how the legs and feet of a robot move when executing the BWG illustrated in Figure A.1. The ST3LMR digital simulation comprises the following seven functions:

1) ST3LMR_MPC2 – This function is the main calling function responsible for sequencing the simulation, and it also increments the robot in time;

2) initParams – This function initializes the simulation and robot parameters, using real experimental results taken from experimentation with the coupled-drive leg in Appendix B and the Draisine robot in Appendix C;

3) invPendulumModel – This function implements a simple inverted pendulum, to update angular acceleration, velocity, and angle;

4) projectedCG – This function computes the projected Center of Gravity (CG) using the Law of cosines to determine the pendulum length and angle (theta) for each time slice;

5) hipLocations – This function computes the hip locations based on the roll, pitch and yaw of the CG and robot physical parameters;

6) mpcSim – This function perform multi-hypothesis planning using branch-based scoring in a depth-limited search; and

7) stateMachine – This function implements the BWG by sequencing and timing the leg stance and flight phases.

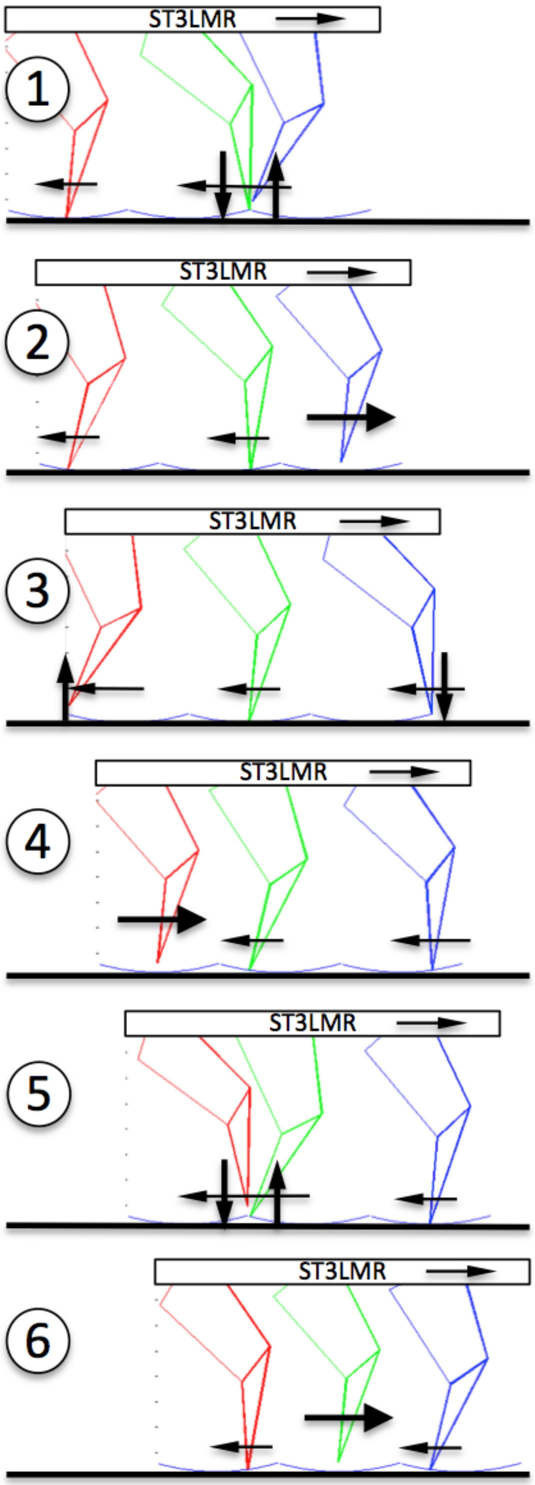


Figure A.2. Example of the ST3LMR legs executing a backward wave gait with 2/3 stance and 1/3 flight phase.

Figure A.3 illustrates the functions and calling sequence for the ST3LMR digital simulation. The function mpcSim recursively calls itself to perform multi-hypothesis planning using branch-based scoring in a depth-limited search. It implements the model-predictive control system illustrated in the block diagram shown in Figure A.4. See Chapter 5, Section 5.2.2 for a discussion of the high-speed model.

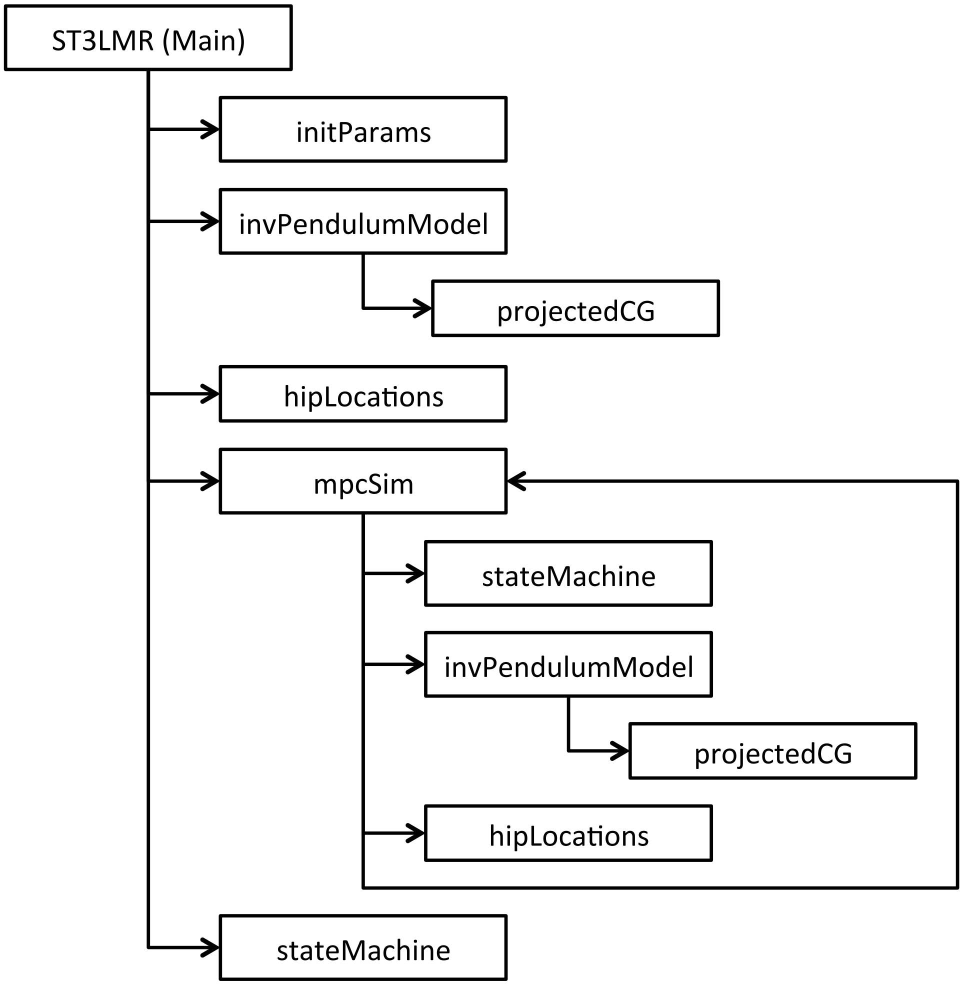


Figure A.3. Function call diagram for the ST3LMR digital simulation.

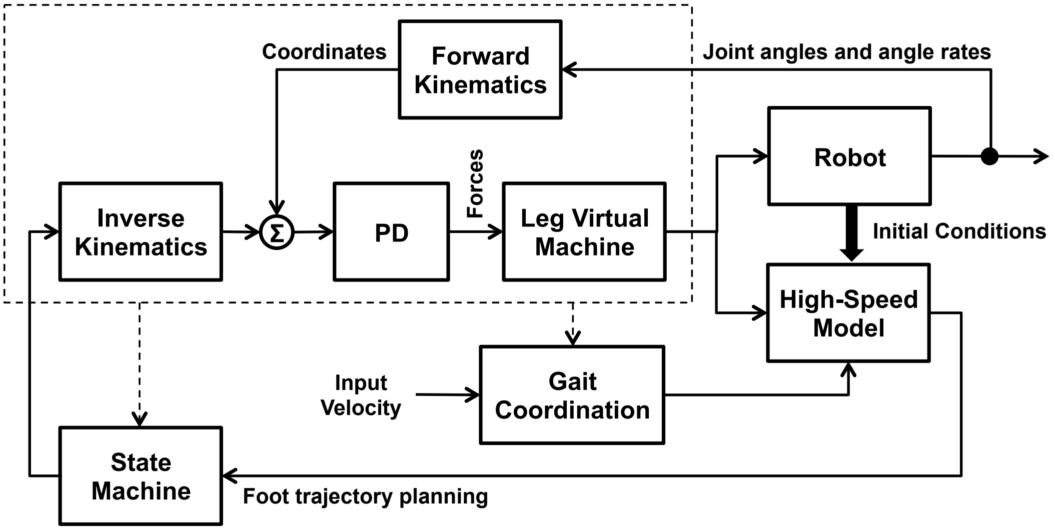
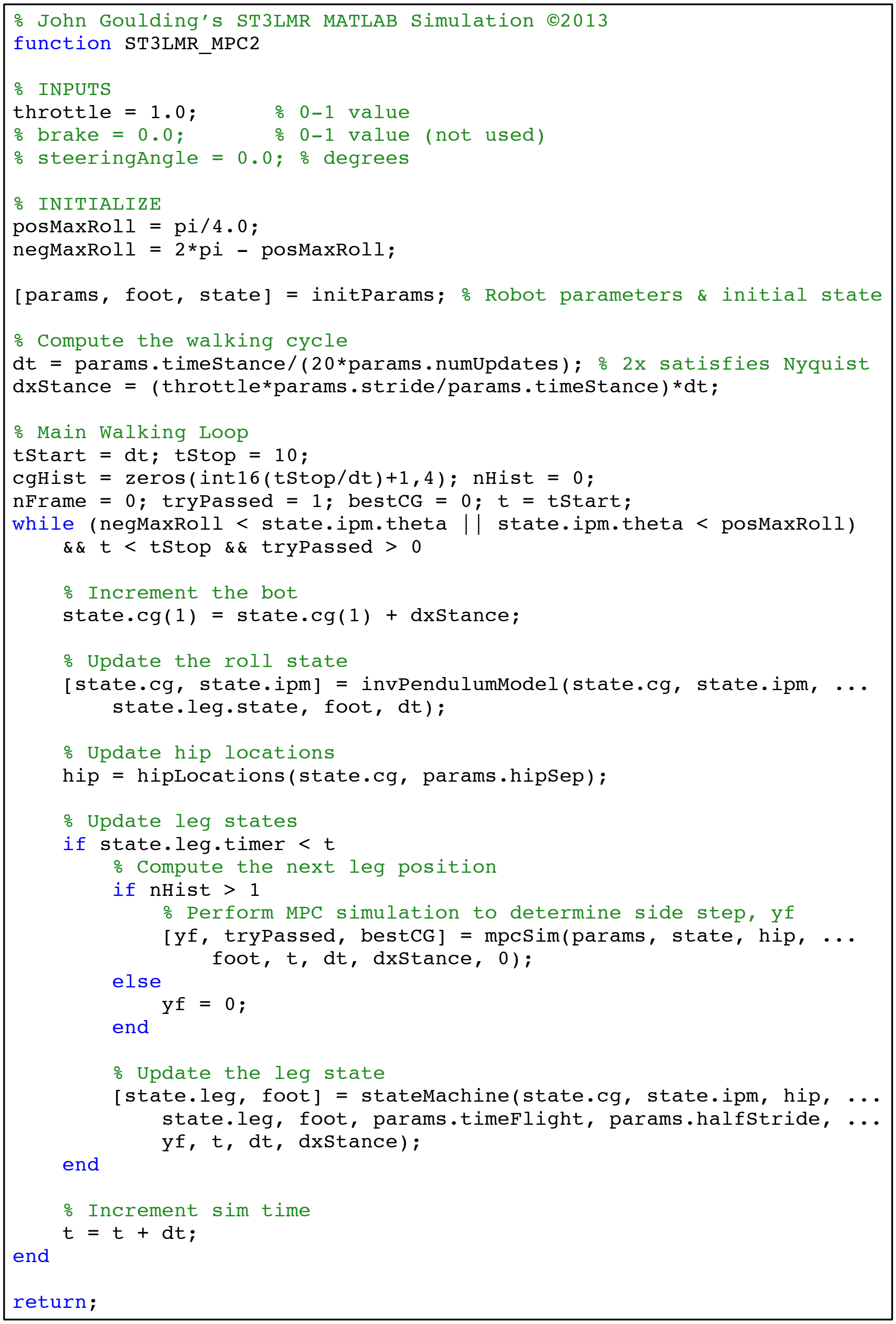


Figure A.4. Block diagram of the model-predictive control (MPC) system developed for the ST3LMR.

Figures A.5 through A.17 present the MATLAB code used in this digital simulation. In as much as possible, real-world parameters are used to accurately model the proposed ST3LMR. For example, the initParams function (Figures A.6 and A.7) derives the Stance and Flight phase time and distance from the experimental results presented in Appendix B, Table B.1. Other parameters, such as hip-to-hip spacing are estimated based on the Draisine prototype, described in Chapter 8, Section 8.2.

To produce useful work, the STLMR must follow a desired path and execute turns when and where directed, all without falling over. The code presented here is merely a starting point to illustrate the Monte Carlo-based inverted pendulum temporal estimation and control method. It is left as an exercise for the reader to modify the code to implement, for example, a more computationally efficient follow-the-leader gait, trajectory-based control methods, and sensory input for obstacle avoidance.

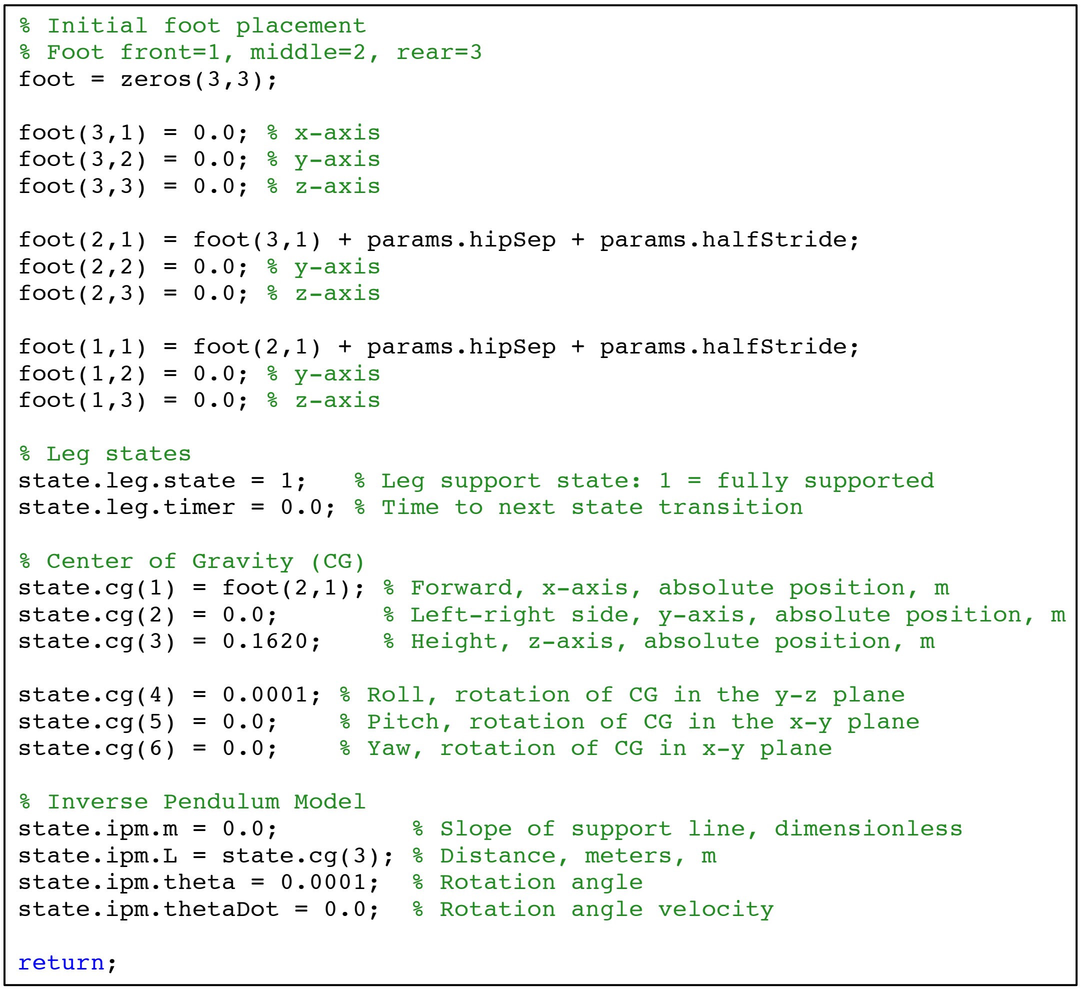


```
% John Goulding’s ST3LMR MATLAB Simulation ©2013
function ST3LMR_MPC2

% INPUTS
throttle = 1.0;        % 0-1 value
% brake = 0.0;         % 0-1 value (not used)
% steeringAngle = 0.0; % degrees

% INITIALIZE
posMaxRoll = pi/4.0;
negMaxRoll = 2*pi - posMaxRoll;

[params, foot, state] = initParams; % Robot parameters & initial state

% Compute the walking cycle
dt = params.timeStance/(20*params.numUpdates); % 2x satisfies Nyquist
dxStance = (throttle*params.stride/params.timeStance)*dt;

% Main Walking Loop
tStart = dt; tStop = 10;
cgHist = zeros(int16(tStop/dt)+1,4); nHist = 0;
nFrame = 0; tryPassed = 1; bestCG = 0; t = tStart;
while (negMaxRoll < state.ipm.theta || state.ipm.theta < posMaxRoll)
    && t < tStop && tryPassed > 0

    % Increment the bot
    state.cg(1) = state.cg(1) + dxStance;

    % Update the roll state
    [state.cg, state.ipm] = invPendulumModel(state.cg, state.ipm, ...
        state.leg.state, foot, dt);

    % Update hip locations
    hip = hipLocations(state.cg, params.hipSep);

    % Update leg states
    if state.leg.timer < t
        % Compute the next leg position
        if nHist > 1
            % Perform MPC simulation to determine side step, yf
            [yf, tryPassed, bestCG] = mpcSim(params, state, hip, ...
                foot, t, dt, dxStance, 0);
        else
            yf = 0;
        end

        % Update the leg state
        [state.leg, foot] = stateMachine(state.cg, state.ipm, hip, ...
            state.leg, foot, params.timeFlight, params.halfStride, ...
            yf, t, dt, dxStance);
    end

    % Increment sim time
    t = t + dt;
end

return;
```

Figure A.5. MATLAB code for the Main function of the ST3LMR simulation.

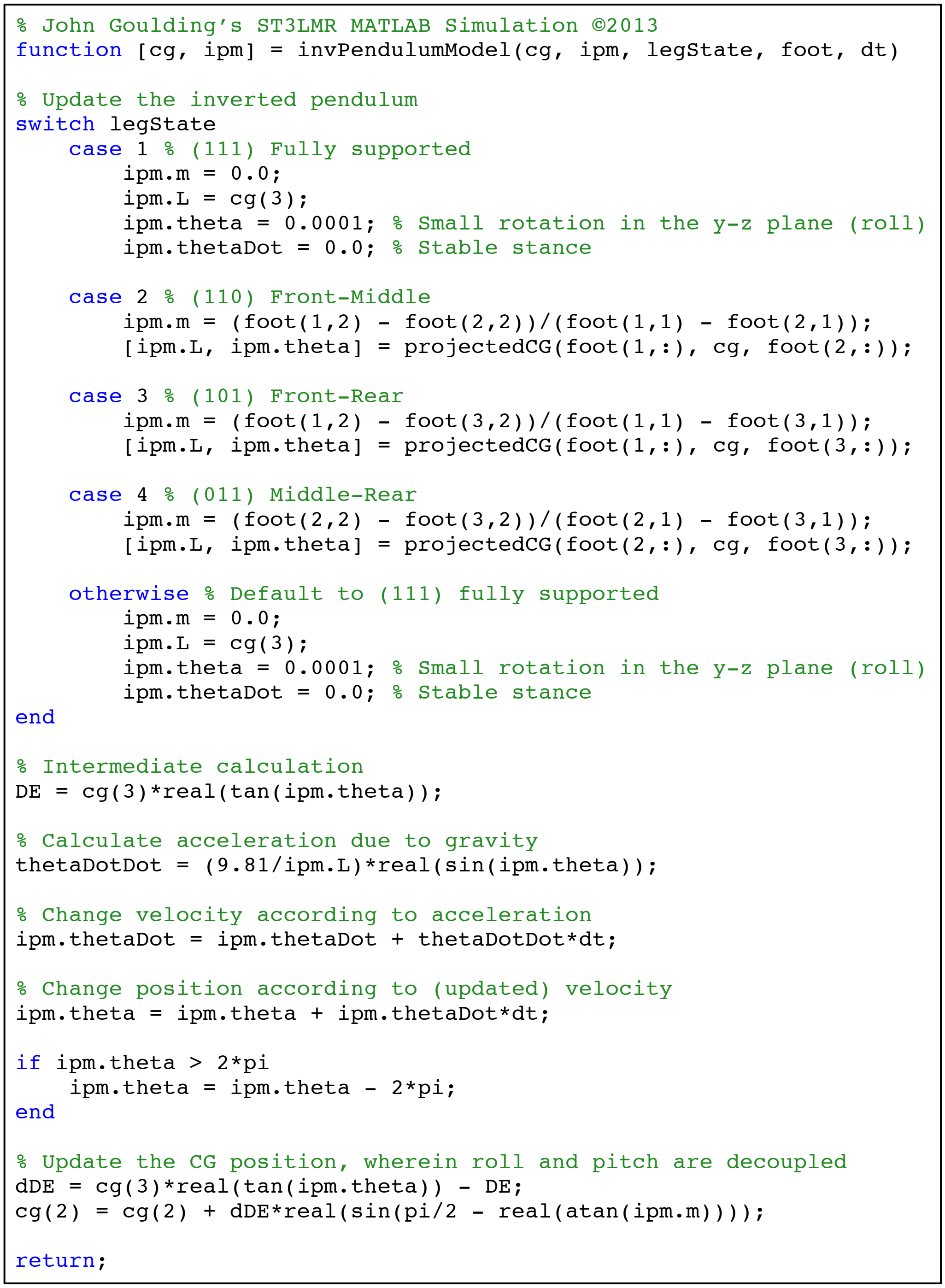
```
% John Goulding's ST3LMR MATLAB Simulation ©2013
function [params, foot, state] = initParams

% Open-Loop foot position data for the Hirose Leg (Table B.1)
RT = [18.75, 18.79, 18.82, 18.86, 18.89, 18.92, 18.96, 18.99, 19.03, ...
      19.06, 19.10, 19.13, 19.16, 19.20, 19.23, 19.27, 19.30, 19.33, ...
      19.37, 19.40, 19.44, 19.47, 19.50, 19.54, 19.57, 19.61, 19.64, ...
      19.67, 19.71, 19.74, 19.78, 19.81, 19.85, 19.88, 19.91, 19.95, ...
      19.98, 20.02, 20.05, 20.08, 20.12, 20.15, 20.19, 20.22, 20.25, ...
      20.29, 20.32, 20.36, 20.39, 20.42, 20.46, 20.49, 20.53, 20.56, ...
      20.60, 20.63, 20.66, 20.70, 20.73, 20.77, 20.80, 20.83, 20.87, ...
      20.90, 20.94, 20.97, 21.00, 21.04, 21.07];

X = [ 18.0,  19.0,  23.0,  29.0,  37.0,  45.0,  54.0,  62.5,  70.0, ...
      75.0,  81.0,  87.5,  94.5, 100.5, 105.0, 106.0, 106.5, 106.0, ...
     105.0, 103.0, 101.5, 100.0,  98.0,  96.5,  94.5,  93.0,  91.5, ...
      90.0,  89.0,  87.0,  85.5,  84.0,  82.0,  81.0,  79.5,  78.0, ...
      76.0,  74.5,  73.0,  72.0,  70.0,  69.0,  67.0,  65.0,  62.5, ...
      61.0,  58.5,  57.0,  55.5,  53.5,  51.5,  49.5,  47.5,  46.0, ...
      44.0,  41.5,  40.0,  38.5,  36.0,  34.5,  32.5,  30.5,  29.0, ...
      26.0,  25.0,  23.5,  21.5,  19.5,  18.0];

Z = [  0.0,  -4.5,  -7.0, -10.0, -12.0, -13.0, -15.0, -18.0, -18.0, ...
     -17.0, -12.0,  -9.5,  -6.0,  -2.0,   0.5,   1.5,   2.0,   2.0, ...
       2.0,   2.0,   1.5,   1.0,   1.0,   0.5,   0.5,   0.0,   0.0, ...
       0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0, ...
       0.0,   0.0,   0.0,   0.0,   0.5,   1.0,   1.0,   1.0,   0.5, ...
       0.5,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0, ...
       0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0,   0.0, ...
       0.0,   0.0,   0.0,   0.0,   0.0,   0.0];

% Derived parameters
stateStance = find(Z >= 0.0);
stateFlight = find(Z < 0.0);

% Duration of Stance phase (Ts), seconds, s
params.timeStance = RT(stateStance(1,length(stateStance(1,:)))) - ...
    RT(stateStance(1,2));
dxStance = (X(stateStance(1,2)) - ....
    X(stateStance(1,length(stateStance(1,:)))))/1000.0; % in meters (m)

% Duration of Flight phase (Ts), seconds, s
params.timeFlight = RT(stateStance(1,2)) - RT(stateFlight(1,1));
dxFlight = (X(stateStance(1,2)) - X(stateFlight(1,1)))/1000.0; % in m

% x-axis foot stride length, m
params.stride = (dxFlight + dxStance)/2.0;
params.halfStride = params.stride/2.0;
params.sideStep = 0.05;

% Hip-to-hip distance or separation
params.hipSep = params.stride;

% Number of updates during Stance phase
params.numUpdates = 10;
```

Figure A.6. MATLAB code for the Parameter Initialization function (part 1 of 2) of the ST3LMR simulation.



```
% Initial foot placement
% Foot front=1, middle=2, rear=3
foot = zeros(3,3);

foot(3,1) = 0.0; % x-axis
foot(3,2) = 0.0; % y-axis
foot(3,3) = 0.0; % z-axis

foot(2,1) = foot(3,1) + params.hipSep + params.halfStride;
foot(2,2) = 0.0; % y-axis
foot(2,3) = 0.0; % z-axis

foot(1,1) = foot(2,1) + params.hipSep + params.halfStride;
foot(1,2) = 0.0; % y-axis
foot(1,3) = 0.0; % z-axis

% Leg states
state.leg.state = 1;   % Leg support state: 1 = fully supported
state.leg.timer = 0.0; % Time to next state transition

% Center of Gravity (CG)
state.cg(1) = foot(2,1); % Forward, x-axis, absolute position, m
state.cg(2) = 0.0;       % Left-right side, y-axis, absolute position, m
state.cg(3) = 0.1620;    % Height, z-axis, absolute position, m

state.cg(4) = 0.0001; % Roll, rotation of CG in the y-z plane
state.cg(5) = 0.0;    % Pitch, rotation of CG in the x-y plane
state.cg(6) = 0.0;    % Yaw, rotation of CG in x-y plane

% Inverse Pendulum Model
state.ipm.m = 0.0;          % Slope of support line, dimensionless
state.ipm.L = state.cg(3); % Distance, meters, m
state.ipm.theta = 0.0001;  % Rotation angle
state.ipm.thetaDot = 0.0;  % Rotation angle velocity

return;
```

Figure A.7. MATLAB code for the Parameters Initialization function (part 2 of 2) of the ST3LMR simulation.



```
% John Goulding's ST3LMR MATLAB Simulation ©2013
function [cg, ipm] = invPendulumModel(cg, ipm, legState, foot, dt)

% Update the inverted pendulum
switch legState
    case 1 % (111) Fully supported
        ipm.m = 0.0;
        ipm.L = cg(3);
        ipm.theta = 0.0001; % Small rotation in the y-z plane (roll)
        ipm.thetaDot = 0.0; % Stable stance

    case 2 % (110) Front-Middle
        ipm.m = (foot(1,2) - foot(2,2))/(foot(1,1) - foot(2,1));
        [ipm.L, ipm.theta] = projectedCG(foot(1,:), cg, foot(2,:));

    case 3 % (101) Front-Rear
        ipm.m = (foot(1,2) - foot(3,2))/(foot(1,1) - foot(3,1));
        [ipm.L, ipm.theta] = projectedCG(foot(1,:), cg, foot(3,:));

    case 4 % (011) Middle-Rear
        ipm.m = (foot(2,2) - foot(3,2))/(foot(2,1) - foot(3,1));
        [ipm.L, ipm.theta] = projectedCG(foot(2,:), cg, foot(3,:));

    otherwise % Default to (111) fully supported
        ipm.m = 0.0;
        ipm.L = cg(3);
        ipm.theta = 0.0001; % Small rotation in the y-z plane (roll)
        ipm.thetaDot = 0.0; % Stable stance
end

% Intermediate calculation
DE = cg(3)*real(tan(ipm.theta));

% Calculate acceleration due to gravity
thetaDotDot = (9.81/ipm.L)*real(sin(ipm.theta));

% Change velocity according to acceleration
ipm.thetaDot = ipm.thetaDot + thetaDotDot*dt;

% Change position according to (updated) velocity
ipm.theta = ipm.theta + ipm.thetaDot*dt;

if ipm.theta > 2*pi
    ipm.theta = ipm.theta - 2*pi;
end

% Update the CG position, wherein roll and pitch are decoupled
dDE = cg(3)*real(tan(ipm.theta)) - DE;
cg(2) = cg(2) + dDE*real(sin(pi/2 - real(atan(ipm.m))));

return;
```

Figure A.8. MATLAB code for the Inverted Pendulum Model function of the ST3LMR simulation.

```
% John Goulding’s ST3LMR MATLAB Simulation ©2013
function [DB, theta] = projectedCG(ptA, ptB, ptC)

% By the Law of cosines
AB = sqrt((ptA(1) - ptB(1))^2 + (ptA(2) - ptB(2))^2 + (ptA(3) - ...
    ptB(3))^2);
AC = sqrt((ptA(1) - ptC(1))^2 + (ptA(2) - ptC(2))^2 + (ptA(3) - ...
    ptC(3))^2);
BC = sqrt((ptB(1) - ptC(1))^2 + (ptB(2) - ptC(2))^2 + (ptB(3) - ...
    ptC(3))^2);

alpha = real(acos((AC^2 + AB^2 - BC^2)/(2*AC*AB)));

DB = AB*sin(alpha);

theta = pi/2 - real(asin(ptB(3)/DB)) + 0.0001;

if (ptA(2) - ptB(2))/(ptA(1) - ptB(1)) > ...
    (ptA(2) - ptC(2))/(ptA(1) - ptC(1))
    theta = 2*pi - theta;
end

return;
```

