AN ABSTRACT OF THE THESIS OF

Andrew T. Peekema for the degree of Master of Science in Robotics presented on March 16, 2015.

Title: Template-Based Control of the Bipedal Robot ATRIAS

Abstract approved: _____

Jonathan W. Hurst

This thesis details the derivation and application of template-based controls on a bipedal robot, as well as a description of the software framework that enabled experimentation. The software framework uses a combination of open-source tools including ROS, OROCOS, EtherCAT, and Xenomai to create a real-time environment for the controllers. The first controller makes the robot (ATRIAS) walk as if it was the Spring Loaded Inverted Pendulum (SLIP) model. This demonstrates that ATRIAS, which was designed to be as close to the SLIP model as possible, can behave similarly to the reduced-order model. While the first controller is implemented with a torso that is mechanically fixed in place, the second controller handles an unlocked torso, which adds another degree of freedom. This controller is implemented based on a Torso SLIP (TSLIP) reduced-order model. The controller imposes a Virtual Pivot Point (VPP) by redirecting ground reaction forces to a geometrically fixed point with respect to the torso above the center of mass. Using VPP control, ATRIAS is able to take up to 22 steps on its own without using other techniques to keep its torso upright or inject energy, and it is shown that the generated ground reaction forces are similar to those produced by the reduced-order model. These controllers demonstrate that SLIP-like reduced-order control methods can be implemented on ATRIAS, and are a promising avenue for further research.

Template-Based Control of the Bipedal Robot ATRIAS

by

Andrew T. Peekema

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented March 16, 2015
Commencement June 2015

Master of Science thesis of Andrew T. Peekema presented on March 16, 2015.

APPROVED:

Major Professor, representing Robotics

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Andrew T. Peekema, Author

ACKNOWLEDGEMENTS

I would like to thank everyone at the Dynamic Robotics Laboratory for their help and humor while working on the robot. A special thanks to Dr. Daniel Renjewski for mentoring me in research and writing. And thanks to my advisor, Prof. Jonathan Hurst, for giving me the opportunity and support to work on such a challenging and fun project.

CONTRIBUTION OF AUTHORS

Daniel Renjewski helped generate the overall picture and helped with some of the writing and figures for the ATRIAS Software paper and the SLIP Walking paper. Mikhail Jones helped with experiments and figures for the SLIP Walking paper and TSLIP Walking paper.

TABLE OF CONTENTS

Page

1 Introduction 1

2 Open-Source Real-Time Robot Operation and Control System for Highly Dynamic, Modular Machines 3

2.1 Introduction . . . 3

2.2 System Requirements and Constraints . . . 4

2.3 System Design . . . 5

2.4 System Operation . . . 8

2.5 System Implementation . . . 8

2.5.1 Xenomai . . . 9

2.5.2 OROCOS . . . 9

2.5.3 Controllers . . . 9

2.5.4 Distributed Clock, EtherCAT, and Microcontrollers . . . 10

2.5.5 GUI . . . 12

2.5.6 Logging . . . 12

2.5.7 Emergency Stop Bus . . . 12

2.6 System Evaluation . . . 13

2.6.1 Jitter . . . 14

2.6.2 Emergency Stop . . . 15

2.7 Conclusion and Future Work . . . 15

2.8 Acknowledgments . . . 16

3 Dynamically Appropriate Template Control of a Bipedal Spring Mass Walker 17

3.1 Abstract . . . 17

3.2 Introduction . . . 17

3.3 Simulation . . . 19

3.4 Control . . . 22

3.5 Experimental Setup . . . 23

3.6 Results . . . 24

3.7 Future Work . . . 27

3.8 Conclusions . . . 29

TABLE OF CONTENTS (Continued)

Page

4 Template Based Torso Control On A Spring Mass Robot 30
4.1 Abstract 30
4.2 Introduction 30
4.3 Simulation 32
4.4 Control 36
4.5 Experimental Setup 39
4.6 Results 40
4.7 Future Work 47
4.8 Conclusions 49

5 Conclusion 50

Bibliography 50

LIST OF FIGURES

Figure Page

2.1 The robot (ATRIAS) and graphical user interface. 4

2.2 The robot software structure 10

2.3 Distributed clock timing. 11

2.4 Histograms for jitter at different load conditions. Dashed lines mark the maximum time delay for 99.9% of the samples (also in parenthesis in each figure title). 14

3.1 The ATRIAS robot along with the Spring Loaded Inverted Pendulum (SLIP) template model. The robot is dynamically patterned after the SLIP template by having series elastic legs, distributing its mass such that the center of mass resides close to the point of leg rotation, and minimizing the leg mass that undergoes impact. 19

3.2 The walking limit cycle: vertical leg orientation (VLO) in single support, touchdown (TD) into double support, and takeoff (TO) back to single support. Gait trigger angles for the robot are indicated on their corresponding state figures (q_1, q_2, q_3, q_4), where q_1 is the touchdown angle, q_2 is the stance leg angle at the start of single support, q_3 is the stance leg angle when the flight leg touches down, and q_4 is the leg angle the instant before flight. The plot shows the ideal forces experienced by each leg through one cycle of VLO to VLO for the gait that was experimentally tested. 21

3.3 The measured force profiles of ATRIAS during walking. The center force profile (from 0.2 to 1 sec) is from the left leg, and the rest are from the right leg. The forces are similar to those generated by the ideal SLIP model in simulation (Figure 3.2). 25

3.4 The center diagram of one ATRIAS leg shows the notation for interpreting the graphs to either side. The left graph shows the mean vertical and horizontal ground reaction force component during stance. The right graph shows the axial deviation of the ground reaction force during stance. Standard deviation is indicated on each graph, and the data is taken over five strides of the left leg. 26

LIST OF FIGURES (Continued)

Figure | Page

3.5 The measured trajectory of the left and right toes over many steps compared to the cubic spline with takeoff and touchdown occurring at their SLIP model simulated angles. These trajectories are represented with respect to the robot's center of mass. 28

4.1 ATRIAS, a human-size bipedal robot developed in the Dynamic Robotics Lab at Oregon State University. It was designed based on the SLIP model, with lightweight legs driven by series-elastic actuators and a center of mass near its leg pivot. The robot is attached to a planarizing boom, with onboard computing and offboard power. 32

4.2 Variables that describe the model. The image on the left displays angles, forces, and torques; the image on the right displays distances. This figure also shows the virtual pivot point control concept: that the spring force (F_s) is redirected using hip torque (τ) such that the total ground reaction force (F) points towards the virtual pivot point (the black dot). This virtual pivot point is stationary in the torso reference frame. 34

4.3 The simulated ground reaction forces for a gait using virtual pivot point control. The robot will not try to force its ground reaction forces to be like this; instead, it will use the parameters ($q_{TD}, r_{vpp}, \theta_{vpp}$) that were used to create this gait to control itself. This results in a new, emergent gait that is similar but not identical to the simulated gait. 37

4.4 This compares the simulated (black), desired (pink), and actual (blue) Virtual Pivot Points (VPPs). The black arrow shows the direction the robot is moving, and the origin of the plot is located at the center of mass of the robot. All forces (blue and pink lines) are plotted with respect to the torso. This shows that while the desired VPP is far away from the simulated VPP (0.21m), the sensed VPP is actually much closer to the simulated VPP (0.04m and 0.15m for the left and right legs, respectively). 42

LIST OF FIGURES (Continued)