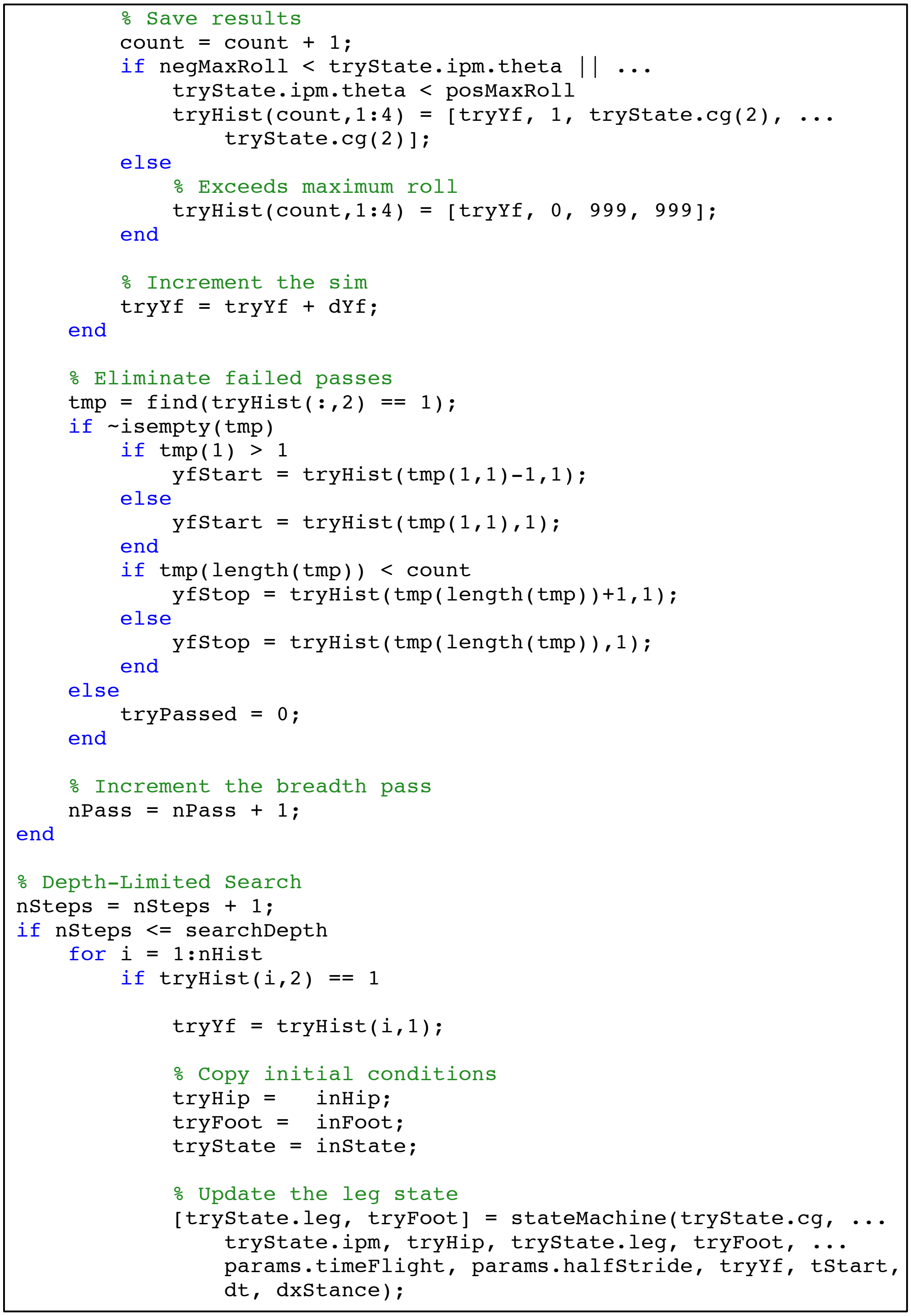
Figure A.9. MATLAB code for the Projected Center of Gravity (projectedCG) function of the ST3LMR simulation.

```
% John Goulding’s ST3LMR MATLAB Simulation ©2013
function [hip] = hipLocations(cg, hipSep)

hip(1,1) = cg(1) + hipSep*cos(cg(6));
hip(1,2) = cg(2) + hipSep*sin(cg(6));
hip(1,3) = cg(3); % Note roll and pitch are decoupled

hip(2,1) = cg(1);
hip(2,2) = cg(2);
hip(2,3) = cg(3);

hip(3,1) = cg(1) - hipSep*cos(cg(6));
hip(3,2) = cg(2) - hipSep*sin(cg(6));
hip(3,3) = cg(3); % Note roll and pitch are decoupled

return;
```

Figure A.10. MATLAB code to compute the Hip Locations (hipLocations) function of the ST3LMR simulation.

```
% John Goulding’s ST3LMR MATLAB Simulation ©2013
function [bestYf, tryPassed, bestCG] = mpcSim(params, inState, ...
    inHip, inFoot, tStart, dt, dxStance, nSteps)

posMaxRoll = pi/4.0;
negMaxRoll = 2*pi - posMaxRoll;

% Multi-hypothesis planning using branch-based scoring
yfStart = inState.cg(2) - params.sideStep;
yfStop = inState.cg(2) + params.sideStep;
tryPassed = 1;
nPass = 1;
while nPass < 3 && tryPassed > 0

    % Try all possible values of yf
    if nPass == 1
        nHist = 200;
    else
        nHist = 10;
    end
    tryHist = zeros(nHist,4); % Stores results
    tryYf = yfStart;
    dYf = (yfStop - yfStart)/nHist;
    count = 0;
    while tryYf < yfStop && count < nHist

        % Copy initial conditions
        tryHip =   inHip;
        tryFoot =  inFoot;
        tryState = inState;

        % Update the leg state
        [tryState.leg, tryFoot] = stateMachine(tryState.cg, ...
            tryState.ipm, tryHip, tryState.leg, tryFoot, ...
            params.timeFlight, params.halfStride, tryYf, tStart, ...
            dt, dxStance);

        % Simulation loop
        t = tStart;
        while (negMaxRoll < tryState.ipm.theta || ...
            tryState.ipm.theta < posMaxRoll) && t < tryState.leg.timer

            % Increment the bot
            tryState.cg(1) = tryState.cg(1) + dxStance;

            % Update the roll state
            [tryState.cg, tryState.ipm] = invPendulumModel( ...
                tryState.cg, tryState.ipm, tryState.leg.state, ...
                tryFoot, dt);

            % Increment sim time
            t = t + dt;
        end
```

Figure A.11. MATLAB code for the Model-Predictive Control (MPC) function (part 1 of 3) of the ST3LMR simulation.



```
        % Save results
        count = count + 1;
        if negMaxRoll < tryState.ipm.theta || ...
            tryState.ipm.theta < posMaxRoll
            tryHist(count,1:4) = [tryYf, 1, tryState.cg(2), ...
                tryState.cg(2)];
        else
            % Exceeds maximum roll
            tryHist(count,1:4) = [tryYf, 0, 999, 999];
        end

        % Increment the sim
        tryYf = tryYf + dYf;
    end

    % Eliminate failed passes
    tmp = find(tryHist(:,2) == 1);
    if ~isempty(tmp)
        if tmp(1) > 1
            yfStart = tryHist(tmp(1,1)-1,1);
        else
            yfStart = tryHist(tmp(1,1),1);
        end
        if tmp(length(tmp)) < count
            yfStop = tryHist(tmp(length(tmp))+1,1);
        else
            yfStop = tryHist(tmp(length(tmp)),1);
        end
    else
        tryPassed = 0;
    end

    % Increment the breadth pass
    nPass = nPass + 1;
end

% Depth-Limited Search
nSteps = nSteps + 1;
if nSteps <= searchDepth
    for i = 1:nHist
        if tryHist(i,2) == 1

            tryYf = tryHist(i,1);

            % Copy initial conditions
            tryHip =   inHip;
            tryFoot =  inFoot;
            tryState = inState;

            % Update the leg state
            [tryState.leg, tryFoot] = stateMachine(tryState.cg, ...
                tryState.ipm, tryHip, tryState.leg, tryFoot, ...
                params.timeFlight, params.halfStride, tryYf, tStart,
                dt, dxStance);
```

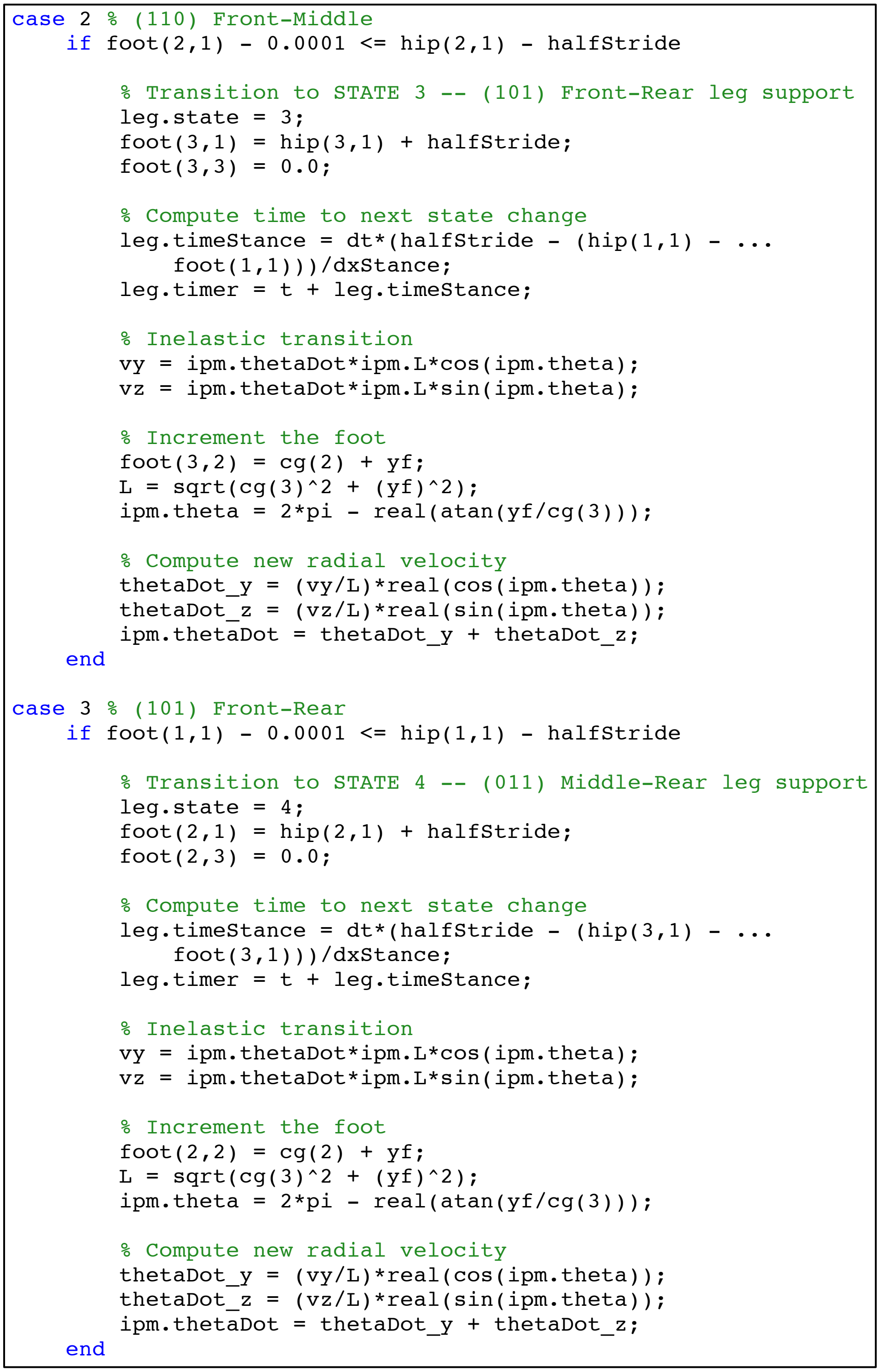
Figure A.12. MATLAB code for the Model-Predictive Control (MPC) function (part 2 of 3) of the ST3LMR simulation.

```
            % Simulation loop
            t = tStart;
            while (negMaxRoll < tryState.ipm.theta || ...
                tryState.ipm.theta < posMaxRoll) && ...
                t < tryState.leg.timer

                % Increment the bot
                tryState.cg(1) = tryState.cg(1) + dxStance;

                % Update the roll state
                [tryState.cg, tryState.ipm] = invPendulumModel(...
                    tryState.cg, tryState.ipm, tryState.leg.state, ...
                    tryFoot, dt);

                % Increment sim time
                t = t + dt;
            end

            % Update hip locations
            tryHip = hipLocations(tryState.cg, params.hipSep);

            % Take another step...
            [tmp, tryHist(i,2), tryHist(i,4)] = mpcSim(params, ...
                tryState, tryHip, tryFoot, t, dt, dxStance, nSteps);
        end
    end
end

% Select yf using a Goal State
tmp = find(tryHist(:,2) == 1);
if ~isempty(tmp)
    tmpHist = tryHist(tmp,:);
    tmpHist(:,3) = abs(tmpHist(:,3)) + abs(tmpHist(:,4));
    % GOAL: Minimize CG_y
    tmp = find(tmpHist(:,3) == min(tmpHist(:,3)));
    bestYf = tmpHist(tmp(1,1),1);
    bestCG = tmpHist(tmp(1,1),4);
else
    bestYf = 0;
    tryPassed = 0;
    bestCG = 9999;
end

return;
```

Figure A.13. MATLAB code for the Model-Predictive Control (MPC) function (part 3 of 3) of the ST3LMR simulation.

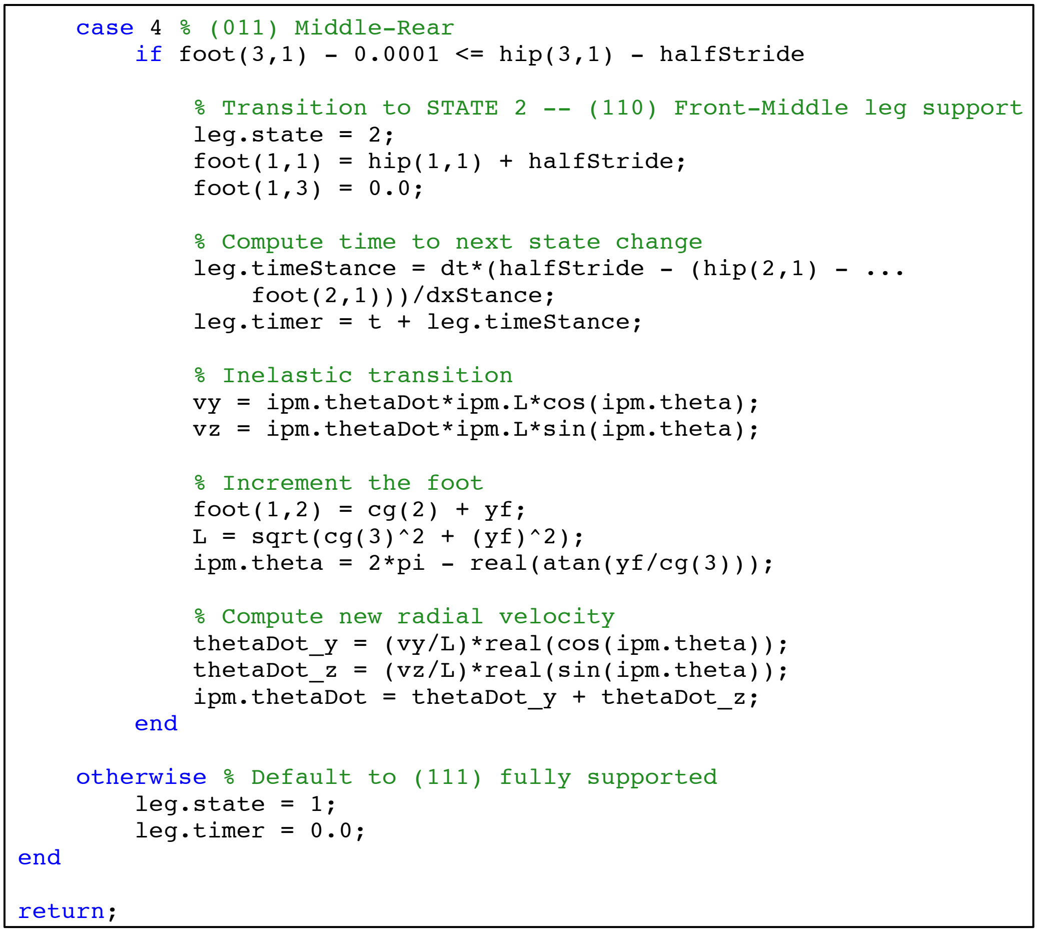
```
% John Goulding's ST3LMR MATLAB Simulation ©2013
function [leg, foot] = stateMachine(cg, ipm, hip, leg, foot, ...
    timeFlight, halfStride, yf, t, dt, dxStance)

% BACKWARD WAVE GAIT has four (4) allowable leg support states:
%   1 - (111) Fully supported
%   2 - (110) Front-Middle
%   3 - (101) Front-Rear
%   4 - (011) Middle-Rear

switch leg.state
    case 1 % (111) Fully supported
        % Check for maximum rearward travel of each leg
        if foot(3,1) - 0.0001 <= hip(3,1) - halfStride

            % Transition to (110) Front-Middle leg support
            leg.state = 2;
            foot(1,1) = hip(1,1) + halfStride;
            foot(1,3) = 0.0;

            % Compute time to next state change
            leg.timeStance = dt*(halfStride - (hip(2,1) - ...
                foot(2,1)))/dxStance;
            leg.timer = t + leg.timeStance;
        else
            if foot(2,1) - 0.0001 <= hip(2,1) - halfStride

                % Transition to (101) Front-Rear leg support
                leg.state = 3;
                foot(3,1) = hip(3,1) + halfStride;
                foot(3,3) = 0.0;

                % Compute time to next state change
                leg.timeStance = dt*(halfStride - (hip(1,1) - ...
                    foot(1,1)))/dxStance;
                leg.timer = t + leg.timeStance;
            else
                if foot(1,1) - 0.0001 <= hip(1,1) - halfStride

                    % Transition to (011) Middle-Rear leg support
                    leg.state = 4;
                    foot(2,1) = hip(2,1) + halfStride;
                    foot(2,3) = 0.0;

                    % Compute time to next state change
                    leg.timeStance = dt*(halfStride - (hip(3,1) - ...
                        foot(3,1)))/dxStance;
                    leg.timer = t + leg.timeStance;
                end
            end
        end
```

Figure A.14. MATLAB code for the State Machine function (part 1 of 3) of the ST3LMR simulation.



```
case 2 % (110) Front-Middle
    if foot(2,1) - 0.0001 <= hip(2,1) - halfStride

        % Transition to STATE 3 -- (101) Front-Rear leg support
        leg.state = 3;
        foot(3,1) = hip(3,1) + halfStride;
        foot(3,3) = 0.0;

        % Compute time to next state change
        leg.timeStance = dt*(halfStride - (hip(1,1) - ...
            foot(1,1)))/dxStance;
        leg.timer = t + leg.timeStance;

        % Inelastic transition
        vy = ipm.thetaDot*ipm.L*cos(ipm.theta);
        vz = ipm.thetaDot*ipm.L*sin(ipm.theta);

        % Increment the foot
        foot(3,2) = cg(2) + yf;
        L = sqrt(cg(3)^2 + (yf)^2);
        ipm.theta = 2*pi - real(atan(yf/cg(3)));

        % Compute new radial velocity
        thetaDot_y = (vy/L)*real(cos(ipm.theta));
        thetaDot_z = (vz/L)*real(sin(ipm.theta));
        ipm.thetaDot = thetaDot_y + thetaDot_z;
    end

case 3 % (101) Front-Rear
    if foot(1,1) - 0.0001 <= hip(1,1) - halfStride

        % Transition to STATE 4 -- (011) Middle-Rear leg support
        leg.state = 4;
        foot(2,1) = hip(2,1) + halfStride;
        foot(2,3) = 0.0;

        % Compute time to next state change
        leg.timeStance = dt*(halfStride - (hip(3,1) - ...
            foot(3,1)))/dxStance;
        leg.timer = t + leg.timeStance;

        % Inelastic transition
        vy = ipm.thetaDot*ipm.L*cos(ipm.theta);
        vz = ipm.thetaDot*ipm.L*sin(ipm.theta);

        % Increment the foot
        foot(2,2) = cg(2) + yf;
        L = sqrt(cg(3)^2 + (yf)^2);
        ipm.theta = 2*pi - real(atan(yf/cg(3)));

        % Compute new radial velocity
        thetaDot_y = (vy/L)*real(cos(ipm.theta));
        thetaDot_z = (vz/L)*real(sin(ipm.theta));
        ipm.thetaDot = thetaDot_y + thetaDot_z;
    end
```

Figure A.15. MATLAB code for the State Machine function (part 2 of 3) of the ST3LMR simulation.



```
    case 4 % (011) Middle-Rear
        if foot(3,1) - 0.0001 <= hip(3,1) - halfStride

            % Transition to STATE 2 -- (110) Front-Middle leg support
            leg.state = 2;
            foot(1,1) = hip(1,1) + halfStride;
            foot(1,3) = 0.0;

            % Compute time to next state change
            leg.timeStance = dt*(halfStride - (hip(2,1) - ...
                foot(2,1)))/dxStance;
            leg.timer = t + leg.timeStance;

            % Inelastic transition
            vy = ipm.thetaDot*ipm.L*cos(ipm.theta);
            vz = ipm.thetaDot*ipm.L*sin(ipm.theta);

            % Increment the foot
            foot(1,2) = cg(2) + yf;
            L = sqrt(cg(3)^2 + (yf)^2);
            ipm.theta = 2*pi - real(atan(yf/cg(3)));

            % Compute new radial velocity
            thetaDot_y = (vy/L)*real(cos(ipm.theta));
            thetaDot_z = (vz/L)*real(sin(ipm.theta));
            ipm.thetaDot = thetaDot_y + thetaDot_z;
        end

    otherwise % Default to (111) fully supported
        leg.state = 1;
        leg.timer = 0.0;
end

return;
```

Figure A.16. MATLAB code for the State Machine function (part 3 of 3) of the ST3LMR simulation.

APPENDIX B

COUPLED-DRIVE LEG EXPERIMENTAL RESULTS

Appendices B and C implement and discuss the results of programming the Draisine (wheel-leg-wheel), shown in Figure B.1 and described in Chapter 8. The Draisine robot replaces the front and rear legs of the Single Track Three Legged Mobile Robot (ST3LMR) with wheels. Like the early Drasine, a pedal-less precursor to the bicycle, the Draisine robot is capable of executing both straight and curved single-track trajectories. Through a series of five experiments that increase in complexity, the fundamental approach to maintaining balance – shifting the foot to the left or right of the center of gravity to move the line of support – is tested. This chapter presents the first two experiments designed to provide basic leg cycle movement and force feedback for controlled motion.

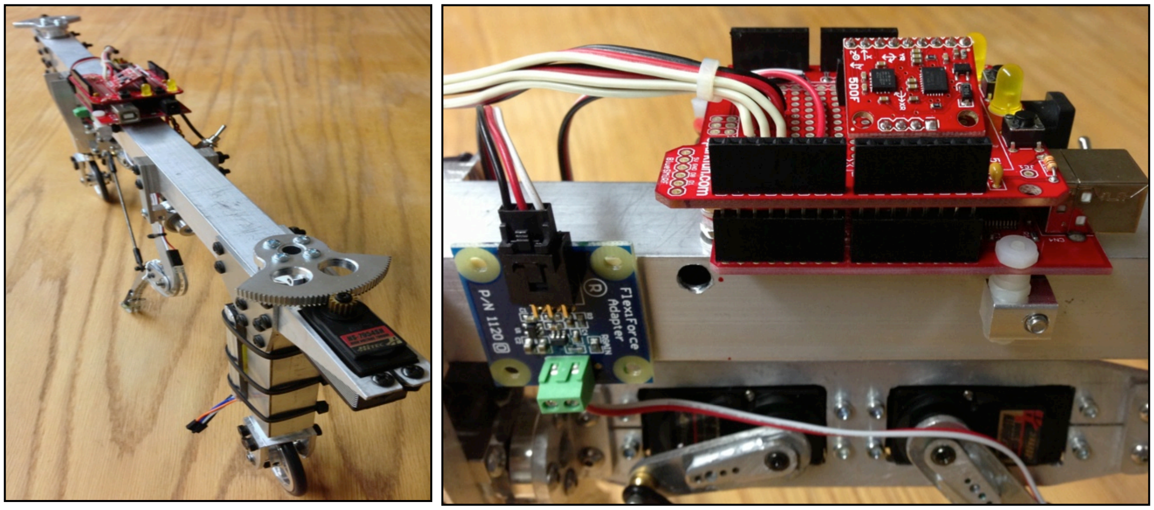


Figure B.1. Experimental 1/3-scale (9" leg) wheel-leg-wheel prototype robot.

Briefly, the Draisine robot comprises a single Hirose style leg, actuated for motion in the x- and z-axis and mounted on a theta stage to rotate the leg outward, normal to the major direction of motion (the x-axis). The theta stage is mounted to a stiff, lightweight tubular aluminum frame measuring 24.0 inches in length and standing 10.5 inches tall (at the center line of the MEMS accelerometer and gyroscope, shown in the left pane of Figure B.1). The one-behind-the-other or in-line attachment of the wheels and legs to the body results in a center of gravity and a center of pressure that are directly in line with the wheels and leg when the leg is simply extended straight down from the body, resulting in inherent instability in the roll axis. A BasicATOM Pro 40m microcontroller executes 32 bit math for inverse kinematics. A FlexForce sensor and MEMS accelerometer and gyro provide foot pressure and roll angle and acceleration feedback.

B.1. Experiment 1 – Basic Leg Cycle

The objective of this first experiment is to program the BasicATOM Pro 40m micro-controller as a State machine to execute a basic leg cycle, selectively performing a stance-to-flight phase, a flight phase, a flight-to-stance phase, and a stance phase (discussed in Chapter 4). In this experiment, the theta stage is turned off, and only motion in the x-axis (along the major length of the body and forward direction of travel) and z-axis (leg extension and retraction) is controlled. The foot position is recorded over time using a digital camera and timer, and the results are compared against the desired trajectory. From these measurements, velocity and acceleration models for the actuator are developed.

The stance-to-flight phase, called the Unload state, involves controlling foot movement that (rapidly) unloads reaction forces between the foot and the ground such that the foot is lifted off the ground (in the z-axis) while continuing to increment the foot in the x-axis, i.e., the foot is stationary in the x-axis with respect to the ground. This provides a smooth transition, allowing the robot to coast as the foot is repositioned. Figures B.2 and B.3 are a block diagram and code illustrating the input parameters and control steps involved with the Unload state. Foot position is incremented upward in the z-axis using the rate, dz. The servo angles are calculated using inverse kinematics, given the leg length parameters. When the servos can accept new commands, the new servo angles are loaded along with the desired servo rate. Otherwise, the program waits at this point. Loading the servos completes the Unload state, and control is transitioned to the Flight state.

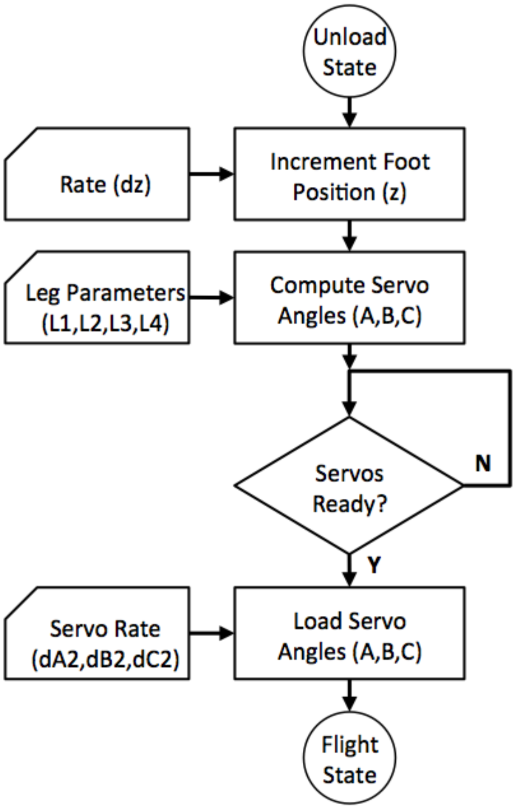
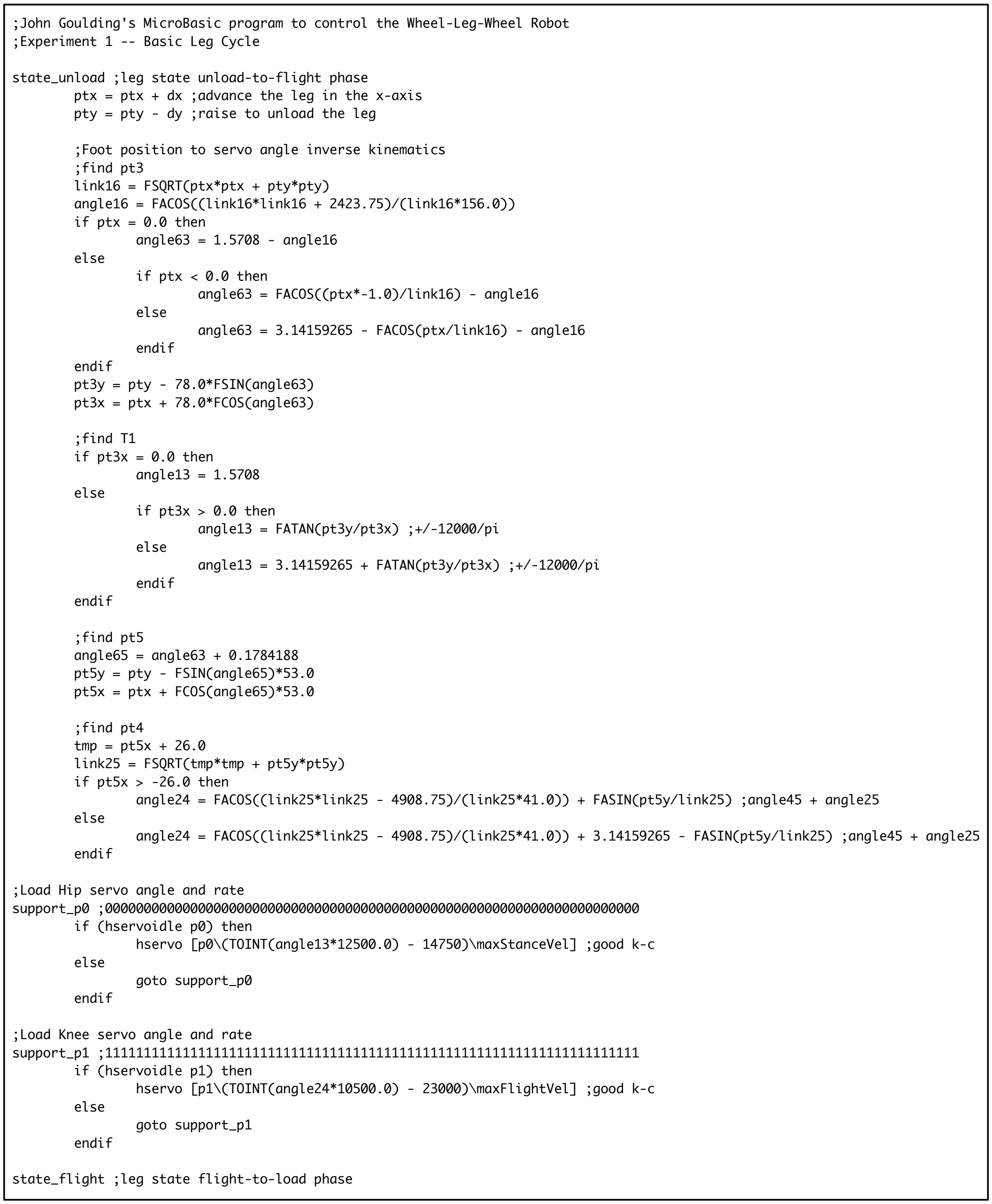

Unload State
Rate (dz)
Increment Foot Position (z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servos Ready?
N
Y
Servo Rate (dA2,dB2,dC2)
Load Servo Angles (A,B,C)
Flight State

Figure B.2. Unload state block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 1 -- Basic Leg Cycle

state_unload ;leg state unload-to-flight phase
        ptx = ptx + dx ;advance the leg in the x-axis
        pty = pty - dy ;raise to unload the leg

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxStanceVel] ;good k-c
        else
                goto support_p0
        endif

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxFlightVel] ;good k-c
        else
                goto support_p1
        endif

state_flight ;leg state flight-to-load phase
```

Figure B.3. Unload state MicroBasic code.

The flight phase, called the Flight state, repositions the foot to the leg start position at a generally faster rate than the direction of motion of the body. Figures B.4 and B.5 are the Flight state block diagram and code. On entering the Flight state, the desired foot start position is computed using inverse kinematics. Note the Hirose leg arcs the foot upward and away from the ground when the hip servo is commanded faster than the knee servo; so, only a single servo command is needed. Generally, the hip servo rate is the maximum rate possible. When the servos can accept new commands, the new servo angles are loaded along with the desired server rate. Loading the servos completes the Flight state, and control is transitioned to the Load state.

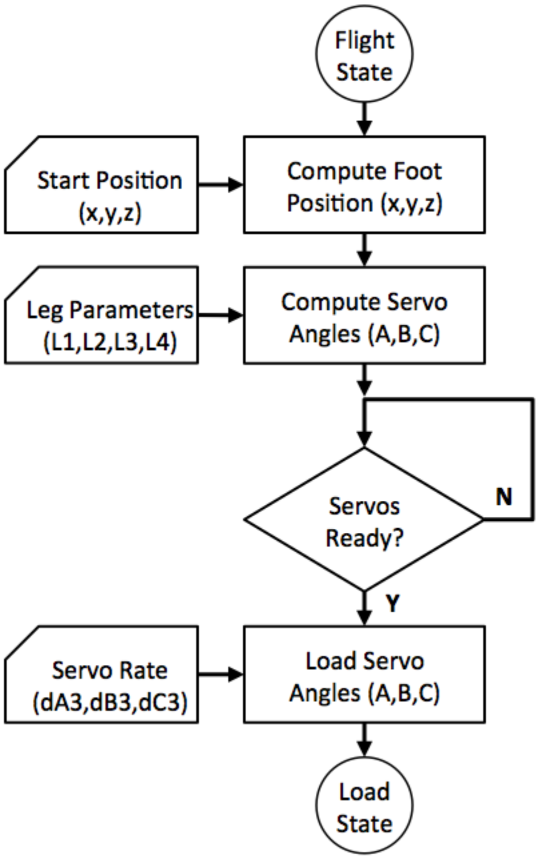

Flight State
Start Position (x,y,z)
Compute Foot Position (x,y,z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servos Ready?
N
Y
Servo Rate (dA3,dB3,dC3)
Load Servo Angles (A,B,C)
Load State

Figure B.4. Flight state block diagram.