Figure Page

4.5 This shows the experimental ground reaction forces over two stance phases, as well as snapshots of the robot at particular events (vertical leg orientation, touchdown, and takeoff). This Figure can be directly compared with Figure 4.3, which shows the simulation in a similar manner. This allows for a qualitative comparison between the experiment and simulation, showing robot posture, force profile, gait timing. Overall, the robot qualitatively comes close to matching its ideal template model. 44

4.6 The mean experimental (solid line) and simulated SLIP (dashed line) force profiles of walking. Standard deviations for the experimental data are shown as a shaded region around the solid line, and the statistics have been taken from the right leg over 22 steps. The horizontal force profiles (red) match closely, and the vertical force profiles (blue) match for the first peak but deviate later on. This is due to damping and associated energy injection in the full order model that is not captured in the reduced order model. 45

4.7 The trajectories of the virtual toe (the toe's location given zero spring deflection) with respect to the torso, taken over 22 steps (solid lines), in comparison with the desired trajectories (dashed lines). The trajectories show a consistent leg recirculation that ends close to the desired endpoint of the trajectory, which is what is required of the flight controller. 48

LIST OF TABLES

Table Page

2.1 Comparison of bus systems 6

2.2 Software frameworks 7

2.3 Microcontroller emergency stop conditions: an emergency stop is triggered if any of these signals occur. 13

2.4 Time from initializing an emergency stop until motor command to stop the motors is issued. 15

3.1 Comparison of simulation and experiment. Experimental data is reported for both legs combined and for each leg separately. 27

4.1 Comparison of simulation and experiment statistics. Experimental data is reported for both legs combined and for each leg separately, and calculated over 22 steps. The only quantities that were being controlled in this table were VPP radius and angle; the rest were left to vary based on whatever gait the robot settled into. Given this, the robot displays remarkably similar characteristics through these unregulated parameters. 47

Chapter 1: Introduction

Bipedal robots are currently more lab curiosities than robots seen in the real world. There are several reasons for this, with the main reasons being energy cost and locomotion stability. One method for reducing energy cost is to increase the mechanical efficiency of locomotion, and one way to do this is to cycle some of the gait energy through a purely mechanical device. This energy cycling can be achieved using springs, which can store energy by deflecting, and then release energy by relaxing. However, introducing springs adds unactuated degrees of freedom to the robot, which complicates control.

In order to produce bipedal locomotion, there needs to be a controller that stabilizes the gait. One way to explore, create, and refine controllers is to capture the primary dynamics of the robot in a reduced-order template model, and then design controllers for that template model. This way controllers can be developed in a computationally efficient manner and still be relevant to the full-order robot. Another aspect of the template model is that it can be designed to match forces produced by humans and animals for walking and running, such as the Spring Loaded Inverted Pendulum (SLIP) model. This template model can then serve to guide how the robot should be built, and the controller act, in order for the robot to reproduce the observed human and animal dynamics.

This thesis details the software system, controllers, and tests performed to make a human-scale bipedal robot (ATRIAS) match the reduced-order template model it was designed after (the SLIP model). The first part of this thesis describes how the software system was implemented. This is important because a fast, deterministic control loop is necessary in order to execute fine force control. The second part of this thesis details the control implementation and robot experimentation results of making ATRIAS behave like the SLIP model. The last part of this thesis describes a previously existing reduced-order torso control technique, how it was implemented on ATRIAS, and the experimental results.

OPEN-SOURCE REAL-TIME ROBOT OPERATION AND CONTROL SYSTEM FOR HIGHLY DYNAMIC, MODULAR MACHINES

Andrew Peekema, Daniel Renjewski, and Jonathan Hurst

IDETC/CIE 2013
August 4-7, 2013, Portland, Oregon, USA
ASME 2013 International Design Engineering Technical Conferences & International Conference on Multibody Systems, Nonlinear Dynamics, and Control

Chapter 2: Open-Source Real-Time Robot Operation and Control System for Highly Dynamic, Modular Machines

2.1 Introduction

Dynamic locomotion requires a software and electronic system that can command signals based on sensor data in short time windows. While at first glance this may appear to be a unique application, it is only one in a rapidly growing field that demands hard real-time, high frequency control. Other applications include impedance controlled reaching [8], human-robot interaction [11], dynamic manipulation [7], dynamic image control [33], and multicopter control [30], to name a few.

Given the many applications, it would seem that hard real-time, high frequency control software and electronic structures should be easily available. Commercially available solutions are typically all-in-one bundles that cover electronics and software, however they are quite costly and closed source. Especially - but not only - in scientific projects, the use of commercial systems is constrained by limited budgets and the need to share and reproduce research findings. Open source solutions on the other hand offer free software, run on standard, low cost electronics, and facilitate concept reuse [13]; but they often take a large sacrifice of time and effort to integrate into a functional system. The described design is a combination of mainly open source components with an aim to reduce custom code.

The following is a comprehensive robot control system designed for, but not limited to, the control of a dynamic bipedal robot (ATRIAS, Figure 2.1) [18]. This paper describes the software and electronics - specifically system constraints, final design, and implementation. This system has been successfully deployed on the aforementioned bipedal robot.

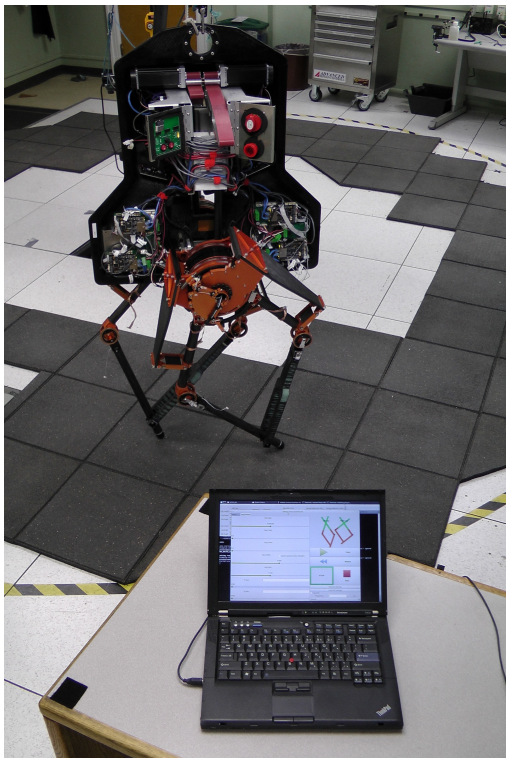

Figure 2.1: The robot (ATRIAS) and graphical user interface.

2.2 System Requirements and Constraints

System architecture requirements can be broken down into three categories: physical, control, and software. The application is a robot that walks around, so the physical focus is mobility. Since the robot will undergo dynamic actions, the control requires determinism at a high execution frequency. Lastly, the software needs to minimize effort, cost, and maintenance, so well-supported open source components were chosen.

Physically, the control computer must be located on the robot and the user interface has to be a remote device. The robot consists of a number of physical modules (e.g. leg, body, boom) that are developed and built in parallel and equipped with different sensors and actuators. The electronics stacks should minimize unique components, support a variety of standard IO-protocols, easily interconnect with each other, allow for independent testing, and easily support the addition of extra sensors and actuators if necessary.