```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 1 -- Basic Leg Cycle

state_flight ;leg state flight-to-load phase
        ptx = ptxStartPos ;advance the leg to the leg cycle start position
        pty = ptyStartPos

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxFlightVel] ;good k-c
        else
                goto support_p0
        endif

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxFlightVel] ;good k-c
        else
                goto support_p1
        endif

state_load ;leg state load-to-stance phase
```

Figure B.5. Flight state MicroBasic code.

The flight-to-stance phase, called the Load state, reverses the servos to match the expected ground speed and lowers the foot to the expected ground position to develop reaction forces between the foot and the ground. Figures B.6 and B.7 are the Load state block diagram and code. Foot position is incremented in the x- and z-axis, and servo angles are calculated using inverse kinematics. When the servos can accept new commands, the new servo angles are loaded along with the desired servo rate. Otherwise, the program waits at this point. Loading the servos completes the Load state, and control is transitioned to the Stance state.

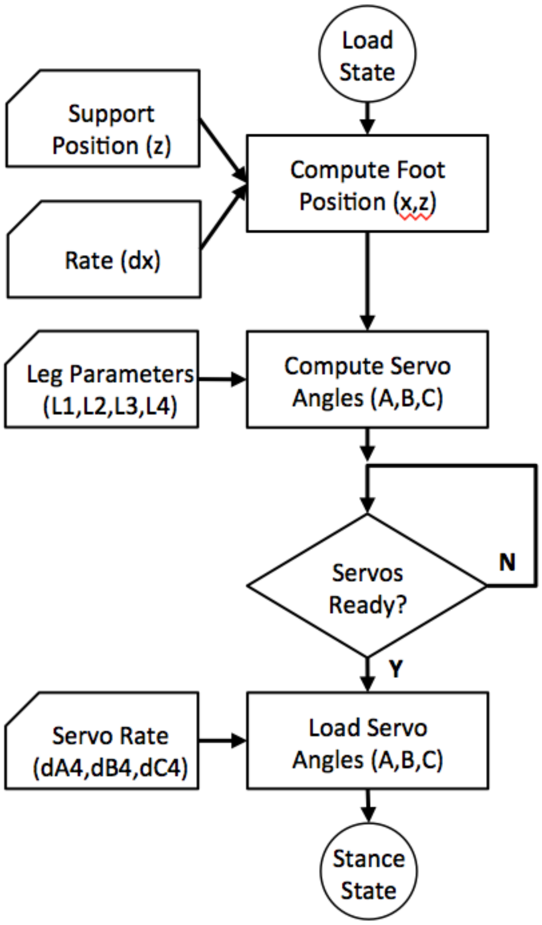
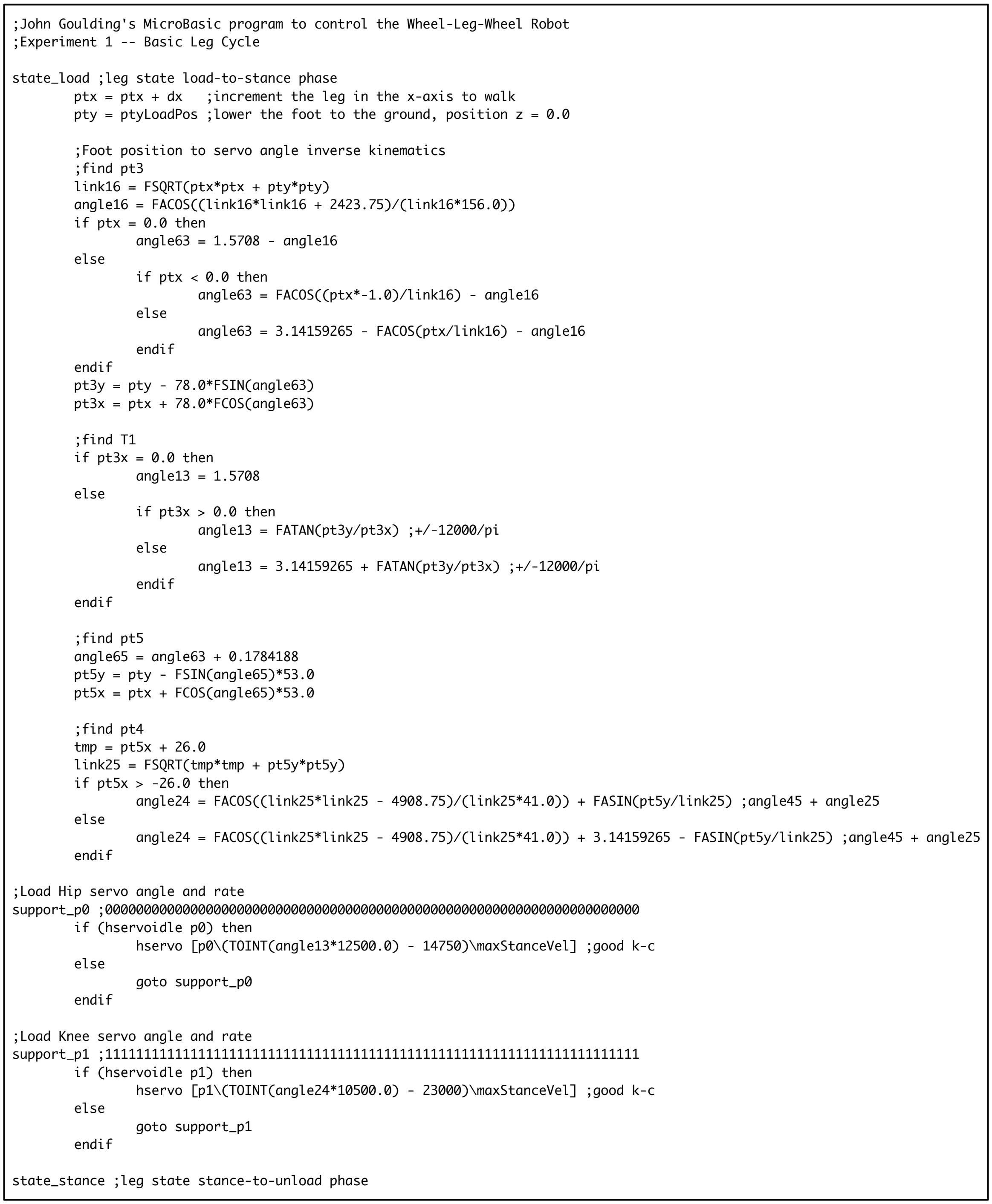

Load State
Support Position (z)
Rate (dx)
Compute Foot Position (x,z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servos Ready?
N
Y
Servo Rate (dA4,dB4,dC4)
Load Servo Angles (A,B,C)
Stance State

Figure B.6. Load state block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 1 -- Basic Leg Cycle

state_load ;leg state load-to-stance phase
        ptx = ptx + dx   ;increment the leg in the x-axis to walk
        pty = ptyLoadPos ;lower the foot to the ground, position z = 0.0

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxStanceVel] ;good k-c
        else
                goto support_p0
        endif

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxStanceVel] ;good k-c
        else
                goto support_p1
        endif

state_stance ;leg state stance-to-unload phase
```

Figure B.7. Load state MicroBasic code.

The stance phase, called the Stance state, involves controlling foot motion such that foot-to-ground interaction develops reaction forces that are transferred from the foot through the leg to propel the Draisine robot in the x-axis. Figures B.8 and B.9 are the Stance state block diagram and code. Foot position is computed in a piecewise linear fashion or compute-wait-load loop to approximate a straight-line trajectory in the z-axis. The stance state increments the foot in the x-axis direction until the desired foot travel limit is reached. Once the travel limit is reached, the Stance state transitions to the Unload state, and the basic leg cycle repeats.

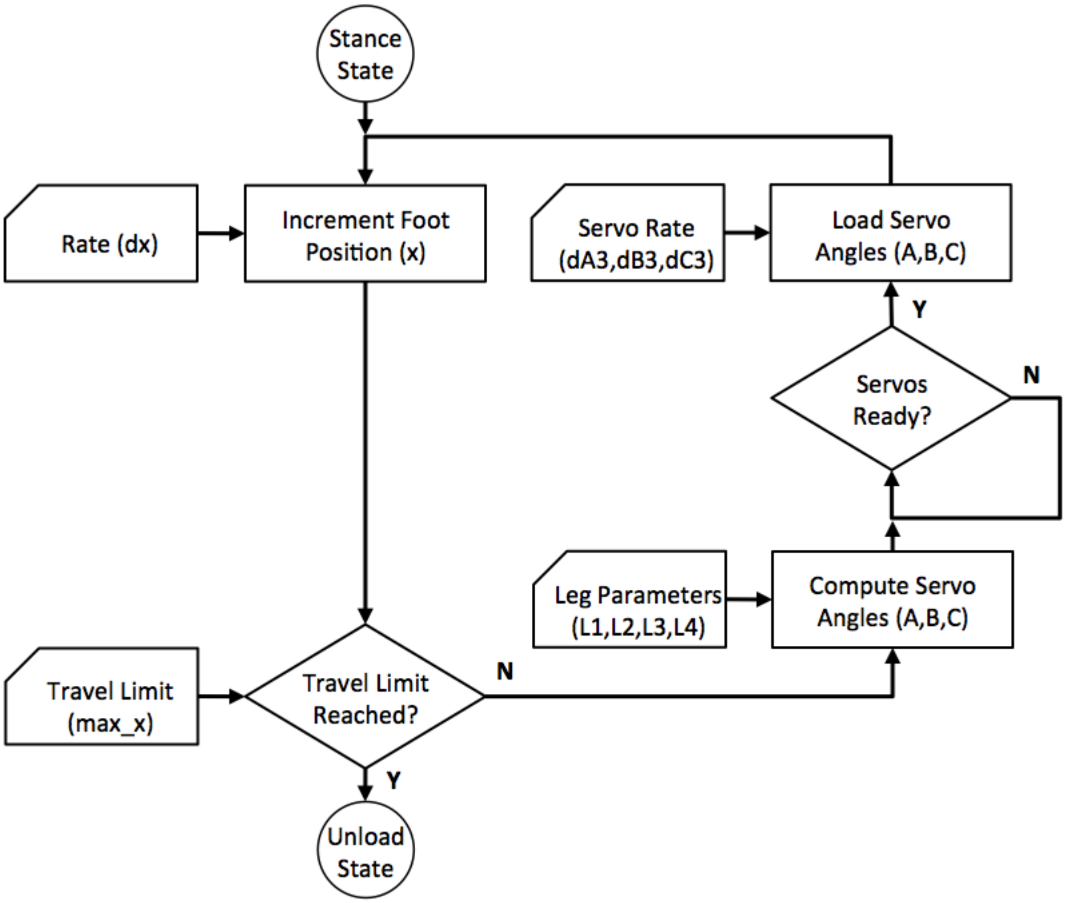
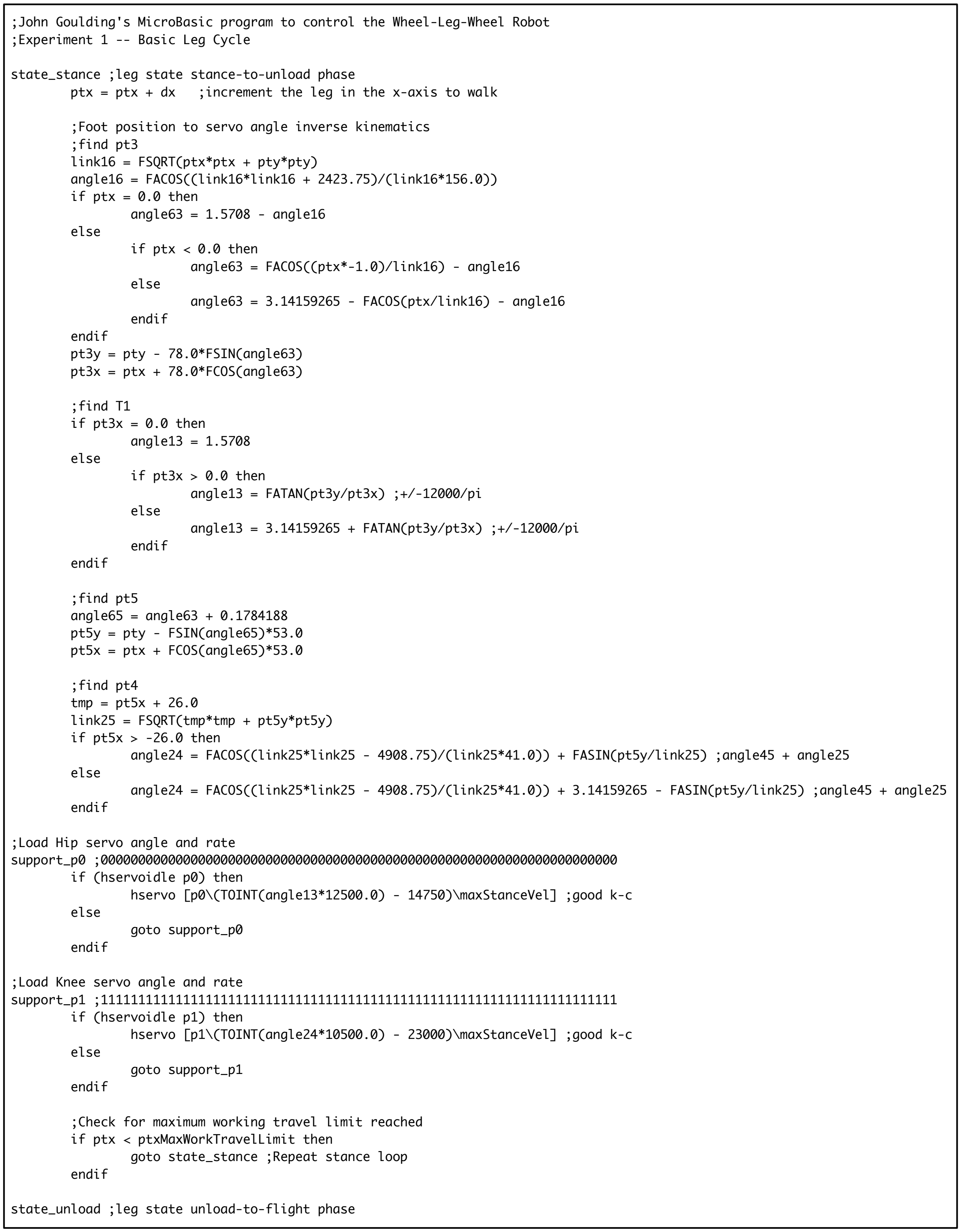


Figure B.8. Stance state block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 1 -- Basic Leg Cycle

state_stance ;leg state stance-to-unload phase
        ptx = ptx + dx   ;increment the leg in the x-axis to walk

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxStanceVel] ;good k-c
        else
                goto support_p0
        endif

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxStanceVel] ;good k-c
        else
                goto support_p1
        endif

        ;Check for maximum working travel limit reached
        if ptx < ptxMaxWorkTravelLimit then
                goto state_stance ;Repeat stance loop
        endif

state_unload ;leg state unload-to-flight phase
```

Figure B.9. Stance state MicroBasic code.

For the Hirose leg, it is necessary to move the servos at different rates to achieve a near straight-line motion. The x-axis rate dx is adjusted, called tuning, such that the overall time as it takes to compute the next foot position, check for travel limit reached, and compute the new servo angles through inverse kinematic equations can all be accomplished at about the same time it takes the servos to reach the previously commanded position. When dx is properly tuned (or timed), the new servo commands are sent without entering the wait loop. Tuning ensures the foot motion is as seamless (without hesitation) as possible. Recall, the Hirose leg parameters were optimized to produce the flattest possible trajectory given constant servo motion. The purpose of optimizing the leg parameters will become more important as the complexity or the number of commands executed during the stance state increases and thus dx increases.

Figure B.10 shows how leg position was measured using the images of the leg, stop watch (bottom foreground), and calibrated grid paper. Using a Digital Image Processing (DIP) algorithm, the tip of the foot was located and measured, with respect to an absolute coordinate system defined by the grid paper. Table B.2 presents rounded averages of the raw data measured over three leg cycles, wherein the first column is the camera image or frame number, the second column is the time stamp displayed on the stop watch to the nearest 1/10 second, the third column is the interpolated time rounded to the nearest 1/100 second, the fourth column is the absolute foot position in the x-axis, and the fifth column is the absolute foot position in the z-axis. Figures B.11 and B.12 illustrate two different plot styles for the data in Table B.2.

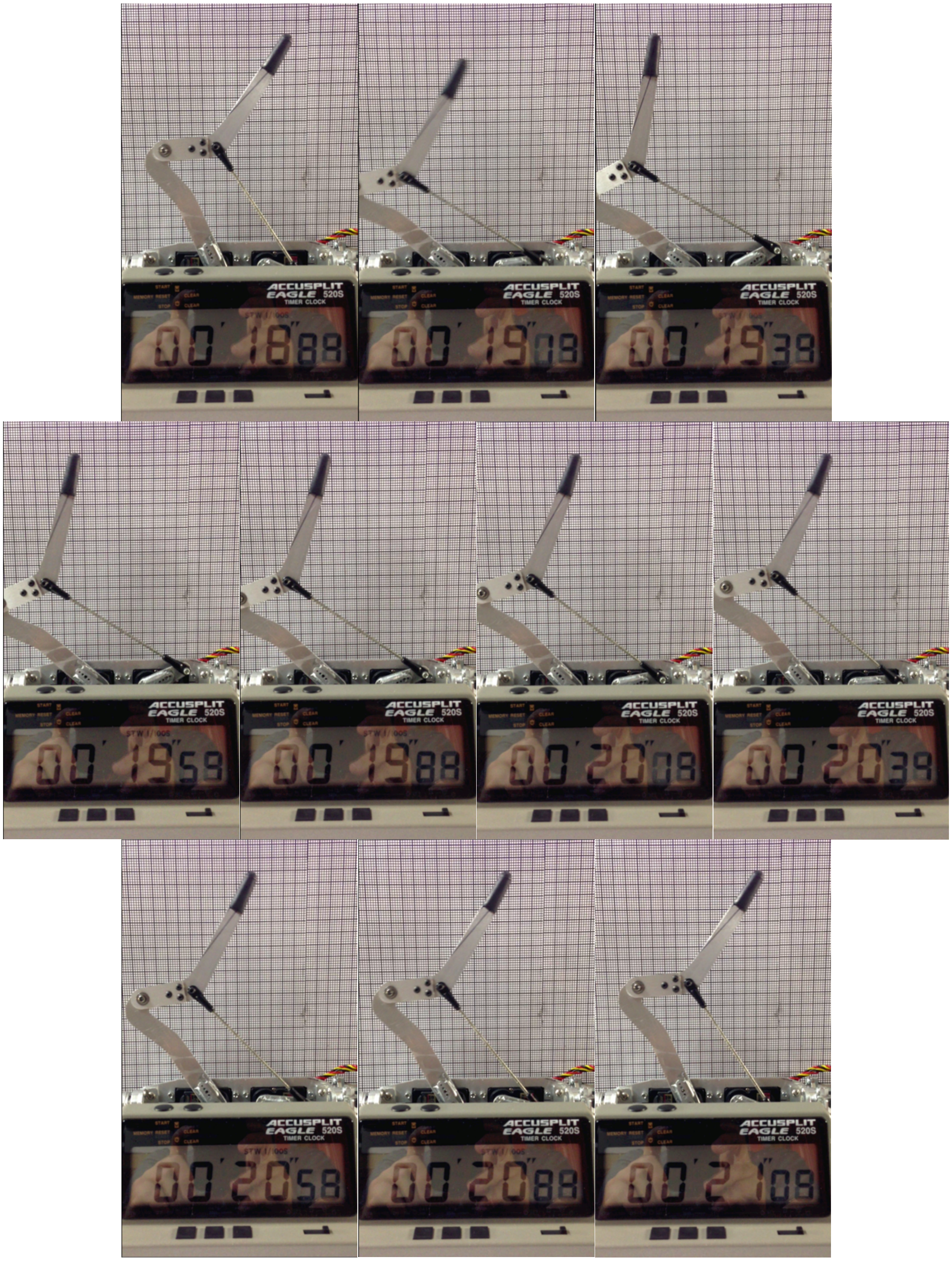


Figure B.10. Basic leg motion, select frames.

Table B.1. Open-loop foot position data.

FRAME	TIME	RT	X	Z
0	18.8	18.75	18.0	0.0
1	18.8	18.79	19.0	-4.5
2	18.8	18.82	23.0	-7.0
3	18.9	18.86	29.0	-10.0
4	18.9	18.89	37.0	-12.0
5	18.9	18.92	45.0	-13.0
6	19.0	18.96	54.0	-15.0
7	19.0	18.99	62.5	-18.0
8	19.0	19.03	70.0	-18.0
9	19.1	19.06	75.0	-17.0
10	19.1	19.10	81.0	-12.0
11	19.1	19.13	87.5	-9.5
12	19.2	19.16	94.5	-6.0
13	19.2	19.20	100.5	-2.0
14	19.2	19.23	105.0	0.5
15	19.3	19.27	106.0	1.5
16	19.3	19.30	106.5	2.0
17	19.3	19.33	106.0	2.0
18	19.4	19.37	105.0	2.0
19	19.4	19.40	103.0	2.0
20	19.4	19.44	101.5	1.5
21	19.5	19.47	100.0	1.0
22	19.5	19.50	98.0	1.0
23	19.5	19.54	96.5	0.5
24	19.5	19.57	94.5	0.5
25	19.6	19.61	93.0	0.0
26	19.6	19.64	91.5	0.0
27	19.7	19.67	90.0	0.0
28	19.7	19.71	89.0	0.0
29	19.7	19.74	87.0	0.0
30	19.8	19.78	85.5	0.0
31	19.8	19.81	84.0	0.0
32	19.8	19.85	82.0	0.0
33	19.9	19.88	81.0	0.0
34	19.9	19.91	79.5	0.0
35	19.9	19.95	78.0	0.0
36	20.0	19.98	76.0	0.0
37	20.0	20.02	74.5	0.0
38	20.1	20.05	73.0	0.0
39	20.1	20.08	72.0	0.0
40	20.1	20.12	70.0	0.5
41	20.2	20.15	69.0	1.0
42	20.2	20.19	67.0	1.0
43	20.2	20.22	65.0	1.0
44	20.3	20.25	62.5	0.5
45	20.3	20.29	61.0	0.5
46	20.3	20.32	58.5	0.0
47	20.4	20.36	57.0	0.0
48	20.4	20.39	55.5	0.0
49	20.4	20.42	53.5	0.0
50	20.5	20.46	51.5	0.0
51	20.5	20.49	49.5	0.0
52	20.5	20.53	47.5	0.0
53	20.6	20.56	46.0	0.0
54	20.6	20.60	44.0	0.0
55	20.6	20.63	41.5	0.0
56	20.7	20.66	40.0	0.0
57	20.7	20.70	38.5	0.0
58	20.7	20.73	36.0	0.0
59	20.8	20.77	34.5	0.0
60	20.8	20.80	32.5	0.0
61	20.8	20.83	30.5	0.0
62	20.9	20.87	29.0	0.0
63	20.9	20.90	26.0	0.0
64	20.9	20.94	25.0	0.0
65	21.0	20.97	23.5	0.0
66	21.0	21.00	21.5	0.0
67	21.0	21.04	19.5	0.0

Figure B.11 shows the open-loop performance of the coupled leg design, wherein the horizontal axis is the frame number or time. The blue upper curve is the position of the foot in the x-axis (along the major axis or forward direction of the Draisine robot). The red lower curve is the position of the foot in the z-axis or height, where zero is the ground (defined by the line intersecting the front and rear wheels) and negative values are raising the foot up and towards the Draisine robot frame. An ideal trajectory would comprise flat-line segments.

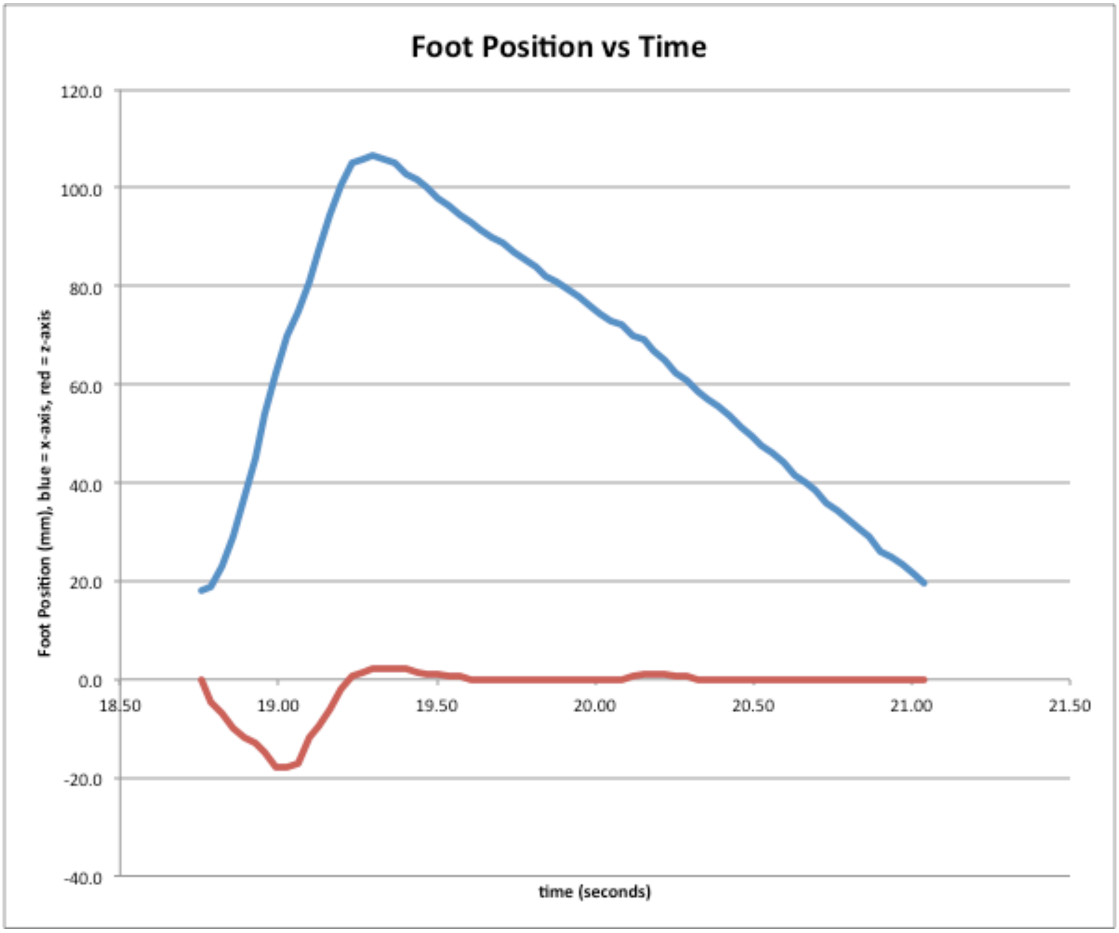


Figure B.11. Basic leg motion plotting foot position in the x- and z-axis (top blue and bottom red curves, respectively) over time along the horizontal axis.

Figure B.12 shows the open-loop performance of the coupled leg design, wherein the horizontal axis is the position of the foot in the x-axis (along the major axis or forward direction of the Draisine robot) and the vertical axis is the position of the foot in the z-axis or height, where zero is the ground (defined by the line intersecting the front and rear wheels) and negative values are raising the foot up and towards the Draisine robot frame. The basic leg cycle begins at the upper left coordinates [18.0, 0.0] where the foot is unloaded and commanded to fly to a start position of [106.5, 0.0]. The lower arc is the foot in Flight phase. The upper line is the foot in Stance phase.

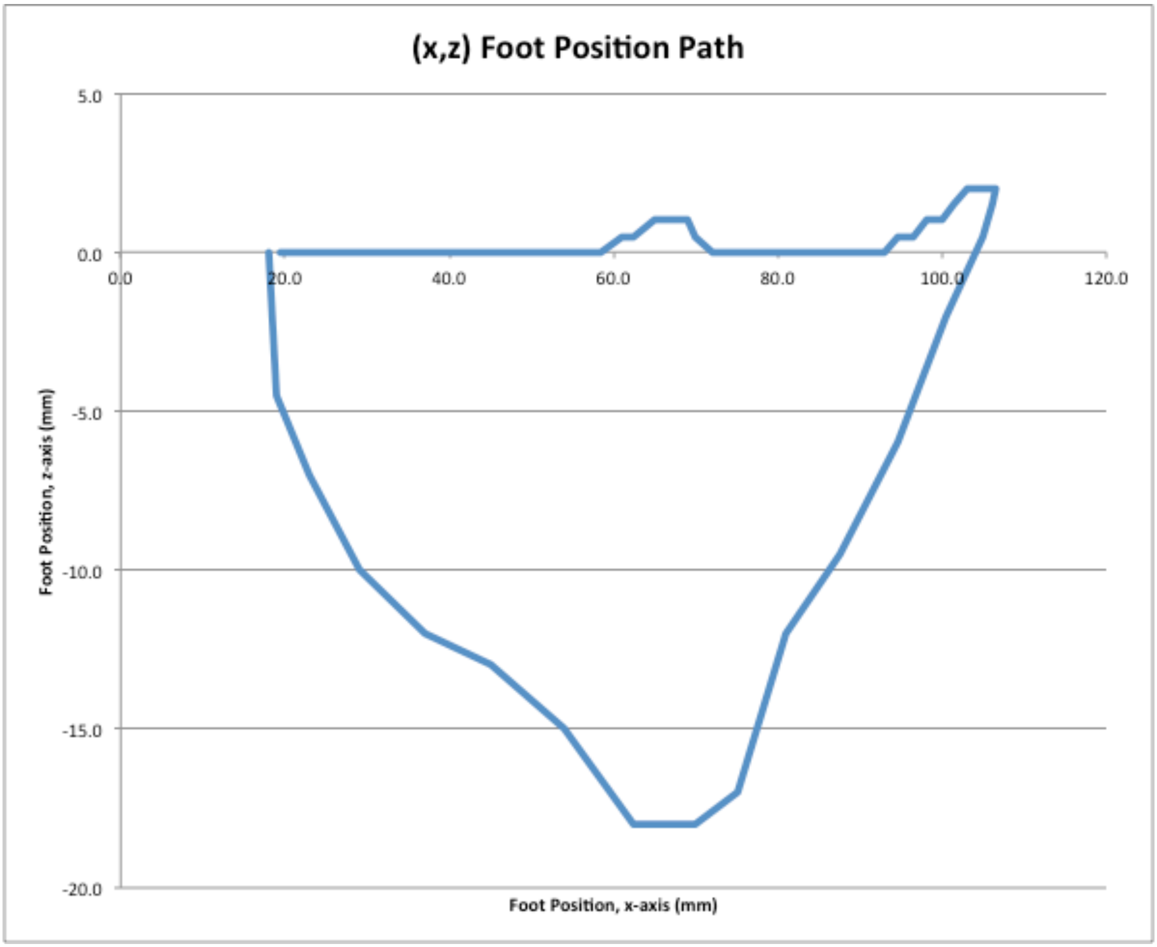


Figure B.12. Basic leg motion plotting foot position along the x- and z-axis.

Figure B.12 demonstrates the inaccuracy of open-loop, nonlinear servo control, when the foot overshoots the commanded position and travels below the z-axis (positive values in the vertical) to position [106.5, 2.0]. Unfortunately, there is no way to tune the HiTec servos. Other (more expensive) off-the-shelf servos, such as those made by Bioloid and described in Chapter 10, use tunable PID loops. Regardless, a better and preferred method is to implement closed-loop, force feedback control, and it is the objective of the next experiment.

B.2. Experiment 2 – Sensing and Controlling Foot Pressure

The second experiment involves adding a foot pressure sensor. The foot pressure sensor provides feedback when the foot is in contact with the ground. Maintaining sufficient foot pressure is necessary to maintain a sufficient static coefficient of friction between the foot and the ground to provide traction. Too little applied foot pressure and the foot will slip. Too much applied pressure will overstress and heat the motors, reducing their useful life. The correct amount of foot pressure is determined through experimentation.

Figure B.13 shows the FlexForce foot pressure sensor detail (left) and integration on leg (right). The 0-1.0 kg_f FlexiForce sensor, US patent 6,272,936, comprises a piezoelectric material printed on and sandwiched between two plastic films, with printed traces leading to connectors. The sensor is bonded to the distal end of the foot using an acrylic glue to provide a rigid backing against the rubber foot. A high-density polyethylene disk is glued to the working side of the sensor to both protect the sensor from abrasion and to provide a working surface to distribute the ground-to-foot contact forces over the surface of the sensor.

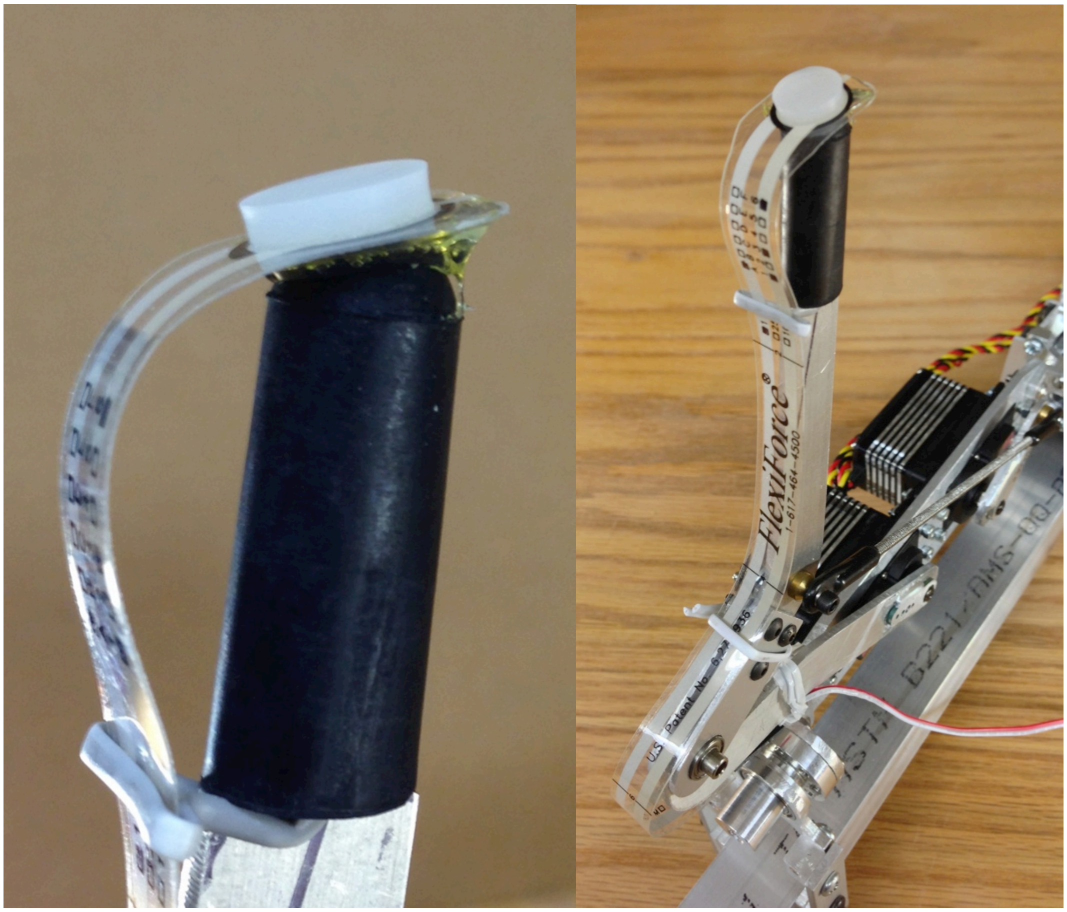
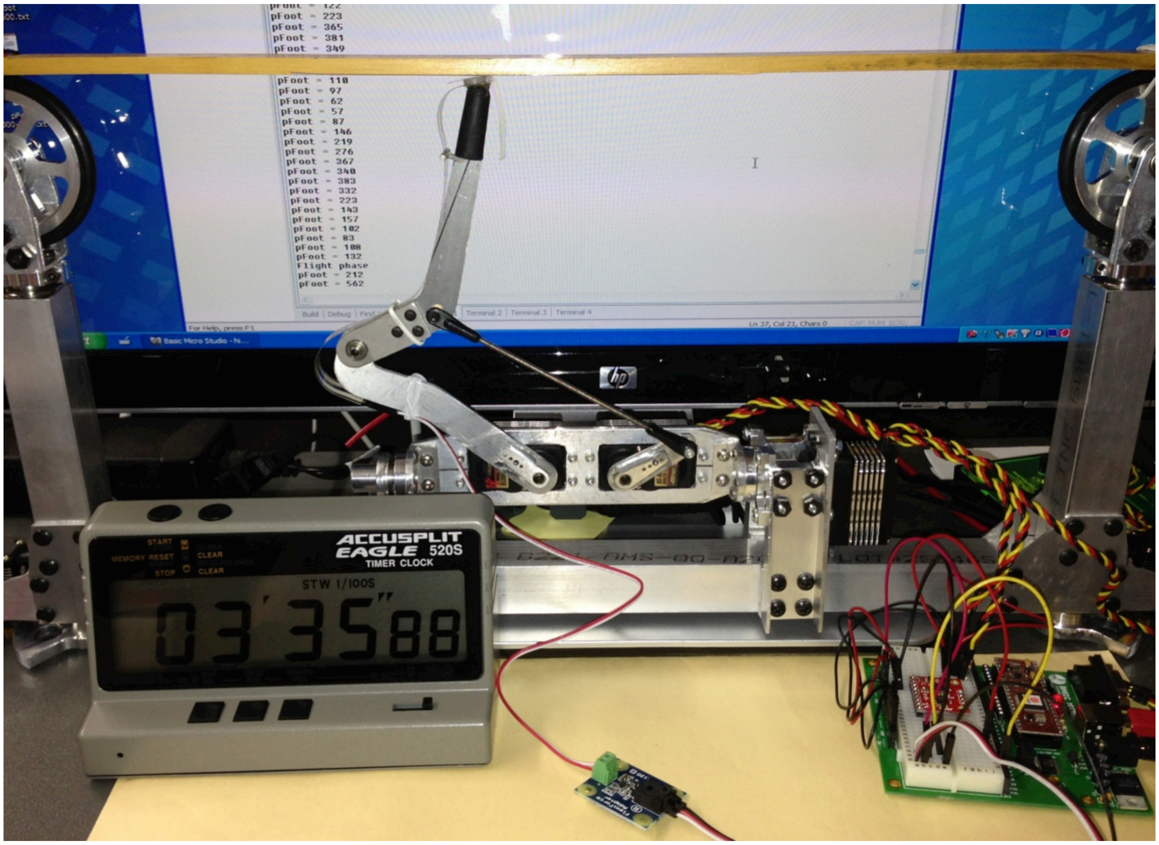


Figure B.13. Foot pressure sensor detail (left) and integration on leg (right).

Figure B.14 shows the experimental setup. The Draisine robot is inverted, with the foot pressing upward against a frictionless surface (¼-inch thick wood coated with a Teflon tape). From left to right, a timer, the FlexiForce signal conditioning board, and BasicMicro prototyping board are visible in the foreground. The computer display in the background is running a terminal window, showing the real-time serial output of the BasicMicro controller, as it executes the Stance state of the basic leg cycle described previously in Experiment 1.