Therefore the electronics stacks must be modular, use a bus system that allows for various topologies, and interface with the control computer using standard electronics. The control software has to be deterministic and execute at a high frequency. Determinism means real-time processing, with no missed cycles and process priority scheduling. The execution frequency goal is 1 kHz, which is fast enough for controllers to use signal derivatives. The control cycle includes taking sensor samples, generating control output, and passing control output to motor amplifiers before the next cycle begins. The controller also must be a software component on the control computer (instead of microcontroller firmware) for ease of testing and debugging. Finally, controller input and output need to be consistent from controller to controller, so they can be easily swapped. The software must be a combination of well supported open source software components to maximize code reuse and minimize development time. Because the bipedal locomotion control concepts [12,21] are not necessarily developed in robot research labs, the software must have a simple controller implementation and interface to increase usability. All of the software needs to be documented, and provide drivers for the chosen physical and wireless communication systems. It is also necessary for the software to have a system dynamics simulator, so that controllers can be validated without hardware risk. Lastly, the software needs to be able to log controller and sensor data.

2.3 System Design

The driving factor for the control system design has been the required communication bus bandwidth and modular topology between the microcontrollers and control computer. Out of a number of available bus systems (Table 2.1), EtherCAT was chosen because it uses standard Ethernet electronics, has flexible network topology, and provides tight device synchronization. In comparison to other bus infrastructures, it can be implemented using standard Ethernet network cards supported by the open-source EtherCAT driver (RT Net) and low cost slave control electronics. For real-time robot control a number of commercial products are available including: LabVIEW, Simulink, dspace, and Gostai RTC. LabVIEW and dspace have dedicated real-time hardware provided by their respective companies. They allow for convenient software development and support many electronic components, but are pricy and cannot be freely shared between developers at different labs. Also - at the time of development - none of these products supported

Table 2.1: Comparison of bus systems

Name	Description
EtherCAT	Ethernet wiring, requires dedicated slave hardware
PROFINET IRT	Ethernet wiring, requires dedicated master and slave hardware
CC-Link IE	Optical Ethernet wiring, ring topology, requires dedicated master and slave hardware
SERCOS III	Ethernet wiring, line/ring topology, requires dedicated slave hardware, optional dedicated master hardware decreases jitter
Powerlink	Ethernet wiring, requires dedicated slave hardware
CAN Bus	Controller Area Network Bus. Fastest standard rate: 1 Mbit/s [45]
Modbus/TCP	Ethernet wiring, requires dedicated slave hardware
Serial Port	Supported by most microcontrollers. Fastest standard rate supported by the XMEGA128: 115,200 Bit/s [1]

EtherCAT. Therefore the control system was designed using standard, off-the-shelf electronics and open source software packages. The developed software is freely available [1]. Given the mobile nature of the robot, a nettop (Mini PC) is the best balance of power consumption, size, and processing power for the control computer on the robot. Another constraint is the network card, which has to be supported by the EtherCAT drivers. The selected nettop is an OEM Production i1000A-i5B1. The machine that runs the user interface can be any computer as long as it can run Linux, has a wireless card, and connect to a monitor, keyboard, and mouse.

Linux is a natural operating system choice because it is open source and all of the robotics software frameworks listed in Table 2.2 support it. Specifically, Xubuntu is the operating system on the robot's control computer, and was chosen because of its large community, variety of supported electronics, and lightweight desktop environment.

A software framework needed to be selected to facilitate wireless communication between the robot and GUI computer. A number of robotic software frameworks exist, and several of them could have been chosen for the non real-time components. ROS was selected because of its large and active community, interoperability with most other frameworks, and encouragement from the grant sponsor.

[1] http://code.google.com/p/atrias/source/checkout

Table 2.2: Software frameworks

Name	Description
ROS	Drivers, libraries, data visualizers, communication system, and package management
YARP	Drivers, libraries, and communication system
Urbi	Drivers, software component generation, scripting language, IDE. Partially closed-source.
Player	Drivers, libraries, and communication system
Rock	Drivers, software component generation (real-time), scripting language (real-time), data visualizers, and communication system (real-time)
OROCOS Toolchain	Software component generation (real-time), scripting language (real-time), and communication system (real-time)

Another required software framework is for generating real-time software components, which enable hard real-time control. OROCOS was chosen as the real-time control environment because of its support for different kernels and ROS communication capability.

In order to achieve real-time in Linux, a real-time kernel is required. The main open source options are: RTAI, RTLinux, RT-Preempt, and Xenomai. RTLinux is no longer developed, and so was not used. RT-Preempt was tested, but was found to have insufficient real-time performance for this application. RTAI was tested next, but the EtherCAT library at the time (EtherLab) was not real-time safe in user space. It was real-time safe in kernel space, but this increased custom software complexity. Xenomai was chosen because of consistent response time under load [5], sufficient real-time performance, EtherCAT support, and OROCOS support.

The system also needs to have an emergency stop (E-Stop), which will cut power to the motors under certain conditions. This needs to be implemented at a low level (firmware on microcontrollers) and be able to override the current program at any time (hardware interrupt). It also needs to be implemented outside main communication channels for redundancy and firmware simplicity.

2.4 System Operation

A user can load a controller on the control computer by clicking a button on the graphical user interface (GUI) running on a remote computer (see the system structure in Figure 2.2). The instruction signal is wirelessly transmitted to the control computer, and received by a non real-time state machine. This program determines if the request is valid (the controller is not already loaded), and if so requests permission to load a controller from a real-time state machine that handles controller input and output. Once granted, the non real-time state machine loads the controller and signals the real-time state machine that the controller loaded. The non real-time state machine then wirelessly signals to the GUI that the controller was successfully loaded. This path is also used by the GUI to stop, start, and reset controllers.

A control cycle starts with a clock pulse sent out from all of the bus microcontrollers ① to the sensor microcontrollers ② (Reference Figure 2.2 to follow ⓗ's in this paragraph). The sensor microcontrollers request cached controller commands from the bus microcontrollers and poll sensors, attaching timestamps to sensor data as it comes in. When all of the data is gathered, the sensor microcontrollers push sensor data to and pull controller commands from the bus microcontrollers. Subsequently, the sensor microcontrollers relay the controller commands to motor amplifiers ③. After a predetermined delay (300 microseconds), the bus program ④ on the control computer requests the cached sensor data from the bus microcontrollers. When the bus program receives the sensor data, it relays it to a real-time state machine ⑤ that handles controller input and output. This in turn relays the sensor data to the controller ⑥. After processing, the controller returns its output to the real-time state machine, where the controller inputs and outputs are sent to a non real-time process to be logged ⑦. The controller outputs are then relayed to the bus program, which sends it out to the bus microcontrollers. The commands wait there until the next clock pulse.

2.5 System Implementation

Recall that Figure 2.2 is a general overview of the system's components and their interconnection. It shows how the user interface, control components, and electronics communicate. OROCOS and EtherCAT handle real-time communication, while ROS

handles non real-time communication. The main components are described in detail in the following sections.

2.5.1 Xenomai

Xenomai provides a real-time environment out of the box with its patch to the Linux kernel and corresponding libraries. A key feature of Xenomai is its ability to determine if a process switches between real-time in Xenomai (primary mode) and non real-time in Linux (secondary mode). This process is called mode switching. The /proc/xenomai/stat file lists all of the tasks within Xenomai and how many times they have mode switched (along with other statistics). Seeing how the mode switch count of a process changes over time is critical for understanding real-time bugs.

2.5.2 OROCOS

The only modification necessary for OROCOS to use Xenomai bindings is to set a shell environment variable. All controller components run within the OROCOS deployer, which allows the code to be kernel-agnostic. Code can be written and tested on a standard Linux kernel, and then verified for real-time performance on a Xenomai kernel. This enables productive code development on a standard Ubuntu kernel with less effort than a full Xenomai install.

2.5.3 Controllers

Controller components are implemented in a hierarchical style that enables code reuse and reduces bugs. For example, a PD controller can be implemented once. Another controller can instantiate several copies of the PD controller and create higher-level functionality - such as a startup controller. This is just one simple example; there is no nesting depth limit.

Controller input and output are standard C++ structs used by all controllers, so controllers can be swapped without the rest of the system having to change. This also standardizes log files, because these two structs are always logged.

Controllers can be tested in place using the Gazebo simulator. In order to accomplish

this, the EtherCAT Connector is replaced by a Simulation Connector. This program acts identically to the EtherCAT Connector as far as the controller is aware, but instead of sending and receiving signals from the robot everything takes place in Gazebo. The Simulation Connector uses RTT-ROS Integration to communicate with a Gazebo plugin, which handles translating the physics simulation into the C++ controller input struct.

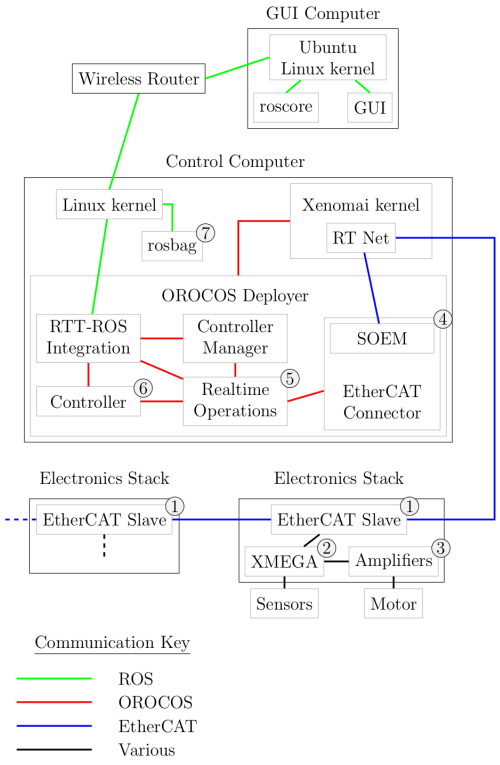

Figure 2.2: The robot software structure

2.5.4 Distributed Clock, EtherCAT, and Microcontrollers

The purpose of the distributed clock is to keep all of the microcontrollers in sync with each other and the control computer. The first distributed clock capable EtherCAT slave is the reference clock for the system. When the EtherCAT master starts, it sends

out a series of packets to determine the time delay between each of the slaves. During run time the EtherCAT master attaches a command to an EtherCAT frame that causes the reference clock to put its time into that frame. All other EtherCAT slaves use that time to synchronize their own clocks, using the delays determined on startup. This is accomplished through several functions in the SOEM (Simple Open EtherCAT Master) library, with RT Net as the EtherCAT driver backend. The synchronization between slaves can be highly accurate; in practice, the difference between slave clocks has been around 30 nanoseconds. To put this in perspective, this is approximately equal to the time taken by one instruction cycle of the XMEGA microcontrollers being used (clocked at 32 MHz).

The EtherCAT slave chips are configured to generate a signal every millisecond, based on the distributed clock time. This signal is sent from the EtherCAT slave chip to a microcontroller pin with a hardware interrupt attached. The microcontroller then to reads sensors (Figure 2.3) and attaches timestamps to data as it comes in. Once complete, the microcontroller writes sensor data to, and read current commands from, the EtherCAT slave chip. The last step the microcontroller takes is to set PWM outputs that signal the amplifiers how much current to command. 300 microseconds after the distributed clock pulse occurs, the EtherCAT master sends out a frame to read the data from the EtherCAT slave chips. The controller is run using this data, and the EtherCAT master writes a frame containing the controller output to the EtherCAT slave chips. This process loops continually until the microcontrollers leave the run state.

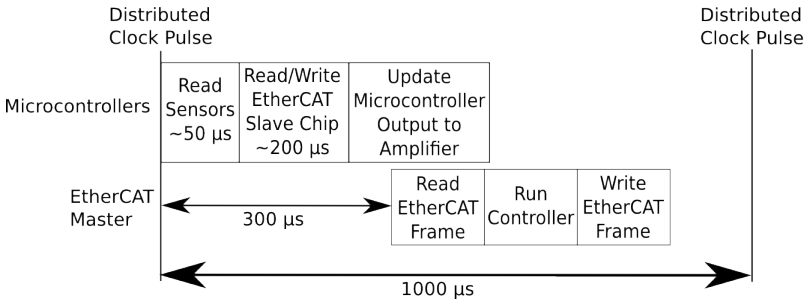

Figure 2.3: Distributed clock timing.

2.5.5 GUI

The GUI is divided up into generic and controller specific portions. The generic section is constantly displayed, and shows data about the robot state (motor positions, current commands, and error states) as well as an interface to start, stop, and reset the currently loaded controller. The controller specific section is displayed only when the controller is loaded, and allows the controller writer to display information from and send information to the robot. The GUI is based on a plugin system written in GTK Builder for the interface, and shared libraries (written in C) for functions that are called by the interface. The GUI computer is connected to the control computer with a wireless router, and uses ROS's publish-subscribe system along with OROCOS's RTT-ROS integration to communicate with the controller and controller manager.

2.5.6 Logging

Rosbag, a datalogging tool provided by ROS, is used to log all GUI and controller data in a binary '.bag' file format. By using the same publish-subscribe system mentioned above, rosbag can easily capture all data over multiple machines as long as the data exists within the ROS network (connected to the same ROS master).
In order to capture data generated within OROCOS, OROCOS's RTT-ROS integration capabilities are used to push controller output, sensor data, system status, and controller-specific messages out of OROCOS and into ROS to be logged by rosbag.
Rxbag, another tool provided by ROS, allows for simple online plotting of numerical data from rosbag log files. A python script exports rosbag files to MATLAB format.

2.5.7 Emergency Stop Bus

The Emergency Stop (E-Stop) bus is a daisy-chained set of wires between the microcontrollers and an external E-Stop switch. The E-Stop line is triggered when the line is pulled low, so if there is a disconnection (ex. the E-Stop switch is pressed) it causes all of the microcontrollers to be sent into E-Stop. Each microcontroller has a dedicated e-stop input connected to a hardware interrupt, which triggers the E-Stop state. It also has a dedicated output line used to assert the bus.

The conditions for an emergency stop are outlined in Table 2.3. When a microcontroller receives any of these signals the E-Stop output line is asserted, and the motors attached to that microcontroller are disabled (The one exception is before a hard stop impact. In this case the motor actively stops its rotation for 100 milliseconds before asserting the E-Stop output line and disabling the motor). When the EtherCAT master sends a reset command the microcontroller de-asserts its E-Stop output line, clears any internal error states, and transitions into the idle state. In the idle state it ignores the E-Stop line - to prevent looping back into E-Stop - and waits for the run command.

Table 2.3: Microcontroller emergency stop conditions: an emergency stop is triggered if any of these signals occur.