```
pFoot = 122
pFoot = 223
pFoot = 365
pFoot = 381
pFoot = 349
pFoot = 110
pFoot = 97
pFoot = 62
pFoot = 57
pFoot = 87
pFoot = 146
pFoot = 219
pFoot = 276
pFoot = 367
pFoot = 340
pFoot = 383
pFoot = 332
pFoot = 223
pFoot = 143
pFoot = 157
pFoot = 102
pFoot = 83
pFoot = 108
pFoot = 132
Flight phase
pFoot = 212
pFoot = 562
```

Figure B.14. Foot pressure sensor experimental setup.

Neither the Unload state nor the Flight state control functions require input from the foot pressure sensor. The purpose of the Unload state is to rapidly lift the foot upward, relieving the foot-to-ground interaction forces. The purpose of the Flight state is to move the foot rapidly forward through the air to a position where the foot can be repositioned on the ground to support the Draisine robot in the next leg cycle. See Experiment 1 and Figures B.2 through B.5 for a discussion of the Unload and Flight states.

Figures B.15 and B.16 show the Load state block diagram and code with the addition of the foot pressure sensor input. As in the first experiment, position must be incremented in the x- and z-axes to both match the ground speed and reposition the foot to the ground to support the Draisine robot. Given the new foot position, the servo angles are computed using the leg parameters. Once the servo angles are computed, the servos are repetitively polled until they are ready to accept new commands. Once ready, the new servo angles and servo rates are loaded. The foot pressure sensor is read immediately after the servo angles are loaded.

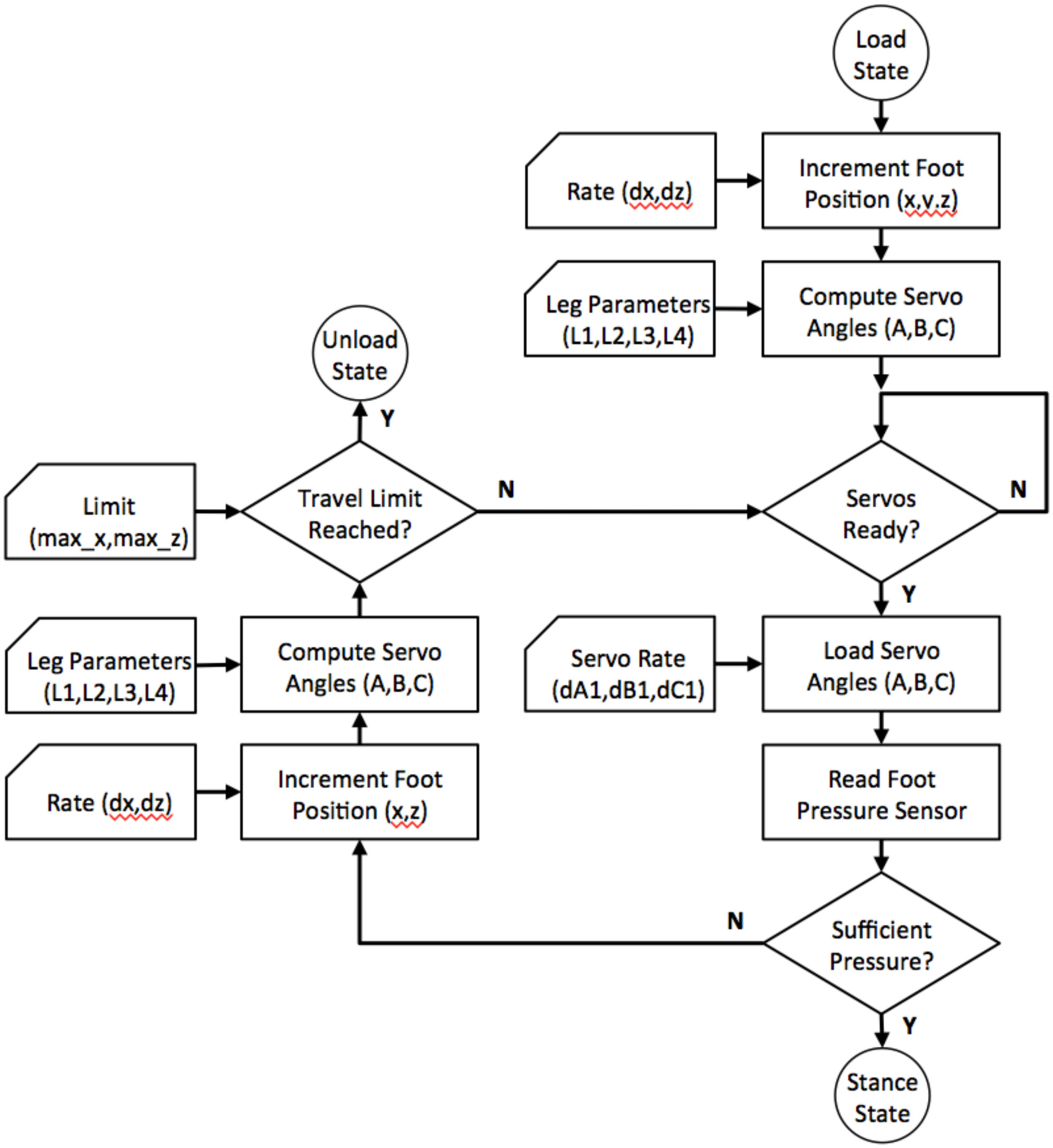
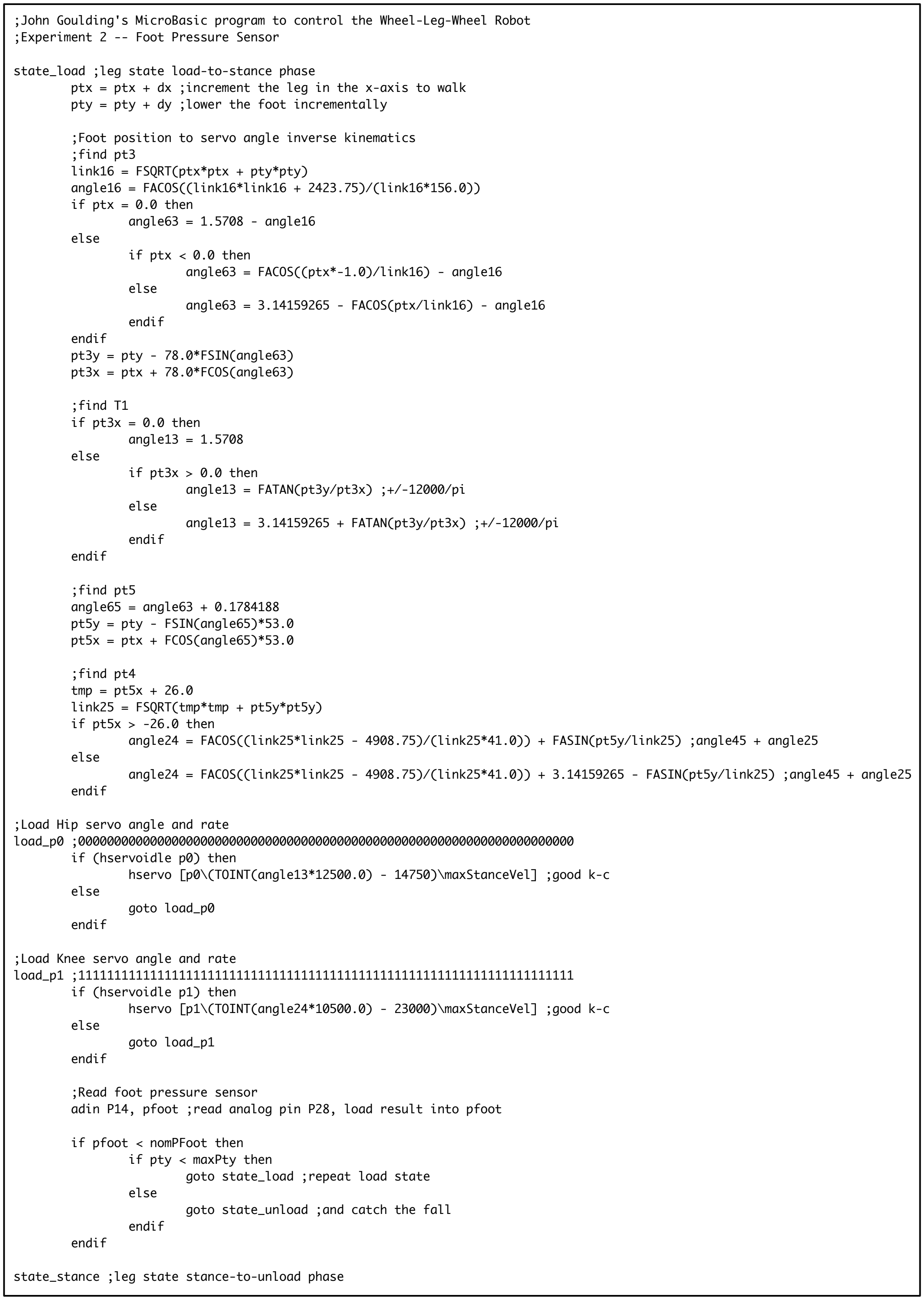

Load State
Rate (dx,dz)
Increment Foot Position (x,y,z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Unload State
Y
Limit (max_x,max_z)
Travel Limit Reached?
N
Servos Ready?
N
Y
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servo Rate (dA1,dB1,dC1)
Load Servo Angles (A,B,C)
Rate (dx,dz)
Increment Foot Position (x,z)
Read Foot Pressure Sensor
N
Sufficient Pressure?
Y
Stance State

Figure B.15. Load state with foot pressure sensor block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 2 -- Foot Pressure Sensor

state_load ;leg state load-to-stance phase
        ptx = ptx + dx ;increment the leg in the x-axis to walk
        pty = pty + dy ;lower the foot incrementally

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
load_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxStanceVel] ;good k-c
        else
                goto load_p0
        endif

;Load Knee servo angle and rate
load_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxStanceVel] ;good k-c
        else
                goto load_p1
        endif

        ;Read foot pressure sensor
        adin P14, pfoot ;read analog pin P28, load result into pfoot

        if pfoot < nomPFoot then
                if pty < maxPty then
                        goto state_load ;repeat load state
                else
                        goto state_unload ;and catch the fall
                endif
        endif

state_stance ;leg state stance-to-unload phase
```

Figure B.16. Load state with foot pressure sensor MicroBasic code.

Again referring to Figure B.15, if sufficient pressure has not been reached, the foot has not made sufficient ground contact and the foot position is again incremented in the x- and z-axes. If either the z-axis or x-axis travel limits are reached, the state machine transitions immediately to the Unload state. (Note that this would indicate an out-of-control situation, which is discussed in Experiment 3.) The loop repeats until sufficient foot pressure is reached.

When the foot is applying sufficient pressure to the ground, the state machine transitions to the Stance state. The purpose of the stance state is to support the Draisine robot and impart force in the x-axis to propel the Draisine robot forward. Figures B.17 and B.18 show the Stance state block diagram and code with the addition of the foot pressure sensor input. In the Stance state, the foot is incremented in the x-axis given the desired rate dx, new servo angles are computed by inverse kinematics given the leg parameters, and once the servos are capable of accepting new commands, the servo angles are loaded along with servo rates. The foot position is checked against the maximum travel limit. If the travel limit has been reached, the state machine transitions to the Unload state. If the foot is operating within the Stance state working range, the foot pressure is sensed. The sensed foot pressure is checked to determine if sufficient pressure is applied. If the foot pressure is insufficient/too great, the foot is incremented/ decremented in the z-axis with rate dz. If the foot is extended too far in the z-axis, the state machine transitions to the Unload state. If the foot can be repositioned within its operating limits, the instruction pointer loops back to repeat the cycle until either travel limits are reached in the x- or z-axis.

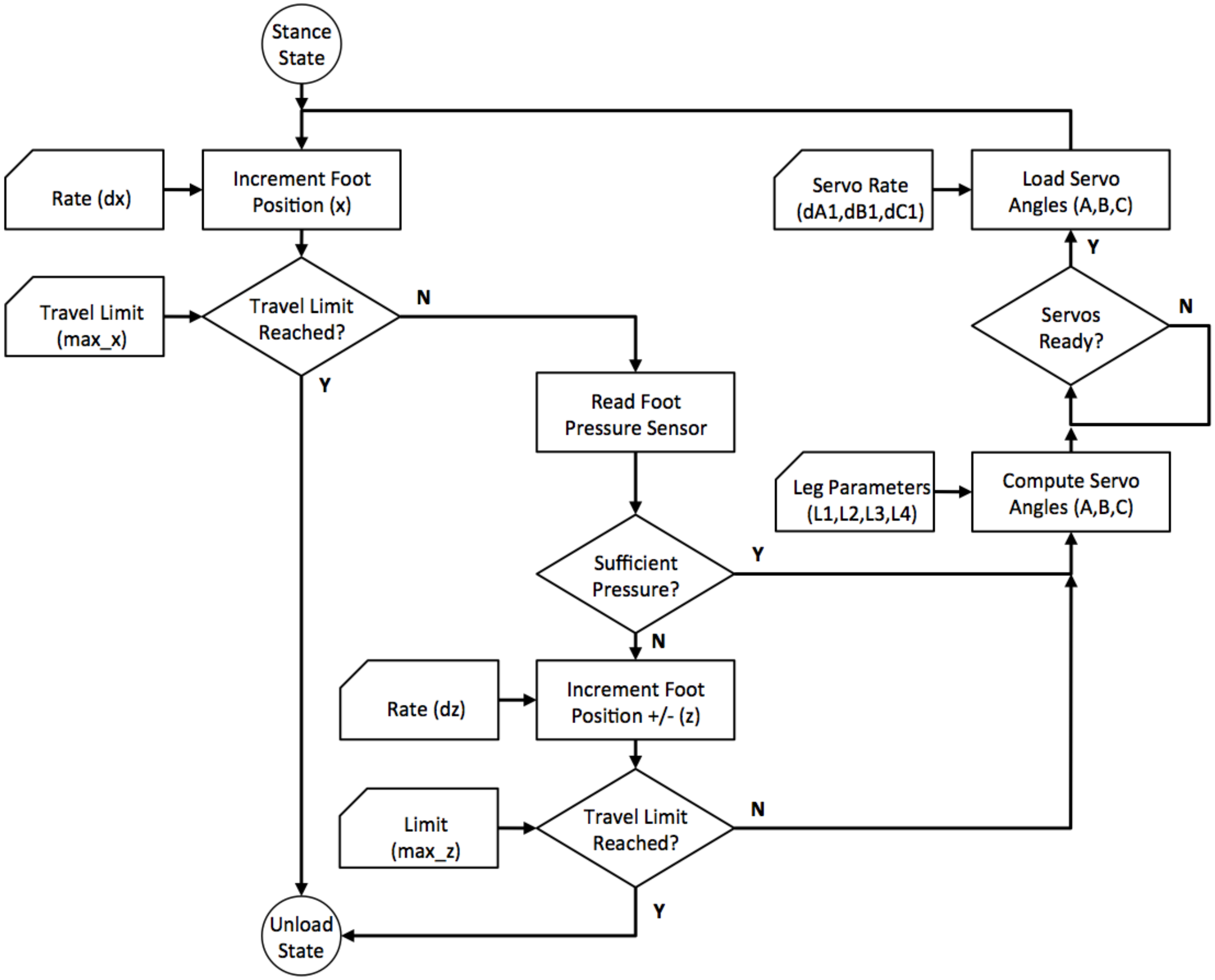
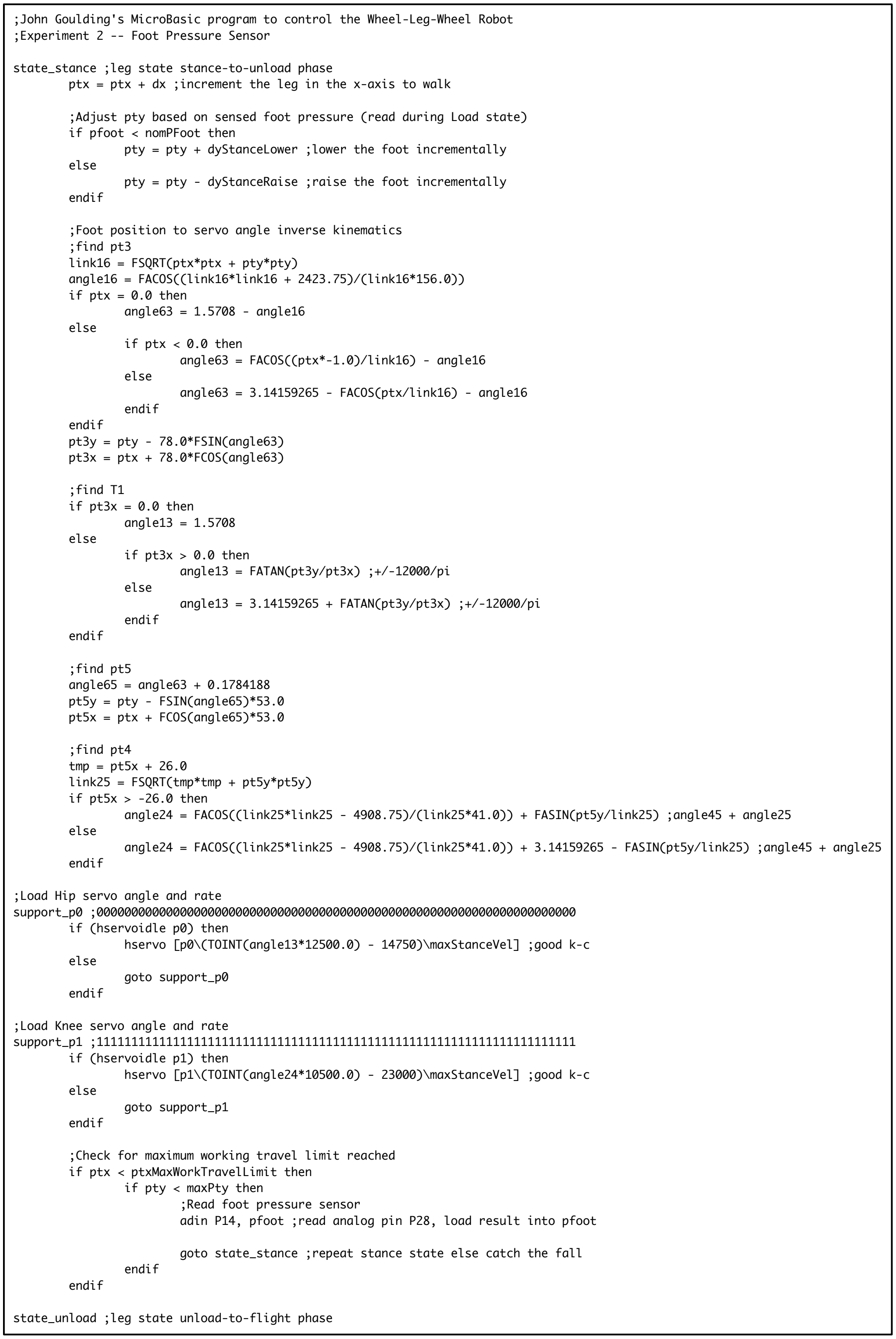


Figure B.17. Stance state with foot pressure sensor block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel Robot
;Experiment 2 -- Foot Pressure Sensor

state_stance ;leg state stance-to-unload phase
        ptx = ptx + dx ;increment the leg in the x-axis to walk

        ;Adjust pty based on sensed foot pressure (read during Load state)
        if pfoot < nomPFoot then
                pty = pty + dyStanceLower ;lower the foot incrementally
        else
                pty = pty - dyStanceRaise ;raise the foot incrementally
        endif

        ;Foot position to servo angle inverse kinematics
        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\maxStanceVel] ;good k-c
        else
                goto support_p0
        endif

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\maxStanceVel] ;good k-c
        else
                goto support_p1
        endif

        ;Check for maximum working travel limit reached
        if ptx < ptxMaxWorkTravelLimit then
                if pty < maxPty then
                        ;Read foot pressure sensor
                        adin P14, pfoot ;read analog pin P28, load result into pfoot