Description	Signal
Something is wrong with another microcontroller or the E-Stop wiring	The E-Stop line is low
A motor is going to hit a hard stop	Projected motor position is going to impact a hard stop given motor acceleration limitations
The robot hits a hard stop	A limit switch is pressed
EtherCAT communication is not reliable	5 milliseconds without an EtherCAT packet
The battery is low	Low motor or logic voltage
A motor temperature is too high	Low thermistor resistance

2.6 System Evaluation

A variety of performance tests can be designed to demonstrate the capabilities of software systems. Two tests have been designed to document the timeliness and safety of the system.

2.6.1 Jitter

Jitter is the deviation of the control loop timing from the desired control frequency. Different scenarios were tested: 1) system idle, 2) controller running, 3) high network load and 4) high CPU load. For each control cycle, the desired and effective start time (based on the EtherCAT reference clock) is recorded. Each scenario ran for five minutes, as this is the time range of usual experiments. The results are seen in Figure 2.4. For the purpose of processing sensor data and generating motor commands to control a bipedal robot, the observed performance is acceptable by all means.

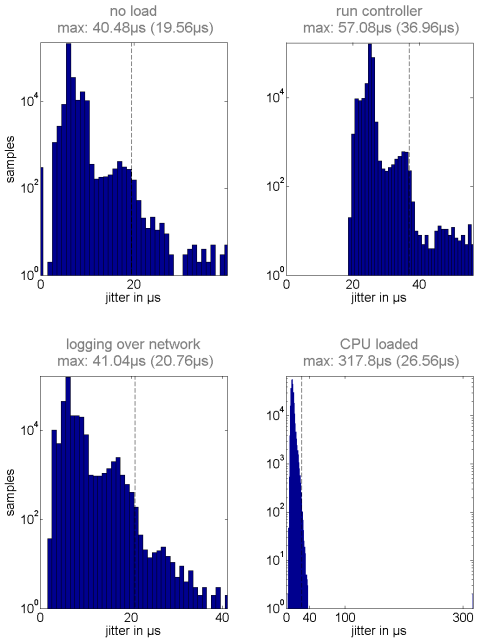

Figure 2.4: Histograms for jitter at different load conditions. Dashed lines mark the maximum time delay for 99.9% of the samples (also in parenthesis in each figure title).

Table 2.4: Time from initializing an emergency stop until motor command to stop the motors is issued.

event	manual	limit switch
duration	0.7μs	3.3μs

2.6.2 Emergency Stop

As safety is an important issue, the system's response to an emergency stop was measured. All processing is done on the slave microcontrollers and the emergency stop signal is distributed in parallel to the EtherCAT bus. For this test, a continuous current signal was sent to the amplifiers. Then, the response time from triggering an e-stop until the amplifier cut the motor current was measured using a digital oscilloscope. Two scenarios were considered: 1) manually pressing an emergency stop button and 2) triggering a limit switch (Table 2.4). The system shuts down well within one control cycle when the e-stop button is pressed. Filtering requirements for the limit switch signals to remove electrical noise leads to a larger but uncritical delay until the motors are shut down.

2.7 Conclusion and Future Work

This paper presents the successful implementation of a low-cost, reliable, and open-source real-time robot control system. The bus system used allows for modular integration of sensors and actuators in flexible topologies. The control system can be reproduced on standard hardware, uses strongly supported open-source components, and requires minimal custom code.

The overall response time of the system can be improved by reducing the time it takes to communicate with the amplifiers. Current work includes designing hardware that allows an amplifier to directly communicate with an EtherCAT slave chip. Also, controller code clarity can be improved with a standard high level state machine, such that logic flow is easily visible. Existing controllers have state machines with varying degrees of readability. Lastly, controller verification can be improved with lock-step simulation communication - the implementation is functional but has detectable response delays.

The system in its current state has been successfully used for experiments in robot walking, and enables highly dynamic robot locomotion.

2.8 Acknowledgments

This project was funded by DARPA grant number W91CRB-11-1-0002 to J. Hurst. The authors would like to thank Kit Morton, Johnathan van Why, Michael Anderson, and Soo-Hyun Yoo for their contributions and feedback.

Chapter 3: Dynamically Appropriate Template Control of a Bipedal Spring Mass Walker

3.1 Abstract

Combining template driven mechanical design with template based control enables dynamic locomotion. This means that the dynamics of a human-sized bipedal robot can be regulated to become close enough to the dynamics of the spring-mass template model such that theories derived using the template model can be implemented on the robot. This paper shows that a human-sized bipedal robot (ATRIAS) can be regulated to act as if it was the template model on which it is based (the Spring Loaded Inverted Pendulum). The data demonstrates that simulation and experiment produce similar force magnitudes, profiles, and duty cycles. This lays the foundation for further template control by tying theory and implementation together.

3.2 Introduction

Walking is such a simple and natural task for humans that we rarely give it any thought as we go about our day. One would suppose that encoding these seemingly effortless motions into a robot would be a simple feat. The reality is quite the opposite. To this day truly versatile, efficient, and dynamic bipedal walking has eluded roboticists.

Current humanoid bipedal robots are not capable of reproducing the locomotion dynamics and energetics observed in animals, and are often not designed with these global task dynamics in mind. Typically, robots are designed to be fully actuated in an effort to create a system that is fully tractable [19, 23, 36]. While this improves controllability it is energetically costly [50]. For systems with unlimited energy, high-bandwidth and high-power actuation this may not pose a problem, but it is much more efficient to utilize the passive dynamics of the system.

On the other end of the actuation spectrum, robots based on passive dynamic walkers use very little or no active control to locomote. Motivated by the simple inverted

pendulum walking model, these machines utilize their mechanical passive dynamics to achieve stable walking [9, 29]. While highly efficient [2], these walkers can only operate in a specific environment with a specific gait, and a lack of control authority becomes detrimental when dealing with disturbances.

The Spring Loaded Inverted Pendulum (SLIP, [3]) is a slightly more complex locomotion template, replacing the rigid legs of the inverted pendulum model with massless springs. This model has been found to better approximate natural locomotion dynamics of various gaits [6, 14, 24]. This also enables a range of locomotion behaviors including hopping, walking [16], running, and walk to run transitions [43]. As a result a number of control theories have been derived to negotiate uneven terrain [4], compliant ground [21], and upper body stabilization [27].

The benefit of this template model is that it simplifies control of steady-state walking by reducing the number of control parameters from a full robot model that includes all links, masses, inertias, spring constants, damping, friction, impacts, etc. down to one mass, two springs, and one controlled variable, i.e. the flight leg touchdown angle [42].

The SLIP model knowledge base can be used to control a robot only if the robot acts like the model. This can be accomplished by intentionally designing the desired passive dynamics of the template model into the robot. In addition, the robot needs to be able to exert active control to compensate for dynamical deviations and energy dissipation.

ATRIAS is a bipedal robot created at the Dynamic Robotics Laboratory to explore energy efficient dynamic walking and running (Figure 3.1). Currently, ATRIAS has demonstrated one-dimensional hopping and two-dimensional walking that utilize its passive dynamics when its torso is held upright [41]. Hybrid Zero Dynamics has been successfully applied to ATRIAS to produce fully unconstrained (three-dimensional) walking [40]. This paper does not aim to showcase a particular controller, but instead show that ATRIAS can match its dynamic template.

The controller described in this paper enforces SLIP dynamics by compliance matching in the leg length direction and minimizing force in the leg angle direction. The data shows that the combination of intentionally designed passive dynamics and control enables the dynamics of the robot to reproduce SLIP dynamics.

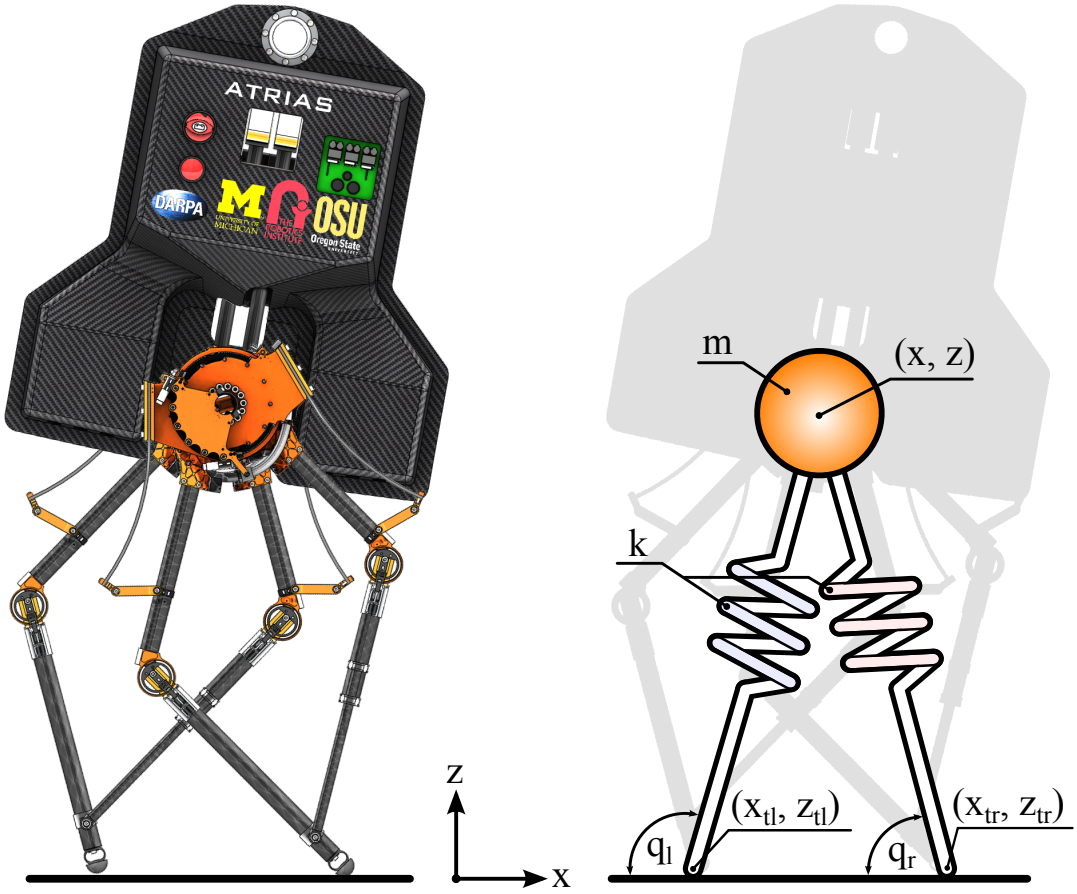

Figure 3.1: The ATRIAS robot along with the Spring Loaded Inverted Pendulum (SLIP) template model. The robot is dynamically patterned after the SLIP template by having series elastic legs, distributing its mass such that the center of mass resides close to the point of leg rotation, and minimizing the leg mass that undergoes impact.

3.3 Simulation

The SLIP model is fully defined by four state variables which are the center of mass position and velocity $[x, z, \dot{x}, \dot{z}]$ and three parameters (mass m, leg length r_0 and spring stiffness k). The control variable to achieve steady state walking is the touchdown angle (q_1).

Equations of motion can be derived using Newtonian methods:

$$m\ddot{x} = F_x,$$
$$m(\ddot{z} + g) = F_z,$$

where F_x and F_z are a combination of spring forces from the legs. If there are two legs on the ground, they both contribute to these forces; if a leg is in flight it applies no force because it is massless. Due to the robot's leg geometry, the spring force is nonlinear with respect to leg deflection. The expected nonlinear force is derived using a static analysis of the ATRIAS leg, and yields the following equation for the leg force in the axial direction:

$$F = \frac{2 * k_s * (acos(r_0) - acos(r))}{(1 - r^2)^{\frac{1}{2}}} \tag{3.1}$$

where k_s is the radial spring constant, r_0 is the rest leg length, and r is the current leg length.

The equations of motion are implemented in MATLAB (2013b, The Mathworks, Natick, MA, USA) and numerically integrated using ode45 (rel. and abs. tolerance 1e-8). The simulation begins at apex with vertical leg orientation (VLO) and a positive forward velocity (Figure 3.2). The initial state and control variable are chosen to match the robot's operation range, and one cycle consisting of single support, double support and single support is simulated. The touchdown angle is optimized using *fminsearch* to yield a limit cycle, i.e. the state where the terminal apex matches the initial conditions. Stance leg angles during phase transitions are recorded as trigger points for the robot controller (Figure 3.2).

The gait simulated for experiments has a VLO state of $x = 0\ m,\ z = 0.870\ m,\ \dot{x} = 0.763\ m/s,\ \dot{z} = 0\ m/s$, control angles of $q_1 = 1.144\ rad,\ q_2 = 1.297\ rad,\ q_3 = 1.845\ rad,\ q_4 = 1.998\ rad$, and a resting leg length of $0.9\ m$. This gait has a duty cycle of 0.59 and a peak vertical force of $630\ N$ (Figure 3.2).