                        goto state_stance ;repeat stance state else catch the fall
                endif
        endif

state_unload ;leg state unload-to-flight phase
```

Figure B.18. Stance state with foot pressure sensor MicroBasic code.

In this experiment, the independent variables are 1) desired foot pressure, 2) z-axis distance to raise the foot, dz, per piece-wise step, dx, and 3) servo velocity. It is desired that the foot should exhibit nearly continuous motion to impart continuous robot motion. Thus, the servo rates and incremental distance the foot travels must be tuned such that the Stance state computations are completed at or slightly before the servos are ready to load the new position and rate commands. If the rates dx and dz are too large, the servos will stop and hesitate before the new commands can be loaded. A hesitation of the foot along the x-axis will likewise cause the robot foot to slip, breaking contact with the ground and resulting in a lower sliding coefficient of friction.

The following Figures B.19 and B.20 show recorded foot pressure for various combinations of desired foot pressure, dz, and servo velocity recorded over an entire Stance period, where the blue diamond, red square, and green triangle lines are different trials. The individual piece-wise steps may be identified as curve maximums and minimums, as dz generally oscillated between positive and negative values for successive piece-wise steps. Note that dx was tuned for each of the various combinations of servo velocities and dz step sizes, resulting in different numbers of piece-wise steps taken during each Stance trial. The following Table B.3 summarizes the foot pressure statistics (average, minimum, maximum, and standard deviation) recorded for various servo velocities and dz step sizes during a Stance period.

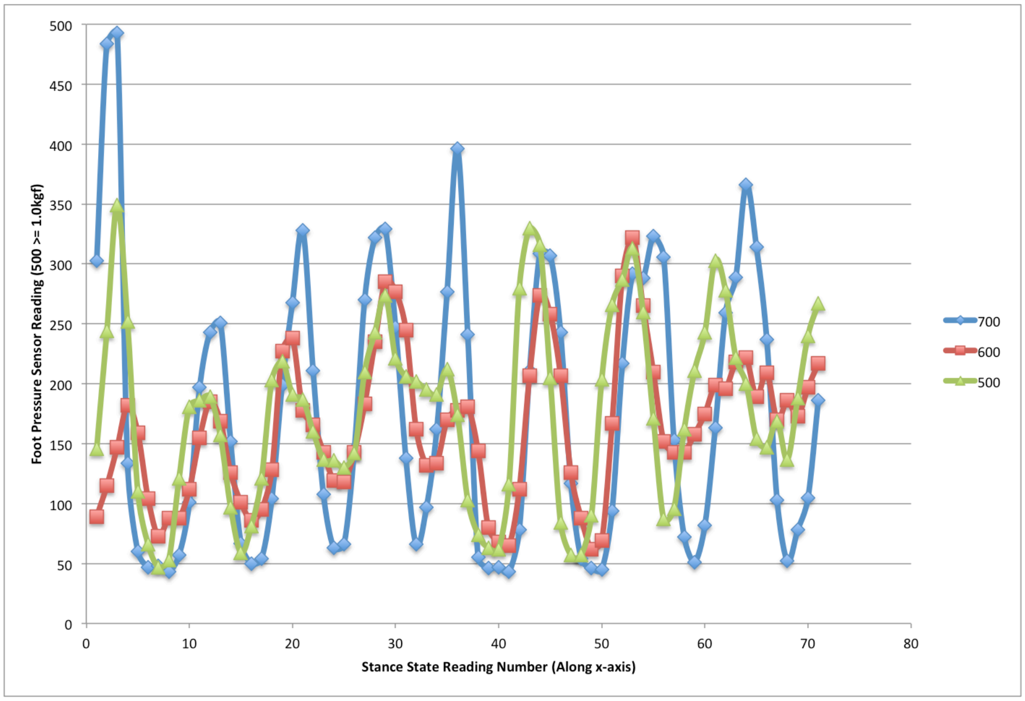


Figure B.19. Stance state foot pressure for constant dz and varying servo rate.

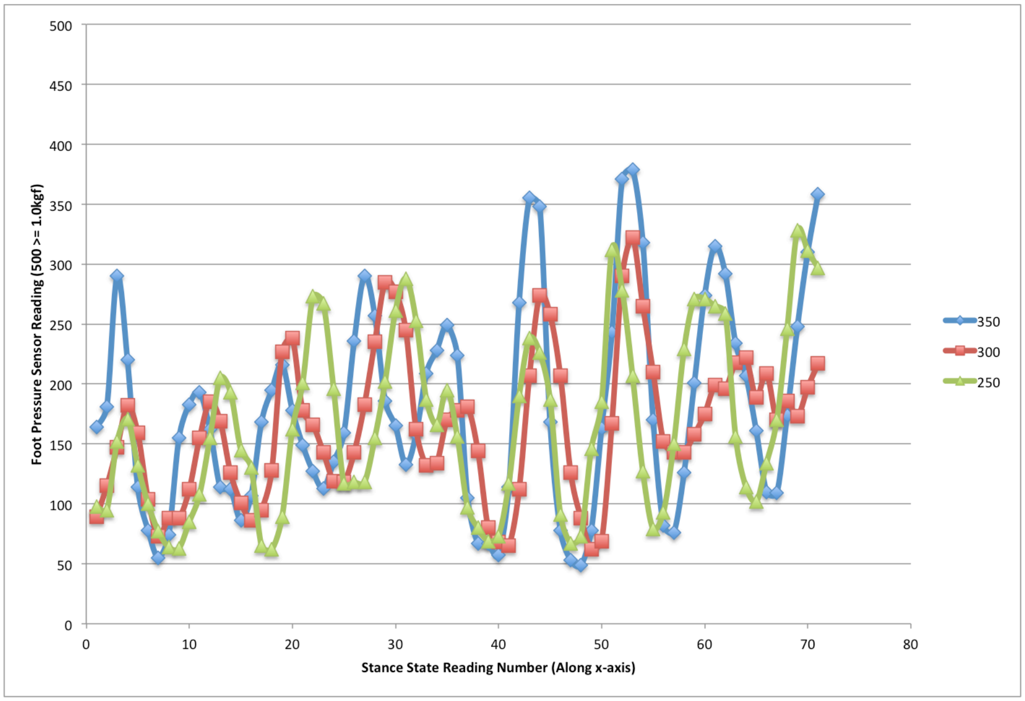


Figure B.20. Stance state foot pressure for constant servo rate and varying dz.

Table B.2. Foot pressure statistics recorded for various servo velocities and dz step sizes during a Stance period.

		Varying Servo Velocity (Step Size = 300)				Varying dz Step Size (Servo Velocity = 600)		
		700	600	500		350	300	250
Foot Pressure	AVG	175.4	164.5	176.5	AVG	180.2	164.5	165.4
	MIN	43.0	62.0	47.0	MIN	49.0	62.0	62.0
	MAX	493.0	322.0	349.0	MAX	379.0	322.0	328.0
	STD	116.7	60.9	76.7	STD	86.8	60.9	74.7

In Figure B.19, a constant servo rate of 700 (out of 1,200 maximum servo units) with a dz step size of 300 (servo units) and a desired foot pressure of 175 (out of 500 maximum ADC units), results in a maximum foot pressure of 493 (ADC units) recorded on the first step. As the servo velocity is reduced (and the system re-tuned), the maximum first step foot pressure is reduced. However, an oscillation in foot pressure is observed to increase mid-stride as servo velocity is reduced. Recall the objective of walking is to move the foot with constant velocity (constant dx per unit time) in the x-axis. Due to the kinematics of the Hirose leg design, the servo rates must be non-linear over the stride length to maintain a constant velocity in the x-axis.

In Figure B.20, a dz step size of 300 (servo units) with a constant servo velocity of 300 (servo units) and a desired foot pressure of 175 (ADC units), results in a maximum foot pressure of 322 (ADC units) recorded on the first step, and a foot pressure standard deviation of 60.9 (ADC units). As the dz step size is varied (and the system re-tuned), the foot pressure standard deviation increases. An oscillation in foot pressure is observed to increase mid-stride as servo velocity is reduced or increased.

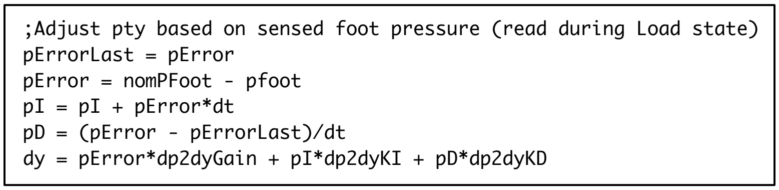
From the above, two modifications to the Stance algorithm are indicated to optimally drive the Hirose leg. First, the servo rate should be proportionally adjusted based on the piece-wise incremental servo angle computed using the inverse kinematic equations, given the (constant) dx rate of the desired trajectory. Figure B.21 shows the Stance state code addition to proportionally adjust the servo velocity. Note that loading the Knee servo was also changed to load first, as it undergoes the largest angular change during Stance and is thus more likely to be busy. To achieve a smooth trajectory, it is important to load the servos near-simultaneously.

```
        ;proportionally adjust servo velocity
        p0Vel = maxStanceVel + TOINT(((angle13*velScale) - velBias)*velGain)
        p1Vel = maxStanceVel + TOINT(((angle24*velScale) - velBias)*velGain)

;Load Knee servo angle and rate
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\(TOINT(angle24*10500.0) - 23000)\p1Vel] ;good k-c
        else
                goto support_p1
        endif

;Load Hip servo angle and rate
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\(TOINT(angle13*12500.0) - 14750)\p0Vel] ;good k-c
        else
                goto support_p0
        endif
```

Figure B.21. Stance state code to proportionally adjust the servo velocity.

Second, a PID algorithm should be used to adjust the dz step size to reduce the oscillation about the desired foot pressure. Figure B.22 shows the PID algorithm, replacing the previous If-Then-Else algorithm shown in Figure B.18. The gain, then KI and then KD parameters were iteratively tuned, and the resulting foot pressure over the

Stance state was measured as shown in Figure B.23. The best foot pressure averaged 174.6 (ADC units), with a peak of 199.4 (ADC units), a minimum of 156.1 (ADC units), and a standard deviation of 8.6 (ADC units).



```
;Adjust pty based on sensed foot pressure (read during Load state)
pErrorLast = pError
pError = nomPFoot - pfoot
pI = pI + pError*dt
pD = (pError - pErrorLast)/dt
dy = pError*dp2dyGain + pI*dp2dyKI + pD*dp2dyKD
```

Figure B.22. Stance state PID algorithm to adjust dz step height.

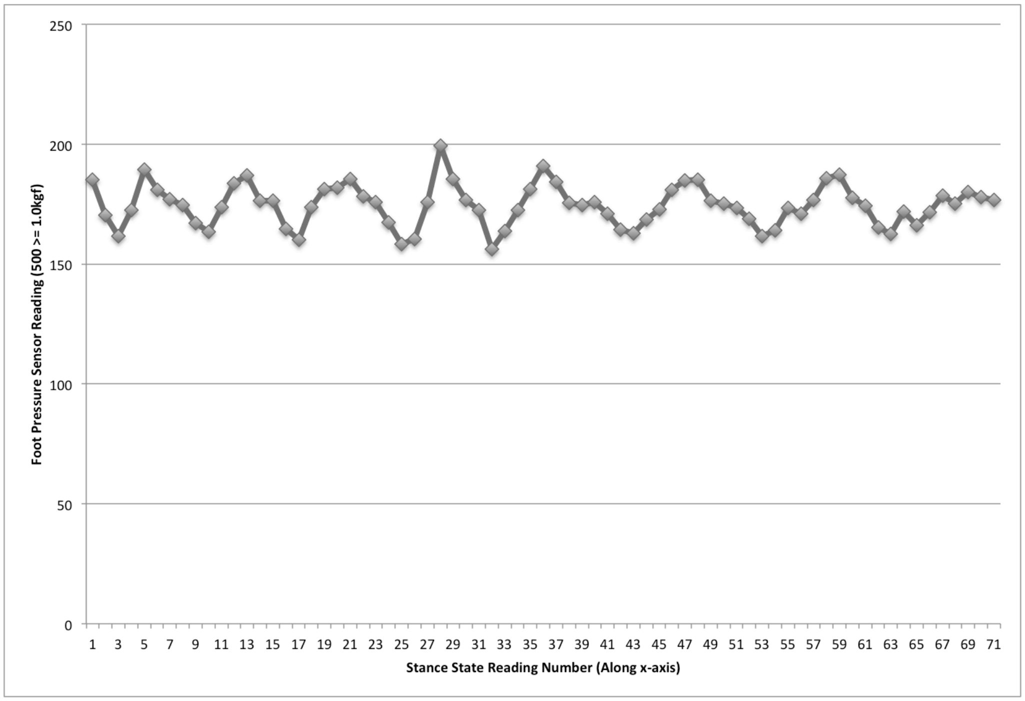


Figure B.23. Stance state foot pressure with dz step size PID algorithm and proportionally adjusted servo velocity.

Appendix C continues the experimentation with the Draisine robot upright. Experiment 3 adds accelerometer and gyroscope feedback to control the theta stage and outward swing of the leg to catch a fall. Experiment 4 implements all of the algorithms from the previous experiments to propel and balance the Draisine robot in a straight-line trajectory. Experiment 5 tests the Draisine robot in a curved trajectory, demonstrating the leaning-into-the-turn maneuver.

APPENDIX C

DRAISINE ROBOT EXPERIMENTAL RESULTS

Appendix B discusses and implements the results of testing the coupled-drive leg, described in Chapter 8. The front and rear legs of the Single Track Three Legged Mobile Robot (ST3LMR) are replaced with wheels, allowing it to be modeled as a simple inverted pendulum. The robot is now called a Draisine robot. Like the early Draisine, a pedal-less precursor to the bicycle, the robot leg is used to both propel the robot and prevent it from falling over while executing both straight and curved single-track trajectories. It controls accelerations in the roll axis by dynamically shifting the foot to the left or right of the center of gravity to move the line of support.

This appendix presents three experiments of increasing capability. Experiment 3 adds accelerometer and gyroscope feedback to control the theta stage and outward swing of the leg to catch a fall. The foot is repositioned during the Flight state to both support and re-start the leg cycle. Repositioning the foot involves calculating the roll velocity and acceleration, computing the neutral position using Raibert’s equations, and commanding the servos. Experiment 4 implements all of the algorithms from the previous experiments to propel and balance the Draisine robot in a straight trajectory. Experiment 5 tests the Draisine robot in a curved trajectory, demonstrating the leaning-into-the-turn maneuver. This chapter concludes with a summary of the control method for the ST3LMR.

C.1. Experiment 3 – Sensing and Controlling Roll

There are many different sensing methods to determine upright orientation and maintain balance: gravity-based sensors (electrolytic fluid U-tube, oil-damped pendulum, etc.), triangulation by direct measurement, echolocation (laser time of flight, radar, etc.), reference systems (the Sun, landmarks, GPS etc.), proprioception or self-posture, inertial measurement units (IMUs), and digital image processing to list but a few. In this experiment we use a microelectromechanical system or MEMS gyroscope and accelerometer to measure the roll angle and role acceleration. Found in games, cell phones, cameras, model airplanes and helicopters, and other consumer and industrial products, the MEMS gyro is a low-cost, reliable sensing device. It typically provides very high sampling rates and fast response times, but its output contains high non-Gaussian noise and time-varying bias and/or scale factor (e.g., from temperature and vibration). The voltage output of the MEMS gyro is modeled as follows:

$$V_g = V_o + g\dot{\theta} \tag{C.1}$$

where V_g is the gyro sensor output voltage, V_o is a constant bias voltage, g is sensitivity, and $\theta^{\cdot}$ is the angular velocity. The angular velocity is integrated to provide the rotation angle:

$$\theta = \theta_o + AV_g dt \tag{C.2}$$

where θ is the angle, θ_o is the angle from the last measurement update time, A is a voltage to angle conversion factor (which may not be necessary for small angles), and *dt* is the time between measurements.

The main drawback to MEMS gyro sensing is sensor drift (Imamura 2008; and Lee 2009). Tilt sensors and accelerometers have been used to compensate drift (Imamura 2008, Lee 2009, and Rehbinder 2000), but tilt sensors have slow response times and both are affected by translational acceleration (Rehbinder 2004). So, a complimentary filter is typically used to combine MEMS gyro and accelerometer data, especially when using low-throughput micro-controllers, like the BasicATOM Pro 40m used in these experiments. The complementary filter takes the form:

$$\theta = B(\theta_o + AV_g dt) + D(CV_a) \tag{C.3}$$

where V_a is the voltage output from the accelerometer, C is a voltage to angle conversion factor (which may not be necessary for small angles), and B and D are weights that sum to 1.0, such as (0.9) and (0.1), to assign the relative importance to gyro and accelerometer data. A large B and small D will respond quickly to the gyro and lag the accelerometer, while a large D and small B will favor the accelerometer, which may be noisy. Note that Kalman filters, such as those described in Chapter 6, are generally not used in microcontrollers, due to the additional complexity of the calculations. Rather, it is more beneficial to have a high sampling rate.

The first two experiments in Appendix C developed a State machine to execute the basic leg cycle, selectively performing a stance-to-flight phase, a flight phase, a flight-to-stance phase, and a stance phase. In experiment 3, the theta stage is turned on and the z-axis (leg extension and retraction) is allowed, but leg x-axis motion (along the major length of the body and forward direction of travel) is disabled (dx = 0). The

Draisine robot rotates about the axis developed by the wheels contacting the ground, allowing it to be modeled as a simple inverted pendulum.

Raibert modeled one-legged hopping robots as an inverted pendulum systems and decomposed control into three separate elements: 1) supporting the body by controlling the vertical hopping height, 2) positioning the feet in key locations on each step using symmetry principles to maintain balance (Equation C.4), and 3) controlling the body attitude by controlling hip torque and angle during the stance phase (Equation C.5), such that the dynamic momentum of the body is estimated ahead in time to calculate the future foot placement and thrust needed to develop complementary dynamic momentum and achieve a desired hopping height, running velocity, and body attitude (Raibert 1986). See Chapter 5 for further background information and details.

$$y_f = \frac{\dot{y}T_S}{2} + k_{\dot{y}}(\dot{y} - \dot{y}_d) \qquad \text{(C.4)}$$

$$\gamma_d = \phi - arcsin\left({}^{y_f}/_{L}\right) \qquad \text{(C.5)}$$

Where y_f is the position of the foot, $\dot{y}$ is the lateral speed, T_s is the duration of the stance phase, $k_{\dot{y}}$ is a feedback gain, $\dot{y}_d$ is the desired lateral speed, γ_d is the angle between the leg and the body, Φ is also the pitch angle of the frame, and L is the expected length of the leg during stance.

The concept of a goal position or goal roll angle is now introduced. Because it is desired to keep the Draisine robot in an upright position, the goal role angle is 0° or vertical. Following Raibert’s work, deviation to the left or right of the goal angle may be corrected by thrusting the foot outward radially in the direction of the fall and incrementing the foot in the z-axis to catch the fall. Once the foot contacts the ground, lengthening the leg up rights the Draisine robot to the goal roll angle.

In this experiment, the Unload state is the same as previously described in Experiment 1, Appendix C, Figures C-2 and C-3, but with a change of leg retraction, dz, such that the foot is lifted higher to clear the expected worst case roll angle. The Flight state, repositions the foot outward to catch the fall. Figures C.2 and C.3 modify the Flight state block diagram and code developed in Experiment 1, Appendix C, Figures C-4 and C-5. Referring to Figure C.2, the accelerometer and gyroscope sensors are sampled, the roll velocity and acceleration are determined using a complimentary filter, the theta stage angle is calculated using a temporal estimation of when the foot will contact the ground, and leg extension, dz, is calculated based on the theta stage angle. Given the new foot position, the corresponding hip and knee servo angles are computed using inverse kinematics. Note the Hirose leg arcs the foot upward and away from the ground when the hip servo is commanded faster than the knee servo; so, only a single servo command is needed. Generally, the hip servo rate is the maximum rate possible. When the servos can accept new commands, the new servo angles are loaded along with the desired server rate. Loading the servos completes the Flight state, and control is transitioned to the Load state.

The Load state incrementally lowers the foot to the expected ground position to develop reaction forces between the foot and the ground. Figures C.4 and C.5 modify the Load state block diagram and code developed in Experiment 1, Appendix C, Figures C-6 and C-7. Here, the roll velocity may be used to sense an out-of-control situation, forcing an Unload state, but this feature is not implemented in this experiment. Loading the servos completes the Load state, wherein control is transitioned to the Stance state.

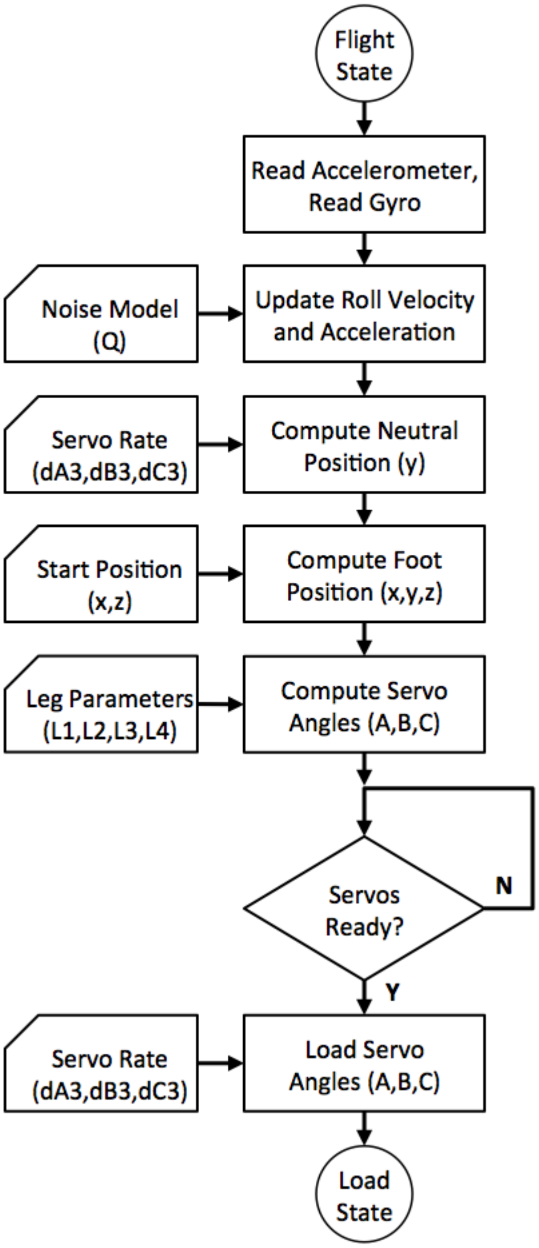
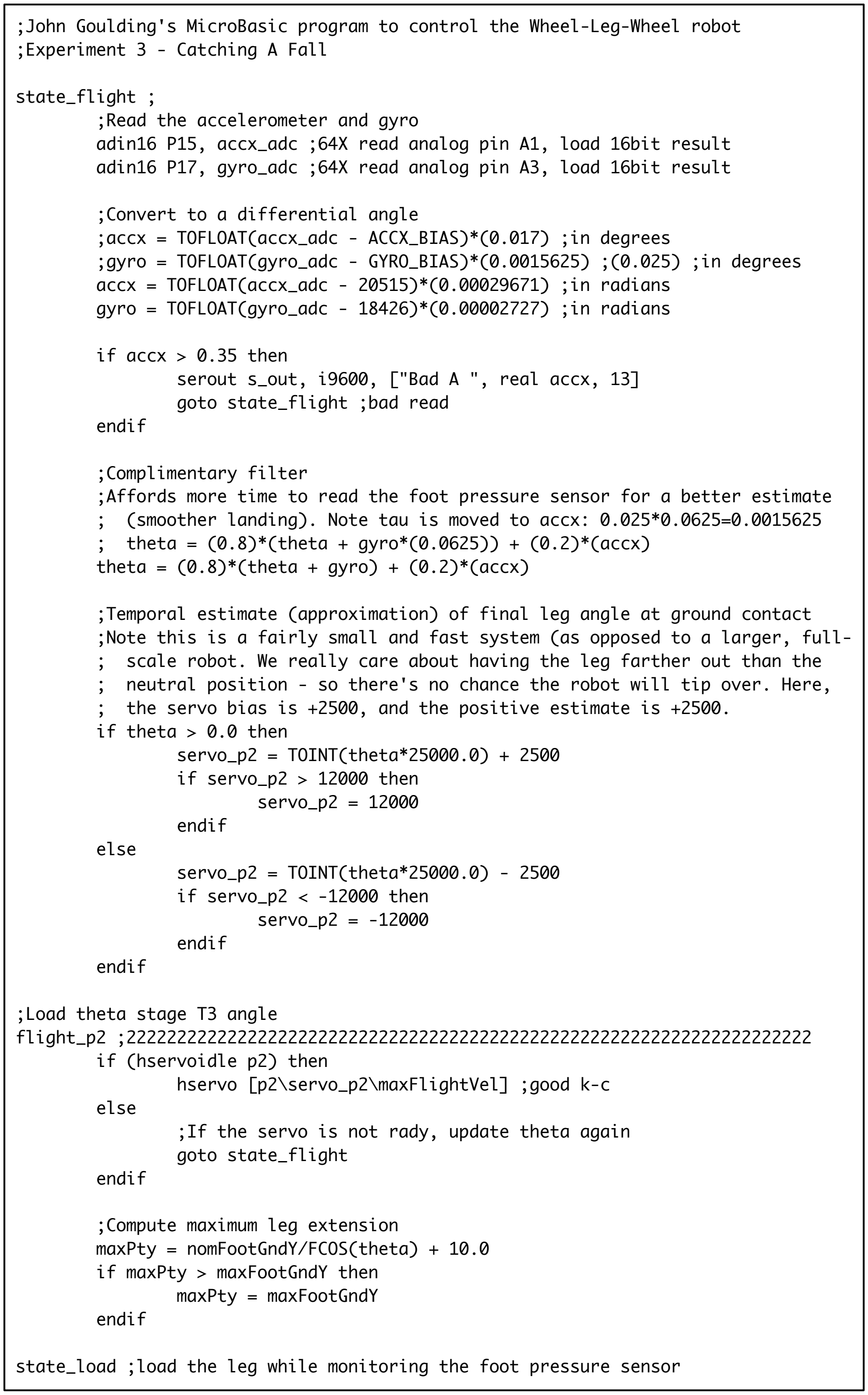

Flight State
Read Accelerometer, Read Gyro
Noise Model (Q)
Update Roll Velocity and Acceleration
Servo Rate (dA3,dB3,dC3)
Compute Neutral Position (y)
Start Position (x,z)
Compute Foot Position (x,y,z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servos Ready?
N
Y
Servo Rate (dA3,dB3,dC3)
Load Servo Angles (A,B,C)
Load State

Figure C.1. Flight state with IMU sensor block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel robot
;Experiment 3 - Catching A Fall

state_flight ;
        ;Read the accelerometer and gyro
        adin16 P15, accx_adc ;64X read analog pin A1, load 16bit result
        adin16 P17, gyro_adc ;64X read analog pin A3, load 16bit result

        ;Convert to a differential angle
        ;accx = TOFLOAT(accx_adc - ACCX_BIAS)*(0.017) ;in degrees
        ;gyro = TOFLOAT(gyro_adc - GYRO_BIAS)*(0.0015625) ;(0.025) ;in degrees
        accx = TOFLOAT(accx_adc - 20515)*(0.00029671) ;in radians
        gyro = TOFLOAT(gyro_adc - 18426)*(0.00002727) ;in radians

        if accx > 0.35 then
                serout s_out, i9600, ["Bad A ", real accx, 13]
                goto state_flight ;bad read
        endif

        ;Complimentary filter
        ;Affords more time to read the foot pressure sensor for a better estimate
        ;  (smoother landing). Note tau is moved to accx: 0.025*0.0625=0.0015625
        ;  theta = (0.8)*(theta + gyro*(0.0625)) + (0.2)*(accx)
        theta = (0.8)*(theta + gyro) + (0.2)*(accx)

        ;Temporal estimate (approximation) of final leg angle at ground contact
        ;Note this is a fairly small and fast system (as opposed to a larger, full-
        ;  scale robot. We really care about having the leg farther out than the
        ;  neutral position - so there's no chance the robot will tip over. Here,
        ;  the servo bias is +2500, and the positive estimate is +2500.
        if theta > 0.0 then
                servo_p2 = TOINT(theta*25000.0) + 2500
                if servo_p2 > 12000 then
                        servo_p2 = 12000
                endif
        else
                servo_p2 = TOINT(theta*25000.0) - 2500
                if servo_p2 < -12000 then
                        servo_p2 = -12000
                endif
        endif

;Load theta stage T3 angle
flight_p2 ;22222222222222222222222222222222222222222222222222222222222222222222
        if (hservoidle p2) then
                hservo [p2\servo_p2\maxFlightVel] ;good k-c
        else
                ;If the servo is not rady, update theta again
                goto state_flight
        endif

        ;Compute maximum leg extension
        maxPty = nomFootGndY/FCOS(theta) + 10.0
        if maxPty > maxFootGndY then
                maxPty = maxFootGndY
        endif

state_load ;load the leg while monitoring the foot pressure sensor
```

Figure C.2. Flight state MicroBasic code.

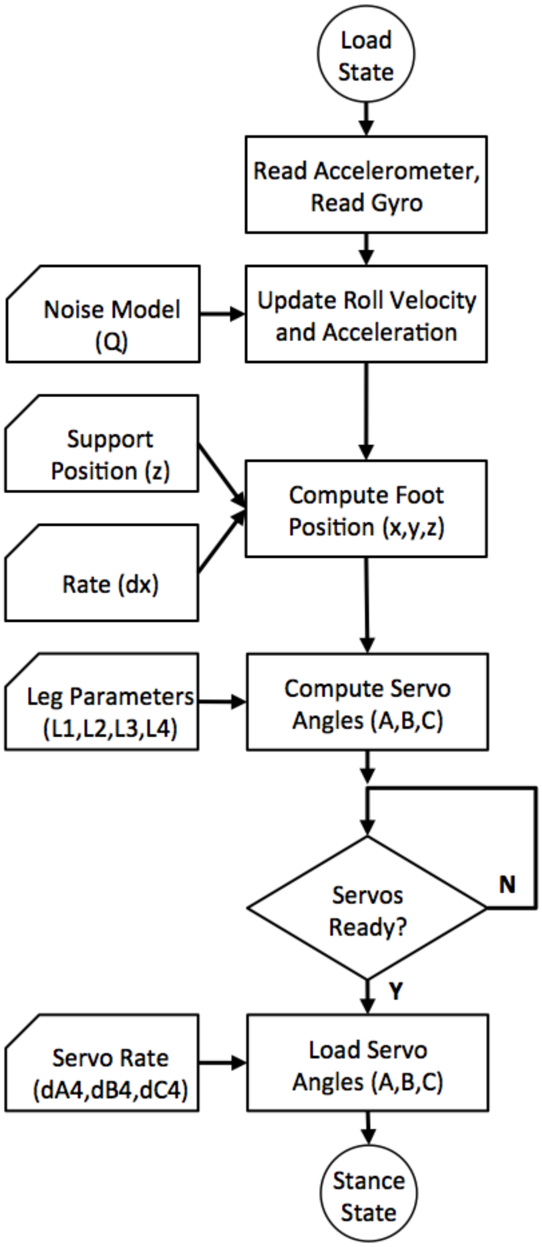
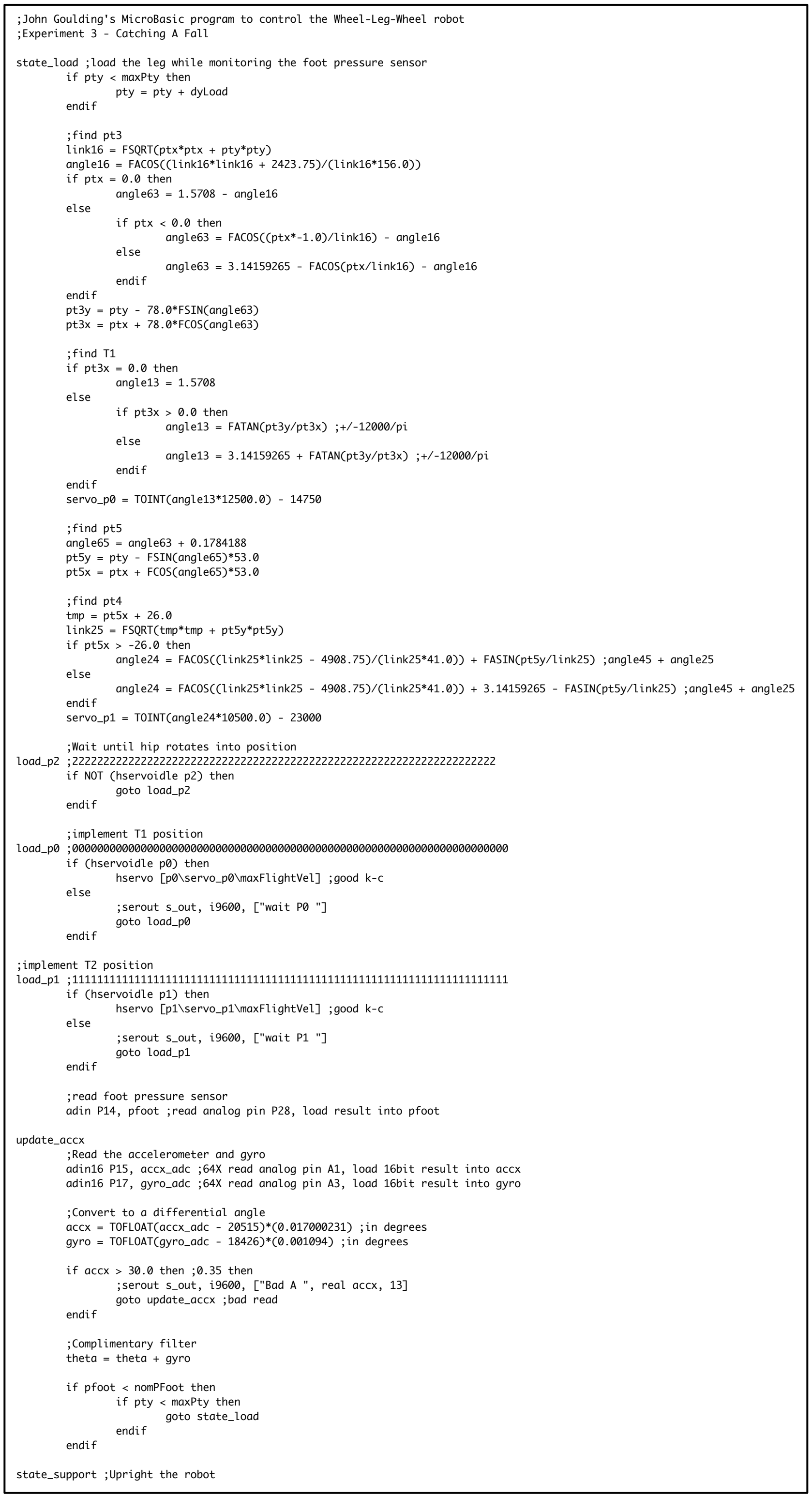

Load State
Read Accelerometer, Read Gyro
Noise Model (Q)
Update Roll Velocity and Acceleration
Support Position (z)
Rate (dx)
Compute Foot Position (x,y,z)
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
Servos Ready?
N
Y
Servo Rate (dA4,dB4,dC4)
Load Servo Angles (A,B,C)
Stance State

Figure C.3. Load state block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel robot
;Experiment 3 - Catching A Fall

state_load ;load the leg while monitoring the foot pressure sensor
        if pty < maxPty then
                pty = pty + dyLoad
        endif

        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif
        servo_p0 = TOINT(angle13*12500.0) - 14750