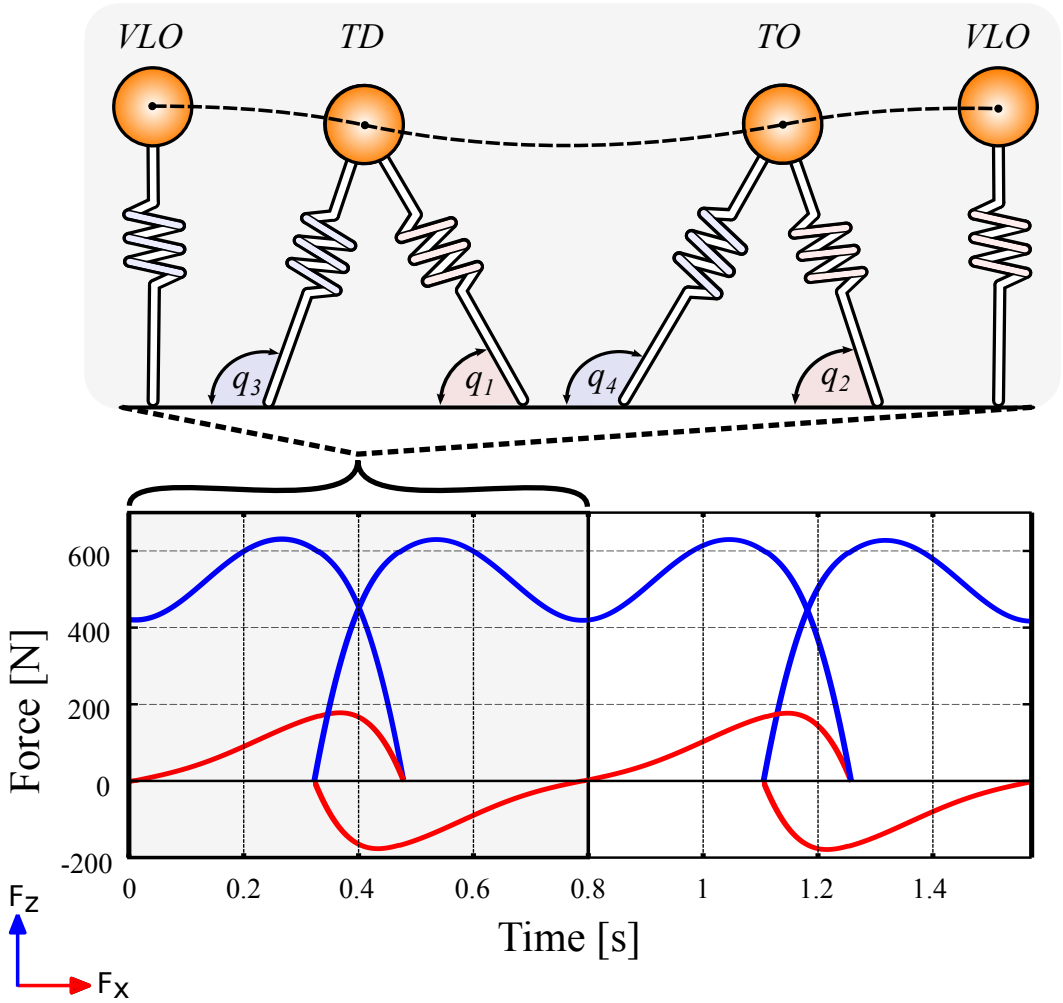

Figure 3.2: The walking limit cycle: vertical leg orientation (VLO) in single support, touchdown (TD) into double support, and takeoff (TO) back to single support. Gait trigger angles for the robot are indicated on their corresponding state figures (q_1, q_2, q_3, q_4), where q_1 is the touchdown angle, q_2 is the stance leg angle at the start of single support, q_3 is the stance leg angle when the flight leg touches down, and q_4 is the leg angle the instant before flight. The plot shows the ideal forces experienced by each leg through one cycle of VLO to VLO for the gait that was experimentally tested.

3.4 Control

While walking, the robot constitutes a hybrid dynamical system. From the perspective of a single leg, there are two dynamically distinct phases: stance and flight. Accordingly, walking control for each leg is broken up into stance and flight.

For stance the SLIP template model dictates zero radial force, and a nominal axial force based on the physical spring stiffness, physical leg configuration, and virtual nominal leg length (Equation 3.1). This force is used as the feed forward term in a force controller, and deviations are compensated with a PD feedback controller. In order to generate the feed forward term, the force controller needs to translate desired end effector forces into joint space. This is accomplished with the Jacobian (see [31] for derivation):

$$\tau = J^T F \tag{3.2}$$

Where F is end effector force, τ is motor torque, and J is the Jacobian. For the case of the ATRIAS leg, the Jacobian takes the following formulation:

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_A} & \frac{\partial x}{\partial q_B} \\ \frac{\partial y}{\partial q_A} & \frac{\partial y}{\partial q_B} \end{bmatrix}$$

Where x, y are the cartesian coordinate equations of the end effector with respect to the base of the kinematic chain, and q_A, q_B are angles that describe the leg geometry. The output of Equation 3.2 gives nominal torques. The current torques are regulated to the nominal torques using a PD controller, and the nominal torques are also added on as feed-forward terms to the output of the PD controller. This completes stance phase control; and while the template model does not require a flight controller as the leg is massless, the robot does need one.

Flight control objectives are to provide ground clearance, fast takeoff, smooth touchdown, and timely leg recirculation considering motor constraints. Two cubic splines were formulated to describe a toe trajectory that satisfies these constraints. One spline is used for leg angle and one for leg length. See Figure 3.5 for the resulting trajectory.

The general formulation for a single cubic spline is described by the following equations:

$$
\begin{aligned}
a_0 &= 2 * (y_1 - y_2) + (\dot{y}_1 + \dot{y}_2) * (x_2 - x_1) \\
a_1 &= y_2 - y_1 - \dot{y}_1 * (x_2 - x_1) - a_0 \\
a_2 &= \dot{y}_1 * (x_2 - x_1) \\
a_3 &= y_1 \\
s &= (x - x_1)/(x_2 - x_1) \\
y &= a_0 * s^3 + a_1 * s^2 + a_2 * s + a_3 \\
\dot{y} &= \dot{x} * \left[-3 * \frac{a_0 * (x - x_1)^2}{(x_1 - x_2)^3} + 2 * \frac{a_1 * (x - x_1)}{(x_1 - x_2)^2} - \frac{a_2}{(x_1 - x_2)} \right]
\end{aligned}
$$

The spline input is $(x, \dot{x})$, where x ranges between x_1 and x_2. The spline starts at state $(y_1, \dot{y}_1)$ and ends at $(y_2, \dot{y}_2)$, with the output of the function being $(y, \dot{y})$. The flight leg's position in the cubic spline (x) is dictated by the stance leg angle.

Switching between flight and stance control is triggered based on stance leg angles derived from simulation (q_1, q_2, q_3, and q_4; see Figure 3.2). Each leg's stance phase is divided into three phases defined by the four angles; first double support, then single support, and lastly double support. The flight phase occurs during the single support of the other leg. This phase timing implicitly dictates the model predicted duty cycle i.e. the ratio of stance time to cycle time.

3.5 Experimental Setup

The ATRIAS robot is roughly human sized, at 1.7 m tall and a mass of 60 kg. It has an on-board computer and carries all required control electronics. The control system runs on the on-board computer at a frequency of 1 kHz (for details, see [34])[1]. The sensor data is captured at the control frequency by the on-board computer for later analysis in MATLAB.

Leg compliance in the ATRIAS robot is enabled by a four bar linkage driven by two series elastic actuators that allow the robot to modulate force magnitude and direction [20]. For these experiments, the radial spring constant of each leaf spring is 1600 N*m/rad. Each leg has an additional degree of freedom in the lateral direction for a

[1]The full system and control code are available here: https://code.google.com/p/atrias/

total of three degrees of freedom per leg.

Since the robot walks in a circle, lateral control is necessary to keep the legs in the robot's sagittal plane. This is governed by PD control of the hip joint to keep the toes constrained to a cylindrical plane centered around the boom pivot. The path around around the boom is covered with 1 cm thick rubber mats to increase friction between the robot's point foot and the ground to reduce slipping.

For the experiments described in this paper, the robot is run using off-board power and its torso is locked from pitching forwards or backwards. It started at rest, and was pushed up to speed by an experimenter. The robot maintained its speed with energy injection provided by external pushing, and was kept that way until the end of the experiment.

3.6 Results

The robot was able to walk successfully in a number of repeated trials. Figure 3.3 shows measured force profiles over one stride. The characteristic double-humped force profile can be observed, though the force progression is not as even as it is in the model. This is most likely due to asymmetries imposed by the boom, where the inside (left) leg progresses more slowly through stance than the outside (right) leg. For this run, the inside leg had an average stance time of 0.79 sec, while the outside leg had an average stance time of 0.69 sec. This asymmetry is inherent when forcing a planar gait onto a plane with curvature, and becomes more pronounced as curvature increases.

An important characteristic of the SLIP model is that it has zero radial force. Figure 3.4 shows the magnitude and direction of ground reaction forces during stance. The right graph shows the deviation of the ground reaction forces from the leg axis during stance. At impact there is a large deviation from nominal (0^o), but during the first force peak this quickly comes close to zero. During the second half of stance the angle stays positive until the end, where the angle deviates largely from nominal. It is important to note that at the beginning and ending of the stance phase where there are the largest angular deviations, the forces are quite small.