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif
        servo_p1 = TOINT(angle24*10500.0) - 23000

        ;Wait until hip rotates into position
load_p2 ;22222222222222222222222222222222222222222222222222222222222222222222
        if NOT (hservoidle p2) then
                goto load_p2
        endif

        ;implement T1 position
load_p0 ;0000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\servo_p0\maxFlightVel] ;good k-c
        else
                ;serout s_out, i9600, ["wait P0 "]
                goto load_p0
        endif

;implement T2 position
load_p1 ;1111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\servo_p1\maxFlightVel] ;good k-c
        else
                ;serout s_out, i9600, ["wait P1 "]
                goto load_p1
        endif

        ;read foot pressure sensor
        adin P14, pfoot ;read analog pin P28, load result into pfoot

update_accx
        ;Read the accelerometer and gyro
        adin16 P15, accx_adc ;64X read analog pin A1, load 16bit result into accx
        adin16 P17, gyro_adc ;64X read analog pin A3, load 16bit result into gyro

        ;Convert to a differential angle
        accx = TOFLOAT(accx_adc - 20515)*(0.017000231) ;in degrees
        gyro = TOFLOAT(gyro_adc - 18426)*(0.001094) ;in degrees

        if accx > 30.0 then ;0.35 then
                ;serout s_out, i9600, ["Bad A ", real accx, 13]
                goto update_accx ;bad read
        endif

        ;Complimentary filter
        theta = theta + gyro

        if pfoot < nomPFoot then
                if pty < maxPty then
                        goto state_load
                endif
        endif

state_support ;Upright the robot
```

Figure C.4. Load state MicroBasic code.

In this experiment, the Stance state, involves up righting the Draisine robot. Figures C.5 and C.6 are the Stance state block diagram and code. Foot position is held constant in the x- and y-axis (theta stage) while extending the leg in the z-axis until the desired roll angle or travel limit is reached, then the Stance state transitions to the Unload state, and “catching a fall” repeats.

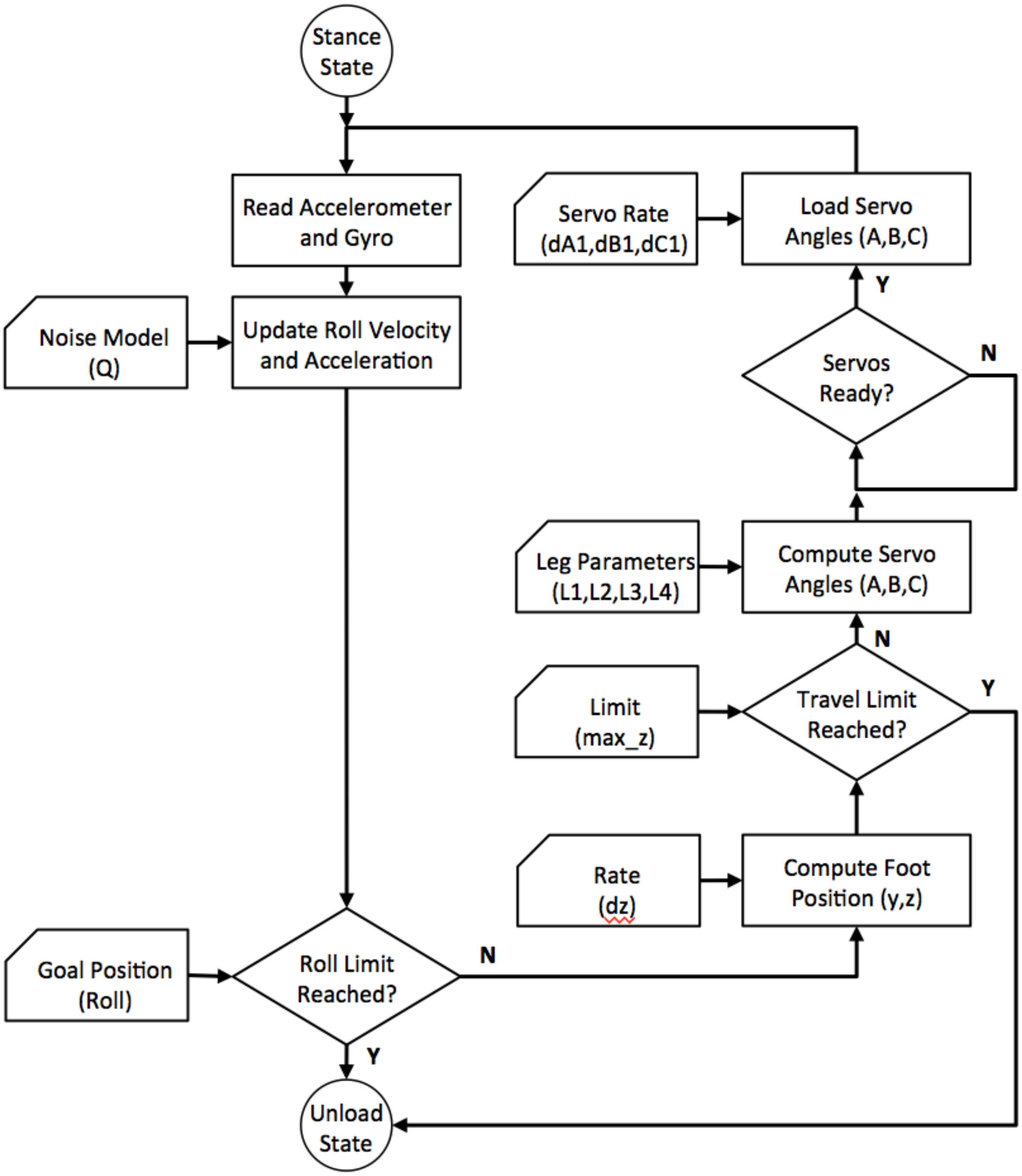
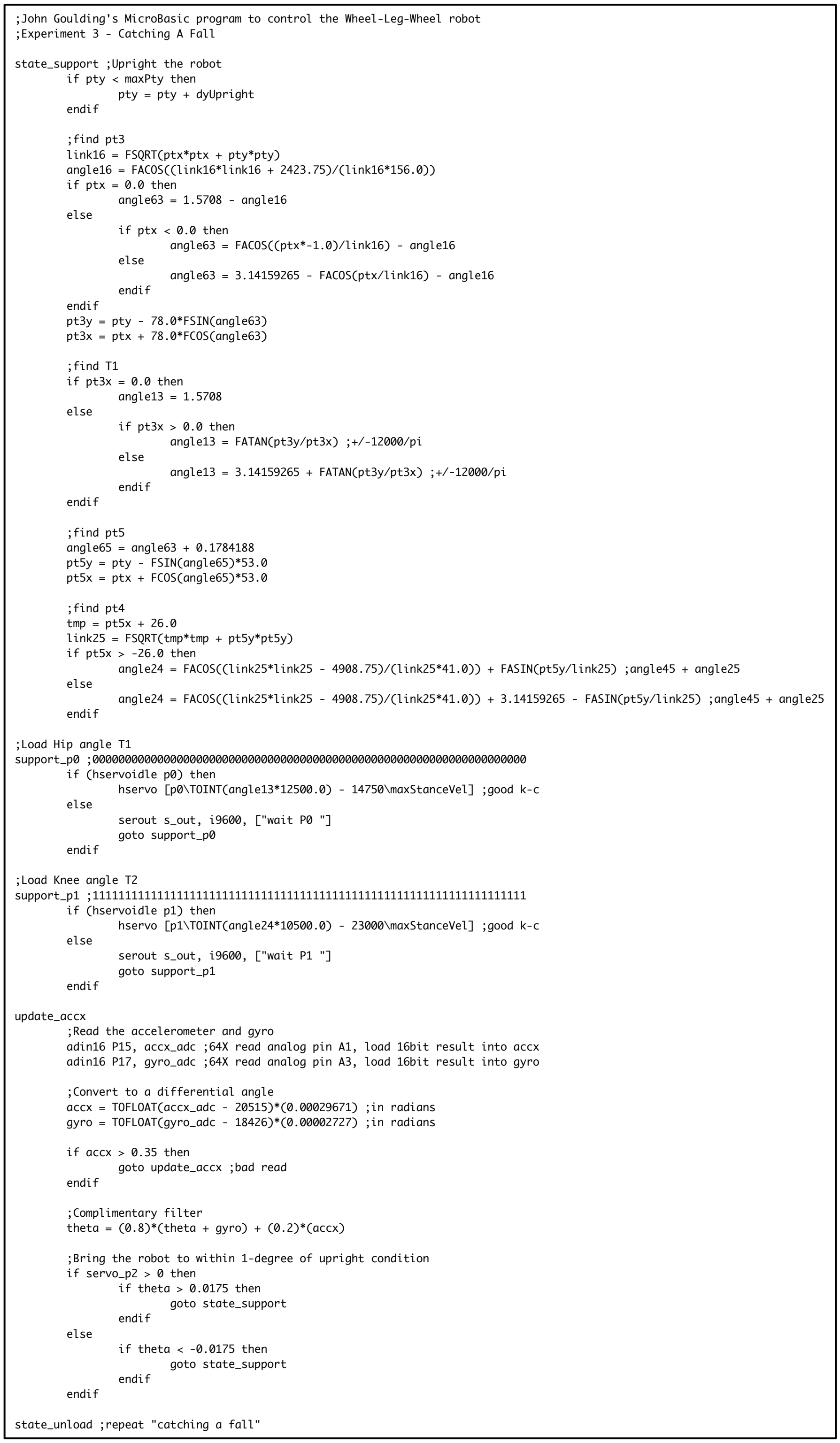

Stance State
Read Accelerometer and Gyro
Servo Rate (dA1,dB1,dC1)
Load Servo Angles (A,B,C)
Noise Model (Q)
Update Roll Velocity and Acceleration
Y
Servos Ready?
N
Leg Parameters (L1,L2,L3,L4)
Compute Servo Angles (A,B,C)
N
Limit (max_z)
Travel Limit Reached?
Y
Rate (dz)
Compute Foot Position (y,z)
Goal Position (Roll)
Roll Limit Reached?
N
Y
Unload State

Figure C.5. Stance state with IMU sensor block diagram.



```
;John Goulding's MicroBasic program to control the Wheel-Leg-Wheel robot
;Experiment 3 - Catching A Fall

state_support ;Upright the robot
        if pty < maxPty then
                pty = pty + dyUpright
        endif

        ;find pt3
        link16 = FSQRT(ptx*ptx + pty*pty)
        angle16 = FACOS((link16*link16 + 2423.75)/(link16*156.0))
        if ptx = 0.0 then
                angle63 = 1.5708 - angle16
        else
                if ptx < 0.0 then
                        angle63 = FACOS((ptx*-1.0)/link16) - angle16
                else
                        angle63 = 3.14159265 - FACOS(ptx/link16) - angle16
                endif
        endif
        pt3y = pty - 78.0*FSIN(angle63)
        pt3x = ptx + 78.0*FCOS(angle63)

        ;find T1
        if pt3x = 0.0 then
                angle13 = 1.5708
        else
                if pt3x > 0.0 then
                        angle13 = FATAN(pt3y/pt3x) ;+/-12000/pi
                else
                        angle13 = 3.14159265 + FATAN(pt3y/pt3x) ;+/-12000/pi
                endif
        endif

        ;find pt5
        angle65 = angle63 + 0.1784188
        pt5y = pty - FSIN(angle65)*53.0
        pt5x = ptx + FCOS(angle65)*53.0

        ;find pt4
        tmp = pt5x + 26.0
        link25 = FSQRT(tmp*tmp + pt5y*pt5y)
        if pt5x > -26.0 then
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + FASIN(pt5y/link25) ;angle45 + angle25
        else
                angle24 = FACOS((link25*link25 - 4908.75)/(link25*41.0)) + 3.14159265 - FASIN(pt5y/link25) ;angle45 + angle25
        endif

;Load Hip angle T1
support_p0 ;000000000000000000000000000000000000000000000000000000000000000000000
        if (hservoidle p0) then
                hservo [p0\TOINT(angle13*12500.0) - 14750\maxStanceVel] ;good k-c
        else
                serout s_out, i9600, ["wait P0 "]
                goto support_p0
        endif

;Load Knee angle T2
support_p1 ;111111111111111111111111111111111111111111111111111111111111111111111
        if (hservoidle p1) then
                hservo [p1\TOINT(angle24*10500.0) - 23000\maxStanceVel] ;good k-c
        else
                serout s_out, i9600, ["wait P1 "]
                goto support_p1
        endif

update_accx
        ;Read the accelerometer and gyro
        adin16 P15, accx_adc ;64X read analog pin A1, load 16bit result into accx
        adin16 P17, gyro_adc ;64X read analog pin A3, load 16bit result into gyro

        ;Convert to a differential angle
        accx = TOFLOAT(accx_adc - 20515)*(0.00029671) ;in radians
        gyro = TOFLOAT(gyro_adc - 18426)*(0.00002727) ;in radians

        if accx > 0.35 then
                goto update_accx ;bad read
        endif

        ;Complimentary filter
        theta = (0.8)*(theta + gyro) + (0.2)*(accx)

        ;Bring the robot to within 1-degree of upright condition
        if servo_p2 > 0 then
                if theta > 0.0175 then
                        goto state_support
                endif
        else
                if theta < -0.0175 then
                        goto state_support
                endif
        endif

state_unload ;repeat "catching a fall"
```

Figure C.6. Stance state MicroBasic code.

Figure C.7 plots four cycles of hand-actuated accelerometer, gyro, and theta complementary filter. Hand actuated means the Draisine robot was rolled from left to right by hand, for the purpose of gathering data to tune the complementary filter. In the plot, the blue short-dashed line is the accelerometer output in degrees, the red long-dashed line is the gyroscope output scaled by 7x for clarity, and the solid black line is the roll angle from the complementary filter. Referring to the complementary filter equation 12-3, dt = 0.0625 (seconds), A = 0.000469 (radians/ADC), B = 0.000297 (radians/ADC), C = 0.8, and D = 0.2. By noting the zero-crossing of the gyroscope as the roll transition, it is observed that theta lags the response of both the gyroscope and the accelerometer.

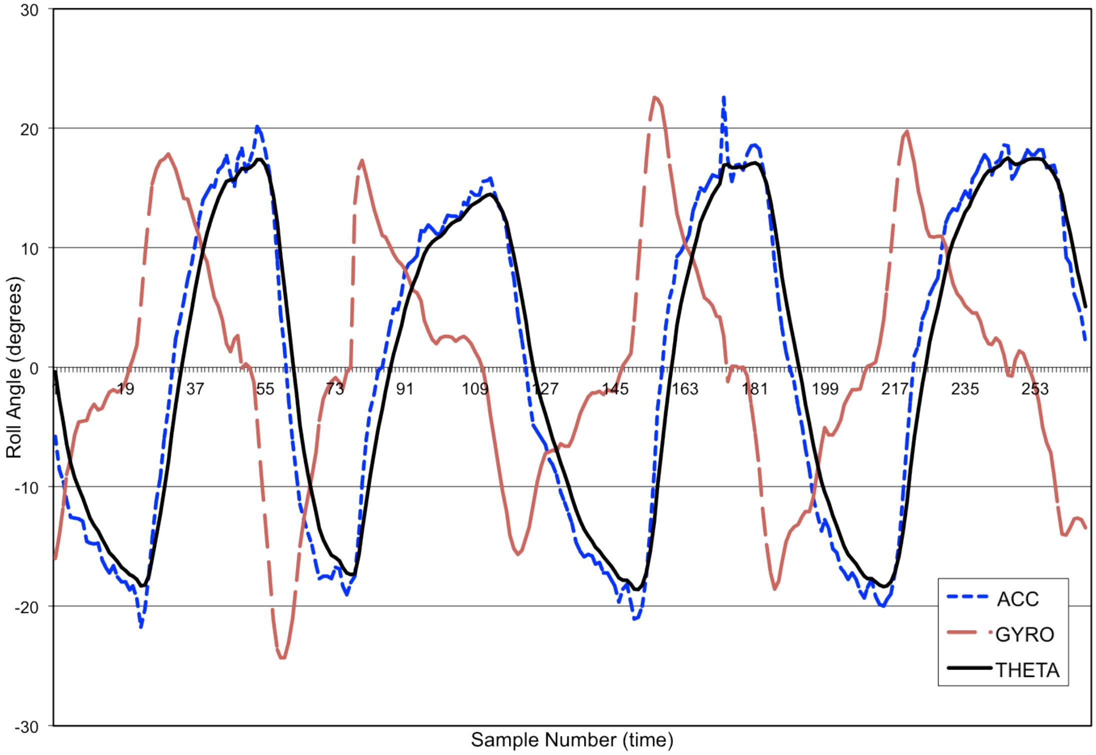


Figure C.7. Hand actuated accelerometer, gyro, and complementary filter (theta).

Figure C.8 plots two cycles of free fall accelerometer, gyro, and complementary filter (theta). Free fall means the Draisine robot was up righted and allowed to roll to left or right under gravity. The outriggers were adjusted to stop the roll at ±20-degrees. In the plot, the blue short-dashed line is the accelerometer output in degrees, the red long-dashed line is the gyroscope output scaled by 7x for clarity, and the solid black line is the roll angle from the complementary filter. Referring to the complementary filter equation 12-3, dt = 0.0625 (seconds), A = 0.001563 (radians/ADC), B = 0.000297 (radians/ADC), C = 0.8, and D = 0.2. To reduce the effect of accelerometer oscillation when the outriggers stop the fall, the gyroscope gain was increased by 33x. This also has the desired effect of improving the complementary filter response, as theta can be seen to be mostly leading the accelerometer curve in the plot.

Finally, Figure C.9 plots five instances of the robot catching its fall, wherein the upper plot the blue short-dashed line is the accelerometer output in degrees, the red long-dashed line is the gyroscope output scaled by 7x for clarity, and the solid black line is the roll angle from the complementary filter, and the lower plot shows the commanded z-axis foot position.

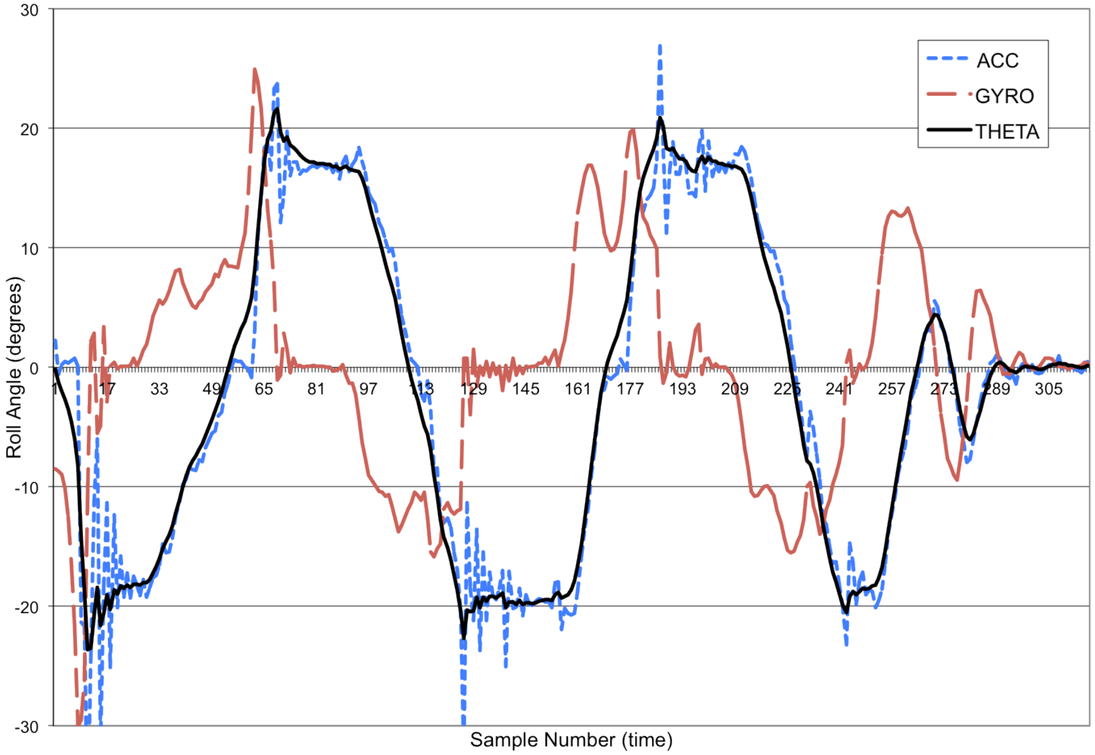


Figure C.8. Free-fall accelerometer, gyro, and complementary filter (theta).

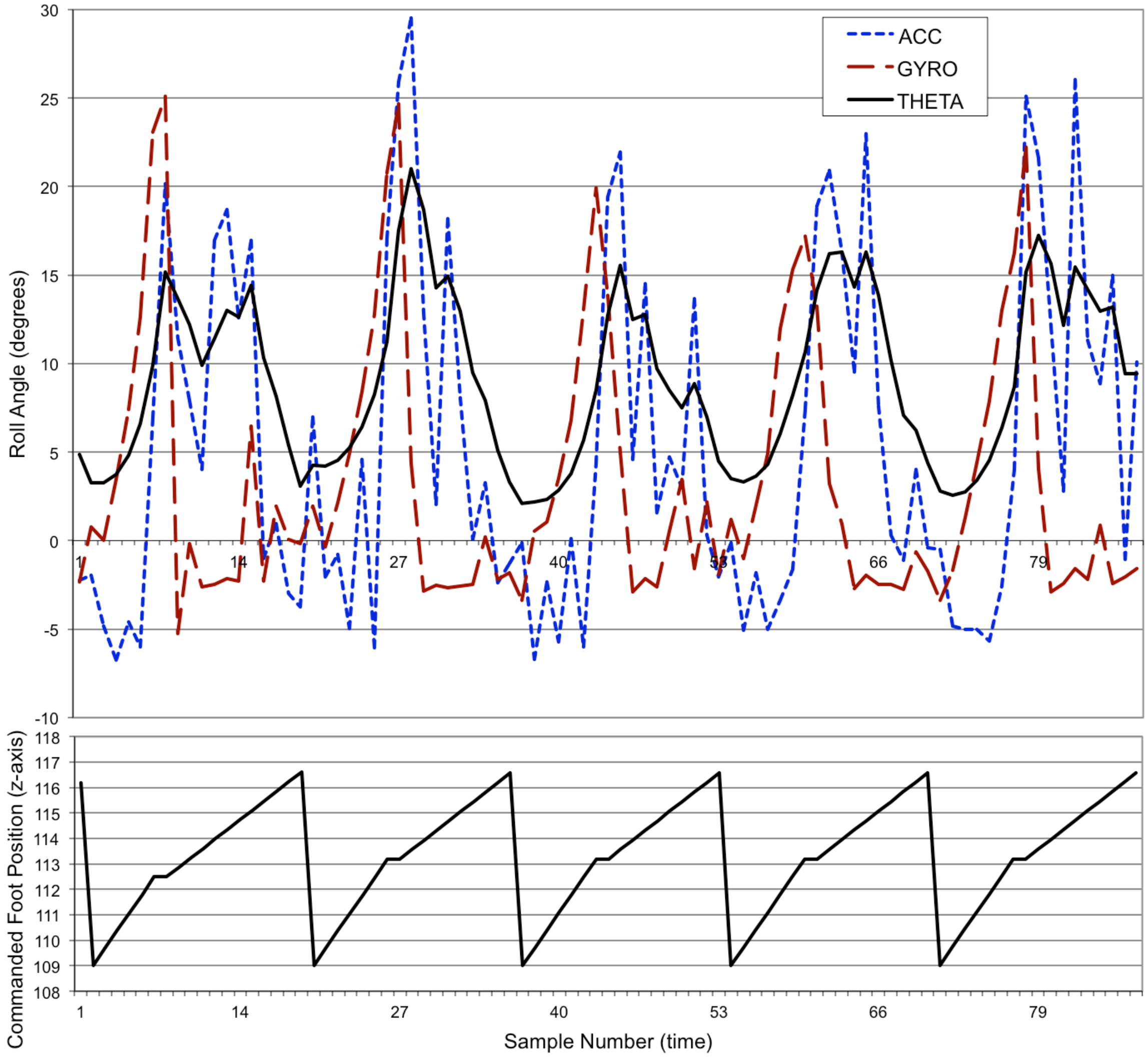


Figure C.9. Catching-a-fall accelerometer, gyro, complementary filter (theta), and z-axis commanded foot position.

The foot is rapidly retracted (seen as decreasing foot position in Figure C.9) to z = 109.0 in the Unload state. During the Load state, the foot is lowered at a fast rate until the foot pressure sensor is nominal. This is indicated on the foot position graph as a flat line (double read). Note that it takes 6*dt* or roughly 0.375 seconds for the foot to touch the ground. Thus supporting the inverted pendulum model and control theory developed in Chapter 6. After the foot contacts the ground, the Draisine robot is up righted at a slower rate until a minimum roll angle of 3.75 degrees is achieved, and the cycle repeats.

In practice, when the leg is rapidly retracted during Unload state and through the Load state, the accelerometer generally outputs negative values. These negative values detrimentally affect the complementary filter. However, the gyroscope is generally accurate. So during Unload and Load states, only the gyroscope data and equation C.2 is used. The complementary filter is used only during Stance state, but this strategy is adequate to correct any gyro drift.

As may also be observed in the plot of Figure C.9, the Draisine robot “bounces” after the leg contacts the ground. The rubber foot most likely compresses, the knee joint appears to flex, and the motors may have compliance, such that the Hirose leg deforms elastically. To reduce this effect, the wheel steering motors and gears were removed for the above experiment.

In summary of Experiment 3, a MEMS accelerometer and gyroscope was added to the Draisine robot to enable the robot to catch its fall and upright itself. The roll angle was computed by 1) integrating angular velocity and 2) using a complimentary filter. Finally, there is roughly 0.3 seconds for the foot to be moved in the x-axis before the foot touches the ground – more than enough time to reposition the foot to take a step and propel the Draisine robot forward.

C.2. Experiment 4 – Dynamic Balance

Like the early Draisine vehicles, the leg of the second-generation prototype is used to both motivate the Draisine robot and prevent it from falling over. Experiment 4 is the culmination of the previous three experiments, combining accurate foot positioning in three dimensions, sensing foot pressure and roll angle, and the Leg State machine (described in Chapter 5) to dynamically balance the Draisine robot. This experiment continues Experiment 3 by enabling motion in the x-axis. A complete listing of the MicroBasic code used in this experiment may be found in Appendix D.

Figure C.10 illustrates the Draisine robot walking through an entire leg cycle. Upper diagram (1) illustrates the Load state, wherein the foot travels rearwards, as in Experiment 1, and incrementally lowers the foot to the ground to develop reaction forces between the foot and the ground, as in Experiment 3. Diagrams (2) through (4) illustrate the Stance state, wherein the foot travels rearward to both propel and upright the Draisine robot, combining Experiments 1, 2, and 3. Diagram (5) illustrates the Unload state, wherein the foot continues to travel rearward and rapidly raises the foot off the ground, as in Experiment 1. Diagram (6) illustrates the Flight state, wherein the leg is repositioned to the start of the stroke, as in Experiment 1, and repositioned to the left or right of the roll axis to catch the fall and upright the robot, as in Experiment 3.

Three methods of uprighting the Draisine robot are proposed: a) the center of mass oscillates on the left or right side of the line of support, b) the center of mass oscillates to the left or right of the line of support, and c) some alternating mixture of (a) and (b). In (a), the robot is brought to a nearly upright condition by the supporting leg, then when the leg is repositioned it falls on that same side, and then the leg catches the fall, as in Experiment 3.

Figure C.11 plots nine steps of the Draisine robot propelling itself and catching its fall, wherein the upper plot the blue short-dashed line is the accelerometer output in degrees, the red long-dashed line is the gyroscope output scaled by 7x for clarity, and the solid black line is the roll angle from the complementary filter, and the lower plot shows the commanded x- and z-axis foot position. The total distance traveled was approximately 14 inches in about 9.5 seconds.

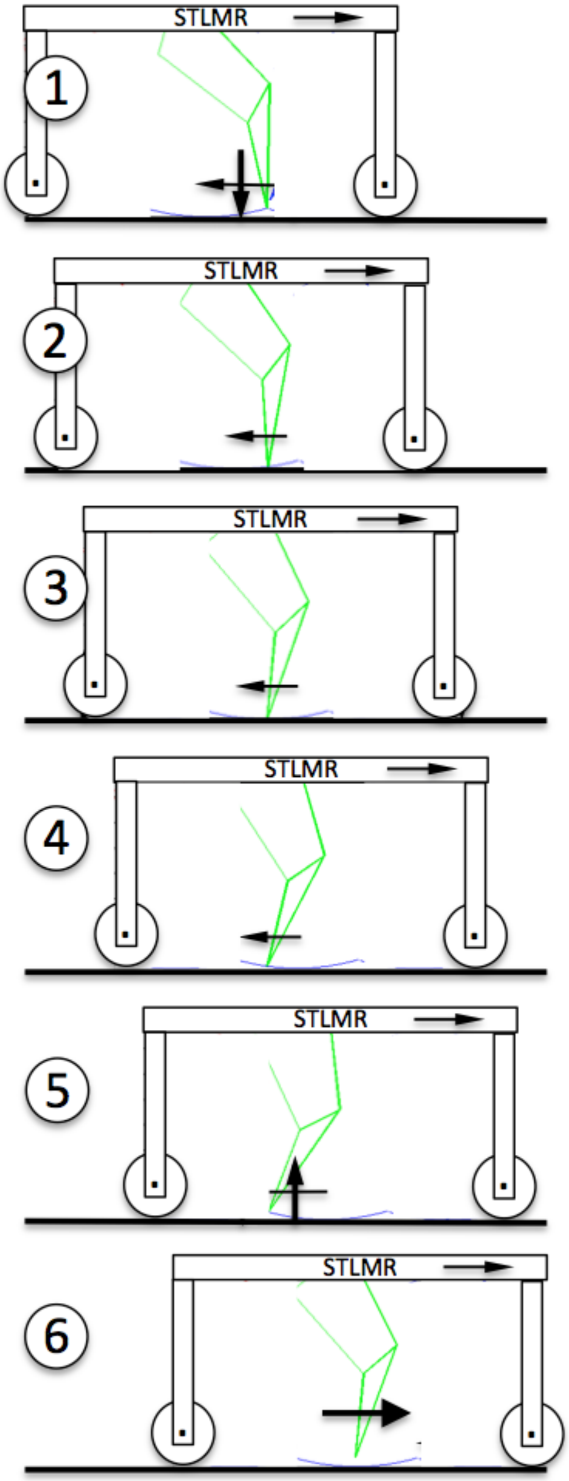


Figure C.10. The wheel-leg-wheel mobile robot shown executing a leg cycle (left to right, 1 through 6, repeating).

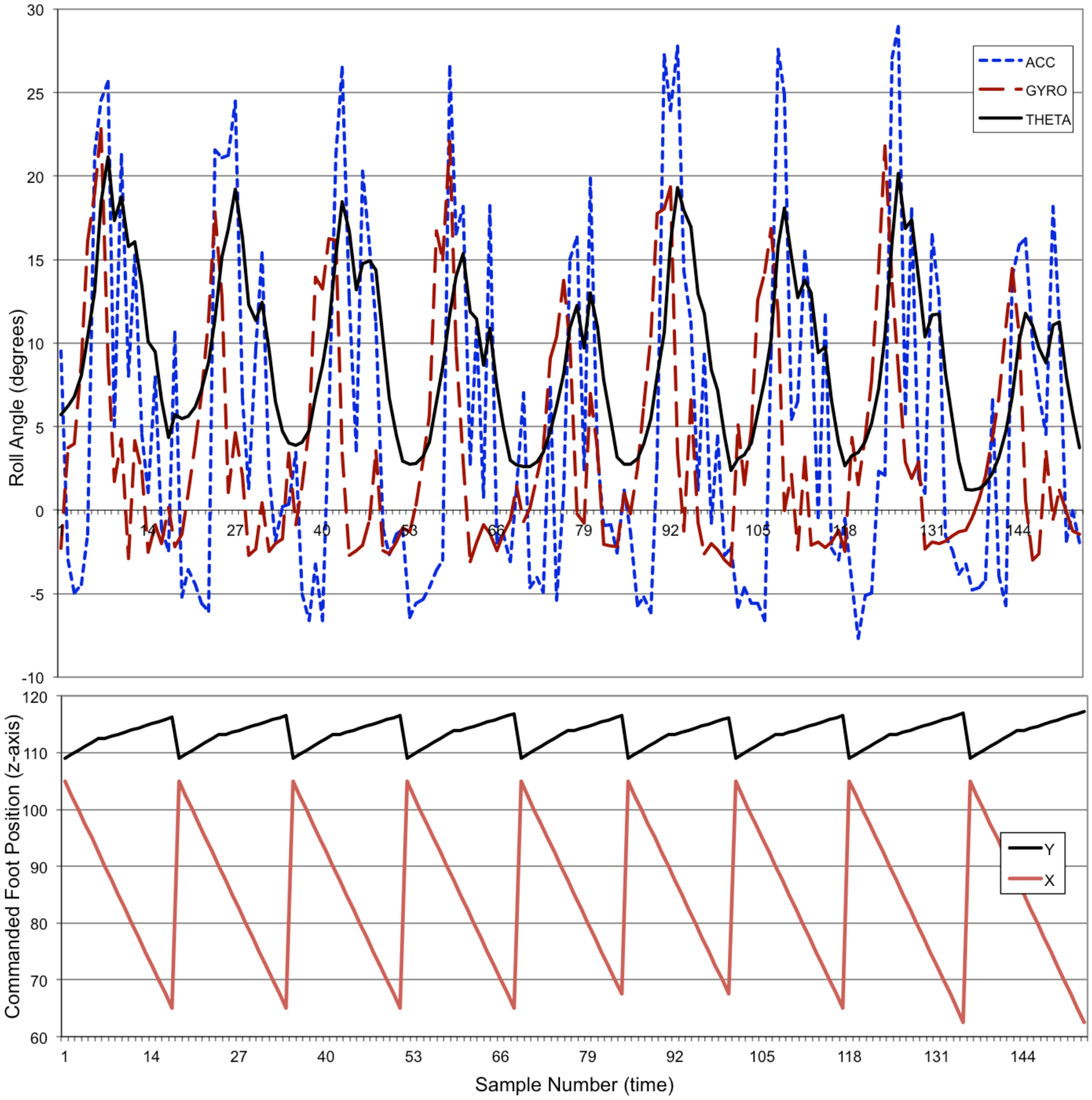


Figure C.11. Draisine robot propelling itself and catching-a-fall accelerometer, gyro, complementary filter (theta), x- and z-axis commanded foot position.

As anticipated from the results of Experiment 3, the Draisine robot is capable of propelling itself in a straight line and catching its fall when the center of mass oscillates on the left or right side of the line of support. Unfortunately, the Hirose leg and theta stage are too slow to test the strategy of oscillating the roll to the left or right of the line of support. Nevertheless, the results are promising.

C.3. Experiment 5 – Leaning Into A Turn

Figure C.12 illustrates the ST3LMR executing a single-track turn whereby the body is spatially and angularly displaced from the normal plane of operation. This is called "leaning into the turn". During this maneuver, the center of gravity and projected center of pressure is moved towards the center of curvature, thus developing a torque about the roll axis (not shown for clarity but refer to the feet) that counteracts the outward centripetal inertial force acting on the center of mass of the body. In leaning into a turn, the feet are following a single track or in-line curve of radius R1 about the center point normal to the ground, and the body is leaning with angle theta between the normal reference plane and the projected plane tangent to the single track or in-line curve on the ground and through the center of gravity and parallel to the body length. The top of the body thus follows a second curve of smaller radius R2 about the projected center point normal to the ground such that the resulting plane of motion is a truncated cone.

For the Draisine or Draisine robot to follow a curved trajectory, it must lean into the turn and use gravitational forces to counteract the outward centripetal force. Implementing this concept in the aforementioned control architecture is straightforward, because the accelerometer senses only the combined forces. So, if the steering angle is dynamically (and gradually) adjusted to follow a curved path, the lean angle will also follow. As in the previous experiment, the goal angle is never zero and the robot is never fully up-righted. Because the accelerometer and gyro sensors measure the combined gravitational and centripetal forces, the robot executes the turn without additional code modification.

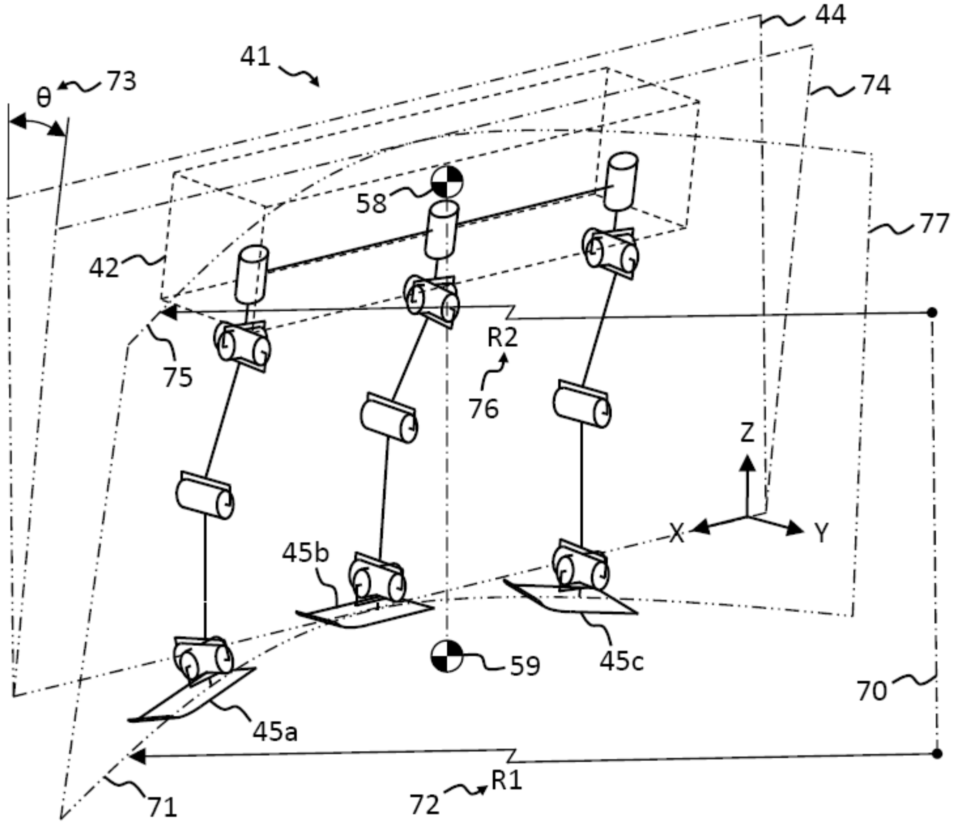


Figure C.12. Skeletal perspective view of the ST3LMR executing a dynamic single-track lean-into-the-turn maneuver. (From US patent 8,457,830 Fig 29.)

Figure C.13 plots eight steps of the Draisine robot leaning into a turn by propelling itself, catching its fall, and bringing itself to a goal angle of 5.0 degrees, wherein the upper plot the blue short-dashed line is the accelerometer output in degrees, the red long-dashed line is the gyroscope output scaled by 7x for clarity, and the solid black line is the roll angle from the complementary filter, and the lower plot shows the commanded x- and z-axis foot position. The total distance traveled was approximately 19 inches in about 8.5 seconds. Note that the steering motors were removed and Draisine robot had to be thrust into the turn to initiate the leaning maneuver.

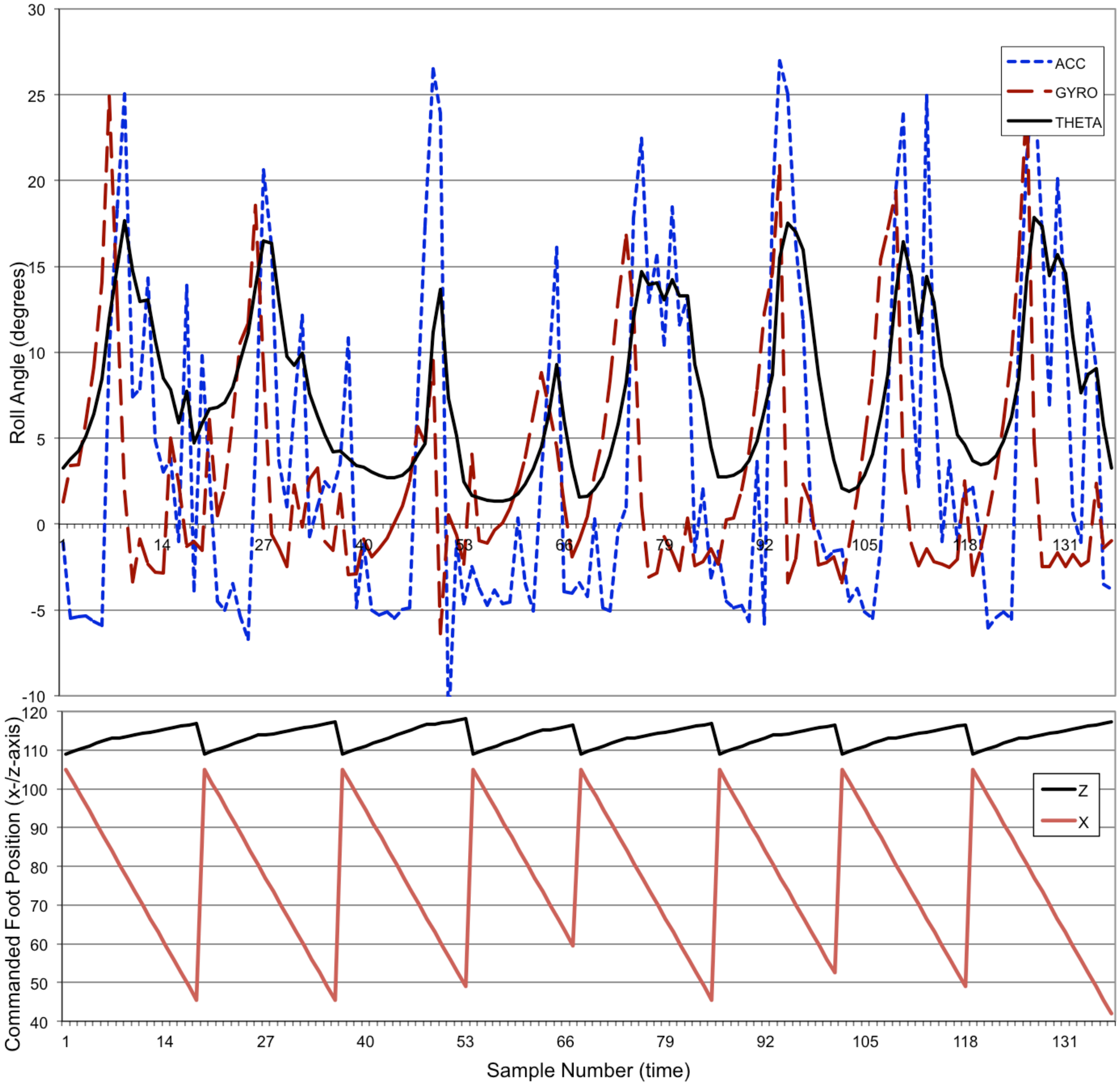


Figure C.13. Draisine robot leaning into a turn, plotting accelerometer, gyro, complementary filter (theta), x- and z-axis commanded foot position.

C.4. Observations from Simulation

Maintaining balance involves uprighting the robot to a near-zero condition, about a center line of support, over a period time. Several goal criteria were evaluated through simulation, such as minimizing roll and roll velocity (theta and thetaDot in the MATLAB code presented in Appendix A). The following summarizes an investigation of the sensitivity of four key design trades affecting multi-hypothesis planning using branch-based scoring in a depth-limited search. These trades include the resolution and breadth of the search, search depth, and stance time or velocity.

The simulation was found to be highly sensitive to the resolution of foot placement in the y-axis. Foot placement is controlled by the variable yf in the MATLAB code (Appendix A), and it is constrained by the maximum physical reach of the legs to the left or right of the hip joint. The model-predictive search uses a two-pass approach to determine viable foot locations, for each step. In the first pass, the entire search space is tested at a high resolution. Typically, only a small subset of this total range is viable. The minimum and maximum viable candidates from the first pass form the set of possible candidates for the second pass and depth search. Search breadth and correspondingly the resolution of side-stepping was found to play an important role in the short-term stability of balance. Changing the depth of the model-predictive search, while holding the search breadth constant, improves short-term stability of balance.

Generally speaking, moving the legs faster provides increased stability of balance. The HiTec HS-7954SH servo used in the Draisine experiments was specified at 333.29 ounce-inches of torque and an operating speed of 300 degrees per second for a 6.0 supply voltage (used for the experimentation). For this servo, the actual performance as measured using the coupled-drive leg provided a closed-loop Stance phase time of 1.84 seconds and an open-loop Flight phase time of 0.44 seconds, both with a stride of 0.0865 meters. As determined by analysis, increasing the voltage to 7.4 volts would improve the operating speed to 352 degrees per second and decreases the Stance period to 1.564 seconds. (Note that this is not recommended by the manufacturer for near-continuous duty applications.) However, from a simple digital simulation (Appendix A), it was found that this level of performance would not be adequate to realize a ST3LMR design using the same coupled-drive leg with point-contact feet. Rather, the recommendation or lesson learned with regard to stability of balance, is that a foot and y-axis (roll) ankle mechanism is required for small, slow-moving in-line robots.

In summary, Experiment 5 demonstrates the Draisine robot capable of leaning into a turn like a Drasine. Maintaining balance involves uprighting the robot to a near-zero condition, as measured by the accelerometer and gyro, such that the sum of torques about the center line of support is held positive, where gravity is (+) and centripetal force is (-). Never fully uprighting the robot, but rather bringing the robot to a near-upright condition proved an effective strategy for leaning into the turn. This would have been difficult to discover/observe with the three active legs of the ST3LMR design, because the line of support is constantly changing as any two legs move the points of support backwards with respect to the center of mass. See Chapters 8 and 9 for additional summary and discussion of the results.

APPENDIX D

DRAISINE ROBOT CODE

A microcontroller program was written in MicroBasic to control the Draisine robot and demonstrate feasibility of the single-track control concept. The annotated MicroBasic code is listed in the following pages. See Chapter 8, Section 8.2 Prototype Draisine Robot and Appendix C, Section C.2 Experiment 4 for additional background and information on the Draisine robot.

```
;EXPERIMENT 3 - CATCHING A FALL
;  Copyright (C) by John R Goulding
;  9 May 2013

;PURPOSE
;  This experiment adds a MEMS accelerometer and gyroscope
;  input to the robot so it may swing its leg outward in
;  the direction of a fall to catch the fall and upright
;  the robot.

;ALGORITHM
;  1. Unload & raise the foot
;  2. Swing the foot outward
;  3. Extend and load the foot (aka catch the fall)
;  4. Upright the robot and monitor foot pressure
;  5. Repeat when upright or maximum foot stroke

;HARDWARE
;  BasicATOMPro Mad Hatter

;INPUT
;  A0/P14 = pFoot, foot pressure sensor
;  A1/P15 = accx, x-axis accelerometer
;  A2/P16 = accy, y-axis accelerometer
;  A3/P17 = gyro, y-axis gyro

;OUTPUT
;  D0/P0 = servo, hip x-axis
;  D1/P1 = servo, calf x-axis
;  D2/P2 = servo, hip theta/y-axis

;**********************************************************
; VARIABLE - VARIABLE - VARIABLE - VARIABLE - VARIABLE
;**********************************************************

servo_p0 VAR SWord
servo_p1 VAR SWord
servo_p2 VAR SWord

pfoot    VAR Word

accx_adc VAR Word
gyro_adc VAR Word

accx     VAR Float
gyro     VAR Float
```

```
theta    VAR Float
legExPty VAR Float

angle13  VAR Float
angle16  VAR Float
angle24  VAR Float
angle45  VAR Float
angle63  VAR Float
angle65  VAR Float

link16   VAR Float
link25   VAR Float

ptx      VAR Float
pty      VAR Float

pt3x     VAR Float
pt3y     VAR Float
pt5x     VAR Float
pt5y     VAR Float

tmp      VAR Float

;**********************************************************
; CONSTANT - CONSTANT - CONSTANT - CONSTANT - CONSTANT
;**********************************************************

nomPFoot     CON 150 ;nominal foot pressure
maxPFoot     CON 1025 ;nominal foot pressure

maxFlightVel CON 1200  ;maximum servo vel. in flight phase
maxLoadVel   CON 800   ;maximum servo vel. in load phase
maxStanceVel CON 500   ;maximum servo vel. in stance phase

dxLoad      FCON 1.0   ;differential foot pos. x-axis
dyLoad      FCON 0.375 ;differential foot pos. y-axis
dxStance    FCON 4.0
dyStance    FCON 0.25

maxFootFwdX FCON 35.0 ;maximum forward foot position x-axis
nomFootFwdX FCON 35.0 ;maximum forward foot position x-axis

nomLegExY   FCON 10.0  ;nominal leg extension z-axis
minFootGndY FCON 110.0 ;maximum forward foot pos. x-axis
nomFootGndY FCON 112.0 ;nominal ground foot pos. y-axis
maxFootGndY FCON 120.0 ;maximum forward foot pos. x-axis
```

```
;**********************************************************
; MAIN - MAIN - MAIN - MAIN - MAIN - MAIN - MAIN - MAIN
;**********************************************************

main
     ;Initialize
     hservo [p0\0] ;hip flexor - set initial servo
                    positions (centered)
     hservo [p1\0] ;knee
     hservo [p2\0] ;hip rotator

     theta = 0.0       ;initial theta stage angle
     ptx = nomFootFwdX ;initial foot (x,y) position
     pty = nomFootGndY

;**********************************************************
; UNLOAD - UNLOAD - UNLOAD - UNLOAD - UNLOAD - UNLOAD
;**********************************************************

state_unload
     ;Lift leg and rotate hip to catch fall
     ;ptx = ptx - 2.1158
     ;pty = minFootGndY

;**********************************************************
; FLIGHT - FLIGHT - FLIGHT - FLIGHT - FLIGHT - FLIGHT
;**********************************************************

state_flight
     ;Lift leg and rotate hip to catch fall
     ptx = maxFootFwdX ;set initial foot (x,y) position
     pty = minFootGndY

     ;find pt3
     link16 = FSQRT(ptx*ptx + pty*pty)
     angle16= FACOS((link16*link16+2423.75)/(link16*156.0))
     if ptx = 0.0 then
          angle63 = 1.5708 - angle16
     else
          if ptx < 0.0 then
               angle63 = FACOS((ptx*-1.0)/link16) - angle16
          else
               angle63=3.14159265-FACOS(ptx/link16)-angle16
          endif
     endif
     pt3y = pty - 78.0*FSIN(angle63)
     pt3x = ptx + 78.0*FCOS(angle63)

     ;find T1
     if pt3x = 0.0 then
          angle13 = 1.5708
     else
          if pt3x > 0.0 then
               angle13 = FATAN(pt3y/pt3x)
          else
               angle13 = 3.14159265 + FATAN(pt3y/pt3x);
          endif
     endif
     servo_p0 = TOINT(angle13*12500.0) - 14750

     ;find pt5
     angle65 = angle63 + 0.1784188
     pt5y = pty - FSIN(angle65)*53.0
     pt5x = ptx + FCOS(angle65)*53.0

     ;find pt4
     tmp = pt5x + 26.0
     link25 = FSQRT(tmp*tmp + pt5y*pt5y)
     if pt5x > -26.0 then
          angle24 = FACOS((link25*link25 -
               4908.75)/(link25*41.0)) +
               FASIN(pt5y/link25) ;angle45 + angle25
     else
          angle24 = FACOS((link25*link25 -
               4908.75)/(link25*41.0)) +
               3.14159265 – FASIN(pt5y/link25) ;angle45 +
angle25
     endif
     servo_p1 = TOINT(angle24*10500.0) - 23000

;implement T1 position
flight_p0
     if (hservoidle p0) then
          hservo [p0\servo_p0\maxFlightVel] ;good k-c
     else
          goto flight_p0
     endif

;implement T2 position
flight_p1
     if (hservoidle p1) then
```

```
          hservo [p1\servo_p1\maxFlightVel] ;good k-c
     else
          goto flight_p1
     endif

update_theta
     ;Read the accelerometer and gyro, load 16bit result
     adin16 P15, accx_adc ;64X read analog pin A1
     adin16 P17, gyro_adc ;64X read analog pin A3

     ;Convert to a differential angle, in radians
     accx = TOFLOAT(accx_adc - 20515)*(0.017000231)
     gyro = TOFLOAT(gyro_adc - 18426)*(0.001094)

     if accx > 30.0 then ;0.35 then
          goto update_theta ;bad read
     endif

     ;Complimentary filter
     ;In this case, a simple gain (P) is more elegant than
     ; a Kalman filter and affords more time to read the
     ; foot pressure sensor for a better estimate (smoother
     ; landing). Note tau accx is: 0.025*0.0625=0.0015625
     theta = (0.75)*(theta + gyro) + (0.25)*(accx)

     ;Temporal estimate (approximation) of final leg angle
     ; at ground contact. Note this is a fairly small and
     ; fast system (as opposed to a larger, full-scale
     ; robot. We really care about having the leg farther
     ; out than the neutral position – so there's no chance
     ; the robot will tip over. Here, the servo bias is
     ; +2500, and the positive estimate is +2500.
     if theta > 0.0 then
          servo_p2 = TOINT(theta*2500.0) + 8000
          if servo_p2 > 9250 then
               servo_p2 = 9250
          endif
     else
          servo_p2 = 8750
     endif

;implement T3 angle
flight_p2
     if (hservoidle p2) then
          hservo [p2\8750\maxFlightVel] ;good k-c
     else
```

```
          goto update_theta
     endif

     ;Compute maximum leg extension
     legExPty = nomlegExY/FCOS(theta) + nomFootGndY
     if legExPty > maxFootGndY then
          legExPty = maxFootGndY
     endif

;**********************************************************
; LOAD - LOAD - LOAD - LOAD - LOAD - LOAD - LOAD - LOAD
;**********************************************************
state_load ;load the leg while monitoring foot pressure
     ptx = ptx - dxLoad
     if pty < maxFootGndY then
          pty = pty + dyLoad
     endif

     ;find pt3
     link16 = FSQRT(ptx*ptx + pty*pty)
     angle16 = FACOS((link16*link16 +
                    2423.75)/(link16*156.0))
     if ptx = 0.0 then
          angle63 = 1.5708 - angle16
     else
          if ptx < 0.0 then
               angle63 = FACOS((ptx*-1.0)/link16) - angle16
          else
               angle63=3.14159265-FACOS(ptx/link16)-angle16
          endif
     endif

     pt3y = pty - 78.0*FSIN(angle63)
     pt3x = ptx + 78.0*FCOS(angle63)

     ;find T1
     if pt3x = 0.0 then
          angle13 = 1.5708
     else
          if pt3x > 0.0 then
               angle13 = FATAN(pt3y/pt3x)
          else
               angle13 = 3.14159265 + FATAN(pt3y/pt3x)
          endif
     endif
     servo_p0 = TOINT(angle13*12500.0) - 14750

     ;find pt5
     angle65 = angle63 + 0.1784188
     pt5y = pty - FSIN(angle65)*53.0
     pt5x = ptx + FCOS(angle65)*53.0

     ;find pt4
     tmp = pt5x + 26.0
     link25 = FSQRT(tmp*tmp + pt5y*pt5y)
     if pt5x > -26.0 then
          angle24 = FACOS((link25*link25 –
               4908.75)/(link25*41.0)) +
               FASIN(pt5y/link25) ;angle45 + angle25
     else
          angle24 = FACOS((link25*link25 –
               4908.75)/(link25*41.0)) + 3.14159265 –
               FASIN(pt5y/link25) ;angle45 + angle25
     endif
     servo_p1 = TOINT(angle24*10500.0) - 23000

;Wait until hip rotates into position
load_p2
     if NOT (hservoidle p2) then
          goto load_p2
     endif

;implement T1 position
load_p0
     if (hservoidle p0) then
          hservo [p0\servo_p0\maxLoadVel] ;good k-c
     else
          goto load_p0
     endif

;implement T2 position
load_p1
     if (hservoidle p1) then
          hservo [p1\servo_p1\maxLoadVel] ;good k-c
     else
          goto load_p1
     endif

     ;read foot pressure sensor
     adin P14, pfoot ;read analog pin P28, load into pfoot
update_accx
```

```
     ;Read the accelerometer and gyro
     adin16 P15, accx_adc ;64X read analog pin A1
     adin16 P17, gyro_adc ;64X read analog pin A3

     ;Convert to a differential angle, in radians
     accx = TOFLOAT(accx_adc - 20515)*(0.017000231)
     gyro = TOFLOAT(gyro_adc - 18426)*(0.001094)

     if accx > 30.0 then ;0.35 then
          goto update_accx ;bad read
     endif

     ;Complimentary filter
     ;During Flight and Load, the accelerometer reading is
     ; very noisy, due to the leg rapidly lifting and
     ; lowering. The gyroscope, however, provides a
     ; reasonably accurate measurement, with low drift for
     ; a short time.
     theta = theta + gyro

     if pfoot < nomPFoot then
          if pty < maxFootGndY then ;maxPty then
               goto state_load
          endif
     endif

update_accx3
     ;Read the accelerometer and gyro
     adin16 P15, accx_adc ;64X read analog pin A1
     adin16 P17, gyro_adc ;64X read analog pin A3

     ;Convert to a differential angle, in radians
     accx = TOFLOAT(accx_adc - 20515)*(0.017000231)
     gyro = TOFLOAT(gyro_adc - 18426)*(0.001094)
     if accx > 30.0 then ;0.35 then
          goto update_accx3 ;bad read
     endif

     ;Complimentary filter
     ;Once the leg is pressing on the ground with some
     ; force, the accelerometer measurement is reasonably
     ; accurate, again.
     theta = (0.75)*(theta + gyro) + (0.25)*(accx)
```

```
;***********************************************************
; STANCE - STANCE - STANCE - STANCE - STANCE - STANCE
;***********************************************************
state_support
     ;Take a step
     ptx = ptx - dxStance

     ;Upright the robot
     if pty < maxFootGndY then ;maxPty then
          pty = pty + dyStance
     endif

     ;find pt3
     link16 = FSQRT(ptx*ptx + pty*pty)
     angle16 =FACOS((link16*link16+2423.75)/(link16*156.0))
     if ptx = 0.0 then
          angle63 = 1.5708 - angle16
     else
          if ptx < 0.0 then
               angle63 = FACOS((ptx*-1.0)/link16) - angle16
          else
               angle63=3.14159265-FACOS(ptx/link16)-angle16
          endif
     endif
     pt3y = pty - 78.0*FSIN(angle63)
     pt3x = ptx + 78.0*FCOS(angle63)

     ;find T1
     if pt3x = 0.0 then
          angle13 = 1.5708
     else
          if pt3x > 0.0 then
               angle13 = FATAN(pt3y/pt3x)
          else
               angle13 = 3.14159265 + FATAN(pt3y/pt3x)
          endif
     endif
     servo_p0 = TOINT(angle13*12500.0) - 14750

     ;find pt5
     angle65 = angle63 + 0.1784188
     pt5y = pty - FSIN(angle65)*53.0
     pt5x = ptx + FCOS(angle65)*53.0

     ;find pt4
     tmp = pt5x + 26.0
     link25 = FSQRT(tmp*tmp + pt5y*pt5y)

     if pt5x > -26.0 then
          angle24 = FACOS((link25*link25 –
               4908.75)/(link25*41.0)) +
               FASIN(pt5y/link25) ;angle45 + angle25
     else
          angle24 = FACOS((link25*link25 –
               4908.75)/(link25*41.0)) +
               3.14159265 – FASIN(pt5y/link25) ;angle45 +
angle25
     endif
     servo_p1 = TOINT(angle24*10500.0) - 23000

;implement T1 position
support_p0
     if (hservoidle p0) then
          hservo [p0\servo_p0\maxStanceVel] ;good k-c
     else
          goto support_p0
     endif

;implement T2 position
support_p1
     if (hservoidle p1) then
          hservo [p1\servo_p1\maxStanceVel] ;good k-c
     else
          goto support_p1
     endif

update_accx2
     ;Read the accelerometer and gyro, load 16bit result
     adin16 P15, accx_adc ;64X read analog pin A1 into accx
     adin16 P17, gyro_adc ;64X read analog pin A3 into gyro

     ;Convert to a differential angle, in radians
     accx = TOFLOAT(accx_adc - 20515)*(0.017000231)
     gyro = TOFLOAT(gyro_adc - 18426)*(0.001094)

     if accx > 30.0 then ;0.35 then
          goto update_accx2 ;bad read
     endif
     ;Complimentary filter
     ;In this case, a simple gain (P) is more elegant than
     ; a Kalman filter and affords more time to read the
     ; foot pressure sensor for a better estimate (e.g.,
     ; a smoother landing). Note tau is moved to accx:
     ;    0.025*0.0625=0.0015625
     theta = (0.75)*(theta + gyro) + (0.25)*(accx)

     ;Check exit conditions
     if ptx > -40.0 then ; exit if foot too far backward
          if theta > 5.0 then ;OR if robot uprighted
               goto state_support
          endif
     endif

     ;Yea, robot is uprighted!
     Goto state_unload ;repeat

;All done!
Goto main
```

REFERENCES

R. Alexander, “Three uses for springs in legged locomotion,” *International Journal of Robotics Research*, vol. 9, no. 2, 1990.

R. Baddeley and P. Hancock, “A statistical analysis of natural images matches psychophysically derived orientation tuning curves,” *Biological Sciences*, vol. 246, no. 1317, pp. 219-223, Dec. 23, 1991.

R. Baddeley, “The correlational structure of natural images and the calibration of spatial representations,” *Cogn. Sci.*, vol. 21, pp. 351-372, 1997.

A. Baerveldt and R. Klang, “A low-cost and low-weight attitude estimation system for an autonomous helicopter,” in *Proc. IEEE Int. Conf. an Intelligent Engineering Systems*, 1997, pp. 391-395.

R. Balboa and N. Grzywacz, "Power spectra and distribution of contrasts of natural images from different habitats," *Vision Research*, vol. 43, pp. 2527-2537, 2003.

W. Baldwin and J. Miller, “Multilegged walker, final report,” *Space General Corp.*, El Monte, CA, 1966.

H. Barlow, Single units and sensation: a neuron doctrine for perceptual psychology? *Perception*, vol. 1, no. 4, pp. 371-394, 1972.

G. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*, MIT, 2005.

M. Bekker, *Off-The-Road Locomotion*, Ann Arbor: University of Michigan Press, 1960.

M. Benjamin, *Interval Programming: A Multi-Objective Optimization Model for Autonomous Vehicle Control*, PhD thesis, Brown Univ. Providence, RI, 2002.

M. Benjamin, “The interval programming model for multi-objective decision making,” *Massachusetts Institute of Technology, Artificial Intelligence laboratory*, AI memo 2004-021, Sep. 2004.

M. Berkemeier and S. Fearing, “Tracking fast inverted trajectories of the underactuated acrobat,” *IEEE Trans. Robot. Automat.*, vol. 15, pp. 740-750, Aug. 1999.

S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, 1999.

R. Briggs, *Real-Time Digital Control of an Electrically Powered Vehicle Leg Using Vector Force Feedback*, M.S. Thesis, Columbus: Ohio State Univ., 1977.

R. Brooks, "A robust layered control system for a mobile robot," *Massachusetts Institute of Technology, Artificial Intelligence laboratory*, AI memo 864, Sep. 1985.

J. Buckett, *Design of an On-Board Electronic Joint System for a Hexapod Vehicle*, M.S. Thesis, Columbus: Ohio State Univ., 1977.

G. Cabodevilla, G. Abba, and H. Sage, "Energetically near optimal gait for a biped robot with double supporting phases," in *Proc. 3rd France-Japan Congr. 1st Europe-Asia Congr. Mechatronics*, Besancon, France, Oct. 1996, pp. 958-961.

C. Canudas, L. Roussel, and A. Goswani, "Periodic stabilization of a 1-dof hopping robot on nonlinear compliant surface," in *Proc. IFAC Symposium on Robot Control*, Nantes, France, Sep. 1977, pp. 405-410.

G. Cavagna, H. Thys, and A. Zamboni, "The sources of external work in level walking and running," *Journal of Physiology*, vol. 262, no. 3, pp. 639-657, 1976.

N. Chaillet, G. Abba, and E. Ostertag, "Double dynamic modeling and computed-torque control of a biped robot," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robotics Systems,* Munich, Sep. 1994, pp. 1149-1153.

C. Chevallereau, A. Formal'sky, and B. Perrin, "Control of a walking robot with feet following a reference trajectory derived from ballistic motion," in *Proc. IEEE Int. Conf. Robotics Automation*, Albuquerque, New Mexico, Apr. 1997, pp. 1094-1099.

C. Chevallereau, D. Djoudi, and J. Grizzle, "Stable bipedal walking with foot rotation through direct regulation of the zero moment point," *IEEE Trans. Robot.*, vol. 24, no. 2, pp. 390-401, 2008.

C. Christensen. *The Innovators Dilema: When New Technologies Cause Great Firms to Fail*. Boston: Harvard Business School Press, 1997.

C. Chow and D. Jacobson, "Studies of human locomotion via optimal programming," *Mathematical Biosciences*, vol. 10, pp. 239-306, 1971.

C. Colby, and M. Goldberg, "Space and attention in parietal cortex," *Annu. Rev. Neurosci.*, vol. 22, pp. 319-349, 1999.

V. Cossalter and R. Lot, “A motorcycle multi-body model for real time simulations based on the natural coordinates approach,” *Vehicle System Dynamics*, vol. 37, no. 6, pp. 423-447, 2002.

F. Delcomyn, “Walking robots and the central and peripheral control of locomotion in insects,” *Autonomous Robots*, vol. 7, pp. 259-270, 1999.

D. Dial, P. DeBitetto, and S. Teler, “Epipolar constraints for vision-aided inertial navigation,” in *Proc. IEEE WACV/MOTION*, pp. 221-228, 2005.

J. Doyle, “The skill of bicycle riding,” PhD thesis, *Dept. of Psychology, Univ. of Sheffield*, Mar. 1987.

J. Doyle, “The essential contribution to bicycle riding,” *Training, Human Decision Making and Control.* North-Holland: Elsevier, pp. 351-370, 1988.

K. Drais, “The Velocipede or Draisena,” *Analectic Magazine*, pp. 518-519, January 1819. (p. 518, Google books link).

K. Drais, “Improved velocipede,” *Mechanics Magazine*, vol. XVII, no. 477, pp. 417-418, September 1832. (p. 418, Google books link).

P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” Jet Propulsion Lab (JPL) white paper, Pasadena, CA, 2007.

A. Frank, “Automatic Control Systems for Legged Locomotion,” Univ. of So. Cal., USCEE Report No. 273, Los Angeles, 1968.

C. Francois and C. Samson, “A new approach to the control of the planar one-legged hopper,” *Intl. Jnl. of Robotics Res.*, vol. 17, no. 11, pp. 1150-1166, 1998.

E. Foxlin and L. Naimark, “VIS-Tracker: A wearable vision-inertial self-tracker,” in *Proc. IEEE VR*, Los Angeles, 2003, pp. 199-206.

T. Fukuda, M. Doi, Hasegawa, and H. Kajima, *Fast Motions in Biomechanics and Robotics*, chapter Multi-Locomotion Control of Biped Locomotion and Brachiation Robot, pp. 121-145, Heidelberg: Springer-Verlag, 2006.

J. Furusho and M. Masubuchi, “Control of a dynamical biped locomotion system for steady walking,” *J. Dyn. Syst., Meas., Control*, vol. 108, pp. 111-118, 1986.

J. Furusho and A. Sano, “Sensor-based control of a nine-link biped,” *Int. J. Robot. Res.*, vol. 9, no. 2, pp. 83-98, 1990.

T. Gawne, T. Kjaer, and B. Richmond, “Latency: another potential code for feature binding in striate cortex”, *J. of Neurophysiology*, vol. 76, no. 2, pp. 1356-1360, 1996.

K. Goldberg, and M. Raibert, “Conditions for symmetric running in single and double support,” in *Proc. of IEEE International Symposium on Robotics*, Raleigh, 1987.

A. Goswani, B. Espiau, and A. Keramane, “Limit cycles and their stability in passive bipedal gait,” in *Proc. IEEE Intl. Conf. on Robotics and Automation*, Minneapolis, April 1996, pp. 246-251.

J. Goulding, “Biologically-Inspired Image-Based Sensor Fusion Approach to Compensate Gyro Sensor Drift in Mobile Robot Systems that Balance.” in *Proc. of IEEE MFI Conf.*, Salt Lake City, 2010.

J. Goulding, “The Single-Track Three Legged Mobile Robot.” in *Proc. of ASME IDETC/CIE, 36th Mechanisms and Robotics Conf.*, Chicago, Aug. 15, 2012.

J. Goulding, US patent 8,457,830. Washington: United States Patent and Trademark Office, 2013.

J. Grizzle, F. Plestan, and G. Abba, “Poincare's method for systems with impulse effects: application to mechanical biped locomotion,” in *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, 1999, vol. 4, pp. 3869-3876.

J. Grizzle, G. Abba and F. Plestan, “Asymptotically stable walking for biped robots: Analysis via systems with impulse effects,” *IEEE Trans. Automat. Contr.*, vol. 46, pp. 51-64, Jan. 2001.

E. Guizzo, “Japanese humanoid robot can keep its balance after getting kicked,” *IEEE Spectrum*, 8 May 2012.

F. Hardarson, “Locomotion for Difficult Terrain,” Royal Institute of Technology, Dept. of Machine Design, Mechatronics Division, April, 1 1997.

S. Harris, *Horse Gaits, Balance and Movement*. New York: Howell Book House, 1993.

Y. Hasegawa, T. Arakawa, and T. Fukuda, “Trajectory generation for biped locomotion,” *Mechatronics*, vol. 10, no. 1-2, pp. 67-89, Mar. 2000.

K. Hashimoto, Y. Sugahara, H-O. Lim, A. Takanishi, “Human-carrying biped walking vehicle,” *Gerontechnology*, vol. 7, no. 2, pp. 119, 2008.

H. Hatze, “The complete optimization of a human motion,” *Mathematical Biosciences*, vol. 28, pp. 99-135, 1976.

J. Heaston, *Design of a Novel Tripedal Locomotion Robot and Simulation of a Dynamic Gait for a Single Step*, M.S. Thesis, Virginia Polytechnic Inst. and State Univ., Blacksburg, 2006.

C. Heath and D. *Heath, Made to Stick: Why Some Ideas Survive and Others Die*. New York: Random House, 2007.

Herr, Seyfarth, and Geyer, “Speed-Adaptive Control Scheme For Legged Running Robots,” *U.S. Patent and Trademark Office*, no. 2002/0085948, 2002.

K. Hirai, M. Hirose, Y. Haikawa, and T. Takenake, “The development of Honda humanoid robot,” in *Proc. IEEE Int. Conf. Robotics Automation*, Leuven, Belgium, May 1998, pp. 1321-1326.

S. Hirose, “A study of design and control of a quadruped walking,” *Int. J. Robotics Res*., vol. 3, no. 2, pp. 113-133, 1984.

H. Hildebrand, “Symmetrical Gaits of Horses,” *Science*, 1967, vol. 150, pp. 701-708.

S. Hirose, Y. Fukuda, K. Yoneda, A. Nagakubo, H. Tsukagoshi, K. Arikawa, G. Endo, T. Doi, and R. Hodoshima, “Quadruped walking robots at Tokyo Institute of Technology: Design, Analysis, and Gait Control Methods,” *IEEE Robotics and Automation Mag.*, pp. 104-114, June 2009.

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

S. Holste, and D. Ciccimaro, “Increasing the Mobility of Dismounted Marines,” Technical Report No. 1988, SPAWAR Systems Center Pacific, Oct. 2009.

D. Hong, H. Lee, H. Cho, Y. Park, and J. Kim, "Visual Gyroscope: Integration of Visual Information with Gyroscope for Attitude Measurement of Mobile Platform," *Intl. Conf. Cntl., Automation and Systems*, Seoul, Oct. 2008.

J. Hu, J. Pratt, and G. Pratt, “Adaptive dynamic control of a bipedal walking robot with radial basis function neural networks,” in *Proc. IEEE/RSJ Intl., Conf. on Intel. Robots and Systems*, Victoria, B. C., Canada, Oct. 1998, pp. 400-405.

Z. Hu, U. Keiichi, H. Lu, and F. Lamosa, “Fusion of vision, 3D gyro and GPS for camera dynamic registration,” in *Proc. 17th Int. Conf. Pattern Recog*., 2004, pp. 351-354.

Q. Huang, Y. Nakamura, and T. Inamura, “Humanoids walk with feed-forward dynamic pattern and feedback sensory reflection,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 4220-4225.

Y. Hurmuzlu and D. Marghitu, “Rigid body collisions of planar kinematic chains with multiple contact points,” *Int. J. Robot. Res.*, vol. 13, no. 1, pp. 82-92, 1994.

R. Imamura, T. Takei, and S. Yuta, “Sensor Drift Compensation and Control of a Wheeled Inverted Pendulum Mobile Robot,” *IEEE Int. Workshop on Advance Motion Control*, 2008, pp. 137-142.

A. Isidori, *Nonlinear Control Systems: An Introduction*, 3rd ed. Berlin: Springer-Verlag, 1995.

K. Ito, F. Matsuno, and R. Takahashi, “Underactuated crawling robot,” in *Proc. IEEE Int. Conf. Intelligent Robotic Systems*, Takamatsu, Japan, Oct.-Nov. 2000, pp. 1684-1689.

W. Johnson, *Two-Time-Scale Predictor Model Using Scan of Anticipated Input*. Case Institute of Technology, Masters Thesis, June 1965.

S. Kajita and K. Tani, “Experimental study of biped dynamic walking,” *IEEE Control Syst. Mag.*, vol. 16, pp. 13-19, Feb. 1996.

D. Kar, K. Issac, and K. Jayarajan, “Gaits and energetics in terrestrial legged locomotion”, *Mechanism and Machine Theory*, vol. 38, pp. 355–366, 2003.

R. Katoh and M. Mori, “Control method of biped locomotion giving asymptotic stability of trajectory,” *Automatica*, vol. 20, no. 4, pp. 405-414, 1984.

P. Kessler, “Motorcycle navigation with two sensors,” Master’s thesis, Dept. of Mech. Engrg., Univ. of Calif. at Berkeley, 2004.

O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Intl. J. of Robotics Res.*, vol. 5, no. 1, pp. 90-98, 1986.

J. Kim, J. Kim, and J. Oh, “Adjustment of Home Posture of a Biped Humanoid Robot Using an Inertial Sensor and Force Torque Sensors”, in *Proc. IEEE/RSJ Int. Conf. Intel. Robots and Systems*, San Diego, Oct. 29-Nov. 2, 2007, pp. 2223-9.

J. Kim, J. Lee, I. Park, and J. Oh, “Vibration Reduction Control For Human-Riding Biped Robot, HUBO FX-1,” *IFAC*, 2006.

J. Kim, J. Lee, and J. Oh, “Experimental realization of dynamic walking for a human-riding biped robot, HUBO FX-1”, *Advanced Robotics*, vol. 21, no. 3-4, pp. 461-484, 2007.

H. Kimura and Y. Fukuoka, “Adaptive dynamic walking of a quadruped robot on irregular terrain by using neural system model,” in *Proc. IEEE/RSJ Int. Conf. Intel. Robots and Systems*, 2000, pp. 979-984.

D. Koditschek and M. Buhler, “Analysis of a simplified hopping robot”, *Intl. J. of Robotics Research*, vol. 10, no. 6, pp. 587-605, 1991.

V. Krovi and V. Kumar, “Modeling and control of a hybrid locomotion system”, Master’s thesis, Mechanical Engineering, Univ. of Pennsylvania, 1996.

B. Kuo, “The relative roles of feedforward and feedback in the control of rhythmic movements, *Motion Control*, vol. 6, pp. 129-145, 2002.

R. Kurazume, S. Hirose, and K. Yoneda, “Feedforward and feedback dynamic trot gait control for a quadruped walking vehicle,” in *Proc IEEE Int. Conf. Tobot. Autom.*, Seoul, Korea, May 21-26, 2001, pp. 3172-3180.

N. Kyriakoulis, E. Karakasis, A. Gasteratos, and A. Amanatiadis, “Pose Estimation of a Volant Platform With a Monocular Visuo-Inertial System,” *Intl. Wkshp. on Imaging Systems and Techniques*, May 2009.

T. Larsen, N. Andersen, O. Ravn, and N. Poulsen, “Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter,” in *Proc. 37th IEEE Conf. Decision and Control*, 1998, vol. 4, pp. 3972-3977.

H. Lee and S. Jung, “Gyro Sensor Drift Compensation by Kalman Filter to Control a Mobile Inverted Pendulum Robot System,” in *Proc. IEEE Int. Conf. on Industrial Technology*, 2009.

Levandowski, A. Schultz, C. Smart, A. Krasnov, H. Chau, B. Majusiak, F. Wang, D. Song, J. Yi, H. Lee, and A. Parish, “Ghostrider: Autonomous motorcycle,” in *Proc. IEEE Int. Conf. Robot. Autom. (video)*, Orlando, FL, 2006.

M. Lewis, “Detecting surface features during locomotion using optic flow,” *Intl. Conf. on Robotics and Automation*, 2002.

M. Lewis and G. Bekey, “Gait adaptation in a quadruped robot,” *Autonomous Robots*, vol. 12, pp. 301-312, 2002.

M. Lewis, R. Etienne-Cummings, M. Hartmann, Z. Xu, and A. Cohen, “An in silico central pattern generator: Oscillator, entrainment, motor neuron adaptation and biped mechanism control,” *Biological Cybernetics*, vol. 88, no. 2, pp. 137-151, 2003.

M. Lewis, A. Fagg, and G. Bekey, “Genetic algorithms for gait synthesis in a hexapod robot,” *World Scientific*, New Jersey, Recent Trends in Mobile Robotics, Y. Zheng, editor, pp. 317-331, 1994.

M. Lewis, F. Tenore, and R. Etienne-Cummings, “CPG design using inhibitory networks,” *IEEE Intl. Conf. on Robotics and Automation*, pp. 3682-3687, 2005.

D. Limebeer and R. Sharp, “Bicycles, motorcycles, and models,” *IEEE Control Syst. Mag.*, vol. 26, no. 5, pp. 34-61, 2006.

R. Listen and R. Mosher, “A versatile walking truck,” in *Proc. Transport. Eng. Conf.*, ASME-NYAS, Wash., D.C., 1968.

J. Lobo and J. Dias, "Vision and Inertial Sensor Cooperation Using Gravity as a Vertical Reference," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1597-1608, Dec. 2003.

P. Manoonpong, T. Geng, T. Kulvicius, B. Porr, and F. Worgotter, “Adaptive, fast walking in a biped robot under neuronal control and learning, *PLos Computational Biology*, vol. 3, issue 7, pp. 1305-1320, Jul. 2007.

D. Marr, and E. Hildreth, “Theory of Edge Detection,” Proc. Of the Royal Society of London. Series B, *Biological Sciences*, vol. 207, no. 1167, pp. 187-217, Feb. 1980.

D. Marhefka and D. Orin, “Fuzzy control of quadrupedal running,” in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, pp. 3063-3069, 2000.

MathWorks, R2013b Documentation Center. Mathworks.com 2013.

S. McCredie, *Balance: In Search of the Lost Sense*. New York: Little, Brown and Co., Ch. 1, 2007.

R. McGhee, “Finite State Control of Quadruped Locomotion,” in *Proc. 2nd Int. Sym. External Control of Human Extremities*, Dubrovnik, Yugoslavia, 1966.

R. McGhee and A. Frank, “On the Stability Properties of Quadruped Creeping Gaits,” *Math. Biosci.*, vol. 3, pp. 331-351, 1968.

J. McKenney, "Investigation for a walking device for high efficiency lunar locomotion," *American Rocket Society*, Space Flight to the Nation, no. 2016-61, 1961.

T. McMahon, "The role of compliance in mammalian running gaits," *J. of Experimental Biology*, vol. 115, pp. 263-282, 1985.

T. McMahon and G. Cheng, "The mechanics of running: How does stiffness couple with speed?," *Journal of Biomechanics*, vol. 23, no. suppl. 1, pp. 65-78, 1990.

J. Mitchell and D. Keirsey, "Planning strategic paths through variable terrain data," in *Proc. SPIE Conf. App. Artif. Intel.*, vol. 485, Arlington, VA, pp. 172-179, 1984.

K. Mitobe, N. Mori, K. Aida, and Y. Nasu, "Nonlinear feedback control of a biped walking robot," in *Proc. IEEE Int. Conf. Robotics Automation*, Nagoya, Japan, May 1995, pp. 2865-2870.

G. Moore, *Crossing the Chasm*. New York: Harper, 1991.

R. Mosher, "Test and evaluation of a versatile walking truck," in *Proc. Off-Road Mobility Res. Sym.*, Intl. Soc. Terrain Vehicle Sys., Washington, D.C., 1968, pp. 359-379.

R. Mosher, "Exploring the Potential of a Quadruped," *Int. Automotive Engr. Cong.*, Detroit, Michigan, 1969, SAE Paper no. 690191.

K. Mostafa, C-S. Tsai, and I. Her, "Alternative gaits for multiped robots with leg failures to retain maneuverability," *Int. J. Adv. Robotic Sys.*, vol. 7, no. 4, pp. 31-38, 2010.

E. Muybridge, *Animals in Motion*. New York: Dover, 1957. (First published in 1899.)

E. Muybridge, *The Human Figure in Motion*. New York: Dover, 1955. (First published in 1901.)

J. Nakanishi, T. Fukuda, and D. Koditschek, "A brachiating robot controller," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 109-123, Apr. 2000.

A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, pp. 145-175, 2001.

K. Ono, R. Takahashi, A. Imadu, and T. Shimada, "Self-excitation control for biped walking mechanism," in *Proc. IEEE Int. Conf. Intelligent Robots Systems*, Takamatsu, Japan, Oct.-Nov. 2000, pp. 1143-1148.

OpenCv Wiki. [Online] Available: http://opencv.willowgarage.com/wiki

M. Osada, H. Mizoguchi, Y. Asano, T. Kozuki, J. Urata, Y. Nakanishi, K. Okada, M. Inaba, “Design of humanoid body trunk with ‘multiple spine structure’ and ‘planar-muscle-driven’ system for achievement of humanlike powerful and lithe motion,” in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2011, pp. 2217-2222.

J. van den Ouden, “Inventory of bicycle motion for the design of a bicycle simulator,” Master’s thesis, Engr. Mechanics, Delft Univ., the Netherlands, Jan. 14, 2011.

H. Park, J. Kim, and J. Oh, “Online walking pattern generation and its application to a biped humanoid robot — KHR-3 (HUBO)”, *Advanced Robotics*, vol. 22, pp. 159-190, 2008.

H. Park, K. Sreenath, and J. Grizzle, “Identification of a bipedal robot with a compliant drivetrain: Parameter estimation for Control Design,” *IEEE Control Systems*, pp. 63-88, Apr. 2011.

J. Park and C. Cho, “An on-line trajectory modifier for the base link of biped robots to enhance locomotion stability,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 3353-3358.

K. Pathak, J. Franch, and S. K. Agrawal, “Velocity Control of a Wheeled Inverted Pendulum by Partial Feedback Linearization,” I*EEE Conf. on Decision and Control*, pp. 3962-3967, 2004.

J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Koffman, 2nd Ed, 1988.

J. Pearl, *Causality: Models, Reasoning, and Inference*. Cambridge: Cambridge University Press, 2000.

D. Perkel, G. L. Gerstein and G. P. Moore, “Neuronal spike trains and stochastic point processes I: The single spike trains,” and “Neuronal spike trains and stochastic point processes II: Simultaneous spike trains,” *Biophysical J.*, vol. 7, pp. 391-418, 1967.

D. Perkel and T. H. Bullock, “Neural coding,” *Neurosciences Research Program Bulletin*, vol. 6, no. 3, pp. 221-348, 1968.

F. Pfeiffer, K. Loffler, and M. Gienger, “The concept of jogging Johnie,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 3129-3135.

F. Plestan, J. Grizzle, E. Westervelt, and G. Abba, “Stable walking of a 7-dof biped robot,” *IEEE Trans Robot. Automat.*, vol. 19, no. 4, pp. 653-668, 2003.

A. Pouget and T. J. Sejnowski, “Spatial transformations in the parietal cortex using basis functions,” *J. Cog. Neurosci.*, vol. 9, no. 2, pp. 222-237, 1997.

I. Poulakakis, “Stabilizing monopedal robot running: reduction-by-feedback and compliant hybrid zero dynamics”, Master’s thesis, Electrical Engineering, Univ. of Michigan, 2009.

J. Pratt, M. Chee, A. Torres, P. Dilworth, and G. Pratt, “Virtual model control: An intuitive approach for bipedal locomotion,” *Int. J. Robot. Res.*, vol. 20, no. 2, pp. 129-143, Feb. 2001.

J. Pratt and G. Pratt, “Intuitive control of a planar bipedal walking robot,” in *Proc. IEEE Int. Conf. Robot. Automat.*, Leuven, Belgium, May 1998, pp. 2014-2021.

D. Pugh, *An Autopilot for a Terrain-Adaptive Hexapod Vehicle*, M.S. Thesis, Columbus: Ohio State Univ., 1982.

W. Rankine, “On the dynamical principles of the motion of velocipedes,” *The Engineer*, vol. 28, pp. 79, 129, 153, and 175, 1869 and vol. 29, pp. 2, 1870.

M. Raibert, “Hopping in legged systems-Modeling and simulation for the 2D one-legged case,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. 14, pp. 451-463, 1984.

M. Raibert, “Legged robots,” *Commun. ACM*, vol. 29, no. 6, pp. 499-514, 1986a.

M. Raibert, *Legged Robots That Balance*, Cambridge: MIT Press, 1986b.

M. Raibert, “Symmetry in running,” *Science*, vol. 231, pp. 1292-1294, 1986c.

M. Raibert and I. Sutherland, “Machines that walk,” *Scientific American*, vol. 248, no. 1, pp. 32-41, Jan. 1983.

M. Raibert, S. Tzafestas, and C. Tzafestas, “Comparitive simulation study of three control techniques applied to a biped robot,” in *Proc. IEEE Int. Conf. Systems, Man Cybernetics Systems Engineering Service Humans*, Le Touquet, France, Oct. 1993, pp. 494-502.

M. Raibert, K. Blankespoor, G. Nelson, R. Playter, and the BigDog Team, “BigDog, the Rough-Terrain Quadruped Robot,” in *Proc. of the 17th World Congress, Intl. Fed. of Automatic Control*, Seoul, July 6-11, 2008.

J. Raol. Multi-Sensor Data Fusion with MATLAB. CRC Press, 2010.

D. Rathbun and B. Capozzi, “Evolutionary approaches to path planning through uncertain environments,” *American Institute of Aeronautics and Astronautics*, AIAA no. 3455, 2002.

H. Rehbinder and X. Hu, “Nonlinear Pitch and Roll Estimation for Walking Robots,” in *Proc. IEEE Intl. Con. Robotics & Automation*, San Francisco, April 2000, pp. 2617-2622.

H. Rehbinder and X.Hu, “Drift-free attitude estimation for accelerated rigid bodies,” *Automatica*, vol. 40, pp. 653-659, 2004.

J. Reynolds, L. Chelazzi, and R. Desimone, “Competitive mechanisms subserve attention in macaque areas V2 and V4,” *J. of Neuroscience*, vol. 19, no. 5, pp. 1736-1753, 1999.

R. Ringrose, *Self-Stabilizing Running*, PhD thesis, MIT, 1996.

J. Rosaa-Flores, J. Alvarez-Gallegos, and R. Castro-Linares, “Stabilization of a class of underactuated systems,” in *Proc. IEEE Int. Conf. Decision Control*, Sydney, Dec. 2000, pp. 2168-2173.

S. Sanborn, “Everyone should walk,” *Wisconsin Engineer*, J. Van Derhei ed., vol. 77, no. 2, pp. 8-9, Nov. 1972.

Sano and J. Furusho, “Realization of natural dynamic walking using the angular momentum information,” in *Proc. IEEE Int. Conf. Robotics Automation*, Cincinnati, May 1990, pp. 1476-1481.

U. Saranli, W. Schwind, and D. Koditschek, “Toward the control of a multi-jointed, monopod runner,” in *Proc. IEEE Int. Conf. Robotics Automation*, Leuven, Belgium, May 1998, pp. 2676-2682.

R. Sharp, “The stability and control of motorcycles,” *J. Mech. Eng. Sci.*, vol. 13, no. 5., pp. 316-329, 1971.

T. Sheridan, “Three models of preview control,” *IEEE Trans. Human Factors in Electronics*, vol. HFE-7, no. 2, pp. 91-102, June 1966.

J. Shigley, “The mechanics of walking vehicles,” in *Proc. 1st Int. Conf. on Mechanics of Soil–Vehicle Systems*, Turin, 1961.

R. Simmons, “The curvature-velocity method for local obstacle avoidance,” in *Proc IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, April 1996, pp. 3375-3382.

J. Smith, “Galloping, bounding and wheeled-leg modes of locomotion on underactuated quadrupedal robots”, Master’s thesis, Mechanical Engineering, McGill Univ., 2006.

A. Smith and M. Berkemeier, “The motion of a finite-width wheel in 3D,” in *Proc. IEEE Intl. Conf. on Robotics and Automation, Leuven, Belgium*, 1998, pp. 2345-2350.

M. Spong, “The swing up control problem for the acrobat,” *IEEE Control Syst. Mag.*, vol. 15, pp. 49-55, Feb. 1995.

K. Sreenath, H. Park, and J. Grizzle, “Design and Experimental Implementation of a Compliant Hybrid Zero Dynamics Controller with Active Force Control for Running on MABEL,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Saint Paul, MN, May 14-18, 2012, pp. 51-56.

A. Stentz, “Optimal and efficient path planning for partially-known environments,” in *Proc Int. Conf. Robot. Automat.*, Los Alamitos, CA, 1994, vol. 4, pp. 3310-3317.

K. Suga and M. Yamaoka, “Legged Robot,” US Patent 20090009124, Jan 8, 2009.

T. Sugihara, Y. Nakamura, and H. inoue, “Realtime humanoid motion generation through ZMP manipulation based on inverted pendulum control,” in *Proc. IEEE int. Conf. Robot. Autom.*, 2002, pp. 1404-1409.

I. Sutherland and M. Ullner, “Footprints in the asphalt,” *INt. J. Robot. Res.*, vol. 3, no. 2, pp. 29-36, 1984.

E. Switkes, M. Mayer and J Sloan, “Spatial frequency analysis of the visual environment: anisotropy and the carpentered environment hypothesis,” *Vision Research*, vol. 18, pp. 1393-1399, 1978.

K. Takita, R. Hodoshima, and S. Hirose, “Fundamental Mechanism of Dinosaur-Like Robot TITRUS-II Utilizing Coupled Drive,” in *Proc. of IEEE Intelligent Robots and Systems, IROS 2000, Takamatsu, Japan*, 2000.

K. Takita, T. Katayama, and S. Hirose, “Development of Miniature Dinosaur-like Robot TITRUS-III,” in *Proc. Intl. Conf. on Intelligent Robots and Systems, Maui*, 2001.

Y. Tanaka and T. Murakami, “Self sustaining bicycle robot with steering controller,” in *Proc IEEE Adv. Motion Control Conf.*, 2004, pp. 193-197.

B. Thuilot, A. Goswani, and B. Espiau, “Bifurcation and chaos in a simple passive bipedal gait,” in *Proc. IEEE Intl. Conf. on Robotics and Automation, Albuquerque*, April 1997, pp. 792-798.

T. Thornton, “Computer animation of quadrupedal locomotion”, Master’s thesis, Texas A&M University, Visualization Sciences, Dec 2004.

S. Thorpe, Fize, D., & Marlot, C. “Speed of processing in the human visual system.” *Nature*, vol. 381, pp. 520-522, 1996.

S. Thorpe, “Spike arrival times: a highly efficient coding scheme for neural networks,” *Parallel processing in neural systems*, North-Holland: Elsevier, R. Eckmiller, G. Hartman, and G. Hauske (Eds.), pp. 91-94, 1990.

S. Thorpe and J. Gautrais, “Rapid visual processing using spike asynchrony,” *Advances in neural information processing systems,* Cambridge: MIT Press, M. Mozer, M. Jordan, and T. Petsche (Eds.), vol. 9, pp. 901-907, 1997.

S. Thorpe and Gautrais, J. “Rank order coding: a new coding scheme for rapid processing in neural networks,” *Computational neuroscience: trends in research*, New York: Plenum Press. 1998.

S. Thorpe, A. Delorme, and R. VanRullen, “Spike-based strategies for rapid processing,” *Neural Networks*, vol. 14, no. 6-7, pp. 715-725, 2001.

R. Tomovic and W. Karplus, “Land Locomotion-Simulation and Control,” in *Proc. of the IRE, translated*, Jan 1961, vol. 49, pp. 268-275.

A. Torralba and A. Oliva, “Statistics of natural image categories,” *Network: Comput. Neural Syst*, vol. 14, pp. 391-412, 2003.

A. van der Schaff and J. van Hateren, "Modelling the Power Spectra of Natural Images: Statistics and Information," *Vision Research*, vol. 36, no. 17, pp. 2759-2770, 1996.

R. VanRullen and S. Thorpe, “The time course of visual processing: from early perception to decision-making,” *J. of Cognitive Neuroscience*, vol. 13, no. 4, pp. 454-461, 2001.

R. VanRullen and S. Thorpe, “Surfing a spike wave down the ventral stream,” *Vision Research*, vol. 42, pp. 2593-2615, 2002.

M. Vukobratovic, B. Borovac, D. Surla, and D. Stokic, *Biped Locomotion*, Berlin: Springer-Verlag, 1990.

M. Vukobratovic and B. Borovac, “Zero moment point – thirty five years of its live,” *Int. J. Humanoid Robot.*, vol. 1, no. 1, pp. 157-173, 2004.

M. Vukobratovic and J. Stepanenko, “On the stability of anthromorphic systems,” *Math. Biosci.*, vol. 15, no. 1, pp. 1-37, 1972

K. Waldron, V. Vohnout, A. Pery, and R. McGhee, “Configuration Design of the Adaptive Suspension Vehicle,” *Int. J. of Robot. Res*., vol. 3, no. 2, pp. 37-48, 1984.

C. Warring, “What keeps the bicyler upright,” *Popular Science Monthly*, pp. 766-75, 1891. (Republication of 1885 report in Vassar Brothers Institute Transactions, Stable pp.13.)

G. Welch and G. Bishop, “An introduction to the Kalman filter,” Chapel Hill, Tech. Rep., 1995. (Revised, July 2006.)

E. Westervelt, G. Buche, and J. Grizzle, “Inducing dynamically stable walking in an underactuated prototype planar biped,” in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, April 2004, pp. 4234-4239.

E. Westervelt, J. Grizzle, and D. Koditschek, “Hybrid Zero Dynamics of Planar Biped Walkers,” *IEEE Trans. on Automatic Control*, vol. 48, no. 1, pp. 42-56, Jan. 2003.

D. Wilson, “Insect Walking”, *Ann. Rev. Entomology*, vol. 11, pp. 103-122, 1966.

C. Xiyuan, “Modeling Random Gyro Drift by Time Series Neural Networks and by Traditional Method,” *IEEE Int. Conf. Neural Networks & Signal Processing*, pp. 810-813, 2003.

A. Yaari, “Implementing MOOS and IvPHelm on a USV,” *AUVSI Ann. Symp.*, June 11, 2008.

S. Yi, D. Moon, Y. Yang, and K. Kim, “Healthcare Robot Technology Development,” in *Proc. 17th World Congress, International Federation of Automatic Control*, Seoul, July 6-11, 2008, pp. 5318-5323.

J. Yi, D. Song, A. Levandowski, and S. Jayasuriya, “Trajectory tracking and balance stabilization control of autonomous motorcycles,” in *Proc. IEEE Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2583-2589.

S. Yu and U. Neumann, "Fusion of Vision and Gyro Tracking for Robust Augmented Reality Registration," in *Proc. VR Conf.*, 2001.