When comparing the gait statistics of the simulated results with the statistics from the experimental data (Table 3.1), the data that most closely matches up is the average duty cycle, while the rest of the variables don't match up quite as well. This makes

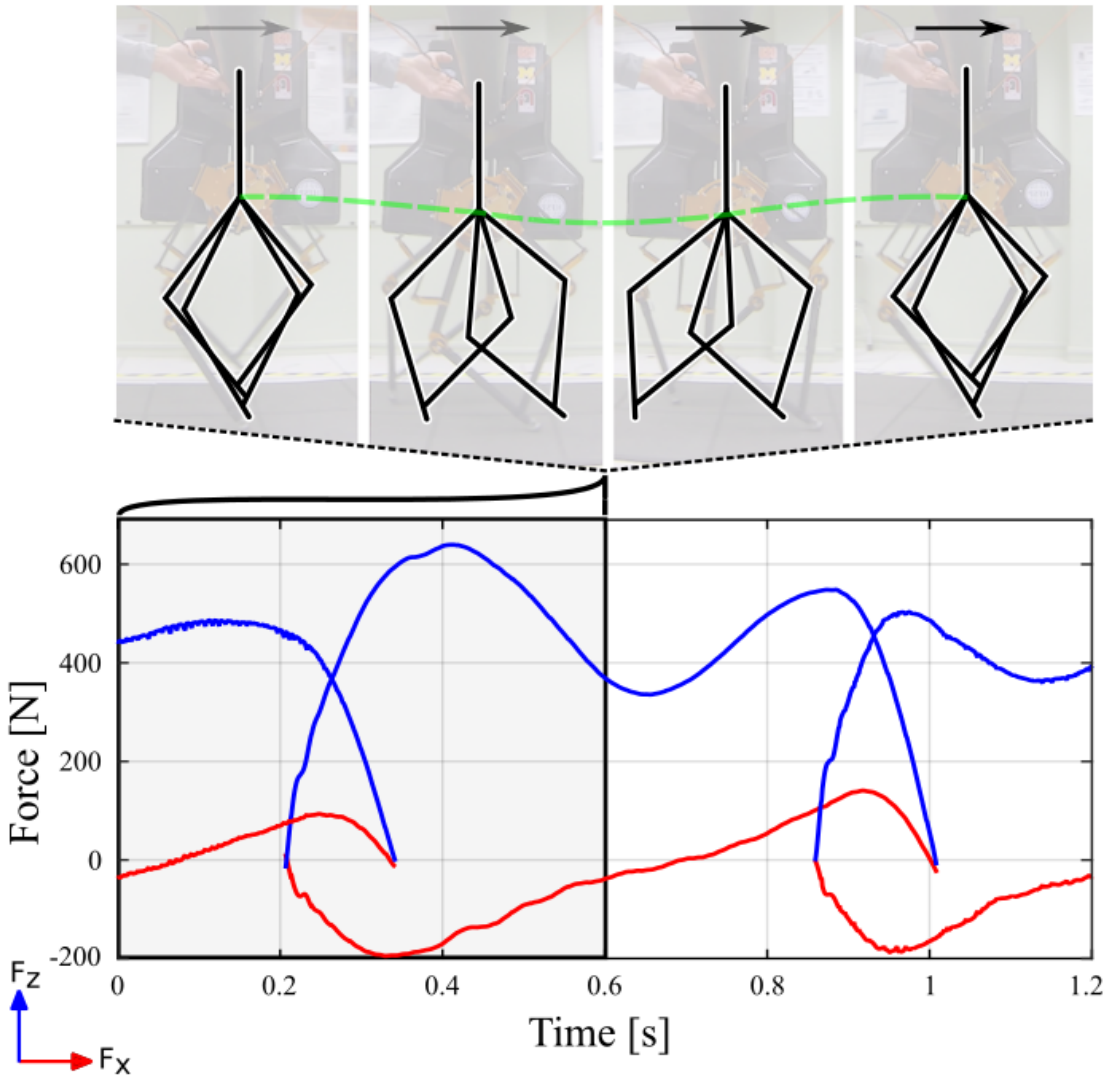

Figure 3.3: The measured force profiles of ATRIAS during walking. The center force profile (from 0.2 to 1 sec) is from the left leg, and the rest are from the right leg. The forces are similar to those generated by the ideal SLIP model in simulation (Figure 3.2).

sense, because the controller indirectly regulates duty cycle (stance time divided by total stride time). This is because the controller dictates what leg angles the robot should be on the ground for, and what leg angles dictate flight. Since the angles enforced by

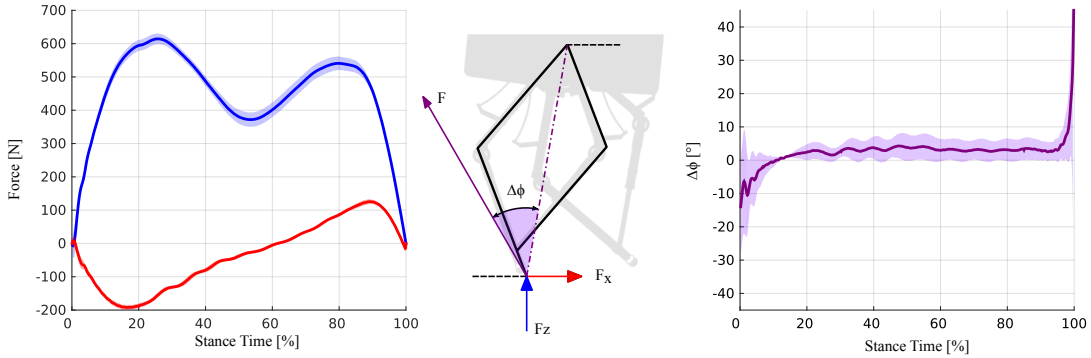

Figure 3.4: The center diagram of one ATRIAS leg shows the notation for interpreting the graphs to either side. The left graph shows the mean vertical and horizontal ground reaction force component during stance. The right graph shows the axial deviation of the ground reaction force during stance. Standard deviation is indicated on each graph, and the data is taken over five strides of the left leg.

the controller are taken directly from simulation, the duty cycle matches quite well in experiment. However, since the rest of the quantities are left unrestrained, the robot is free to follow its own dynamics in terms of peak force, center of mass height, and center of mass speed. Any further correlation between the simulated and experimental statistics is a testament to how close the robot is to having SLIP dynamics.

Another interesting note is that when the statistics are broken up into left and right legs, there is a noticeable asymmetry between the two. This is because the robot is constrained to a boom. The leg furthest away from the boom point of rotation (the right leg) is going faster than the inside (left) leg. Since flight times are determined by the stance angles of the opposite leg, this means that when the left leg is on the ground it will progress through its angles more slowly, giving the right leg a longer time in flight, and vice versa. This is further evidenced in Figure 3.5, where the right toe trajectory tracks better than the left because of the longer flight time.

Figure 3.5 shows the measured toe trajectory in comparison to the cubic spline generated with takeoff and touchdown angles of the simulated SLIP model. The left toe takes off earlier than expected, touches down later than expected, and does not follow the ideal trajectory. The right toe takes off and touches down at the expected positions, and also tracks the trajectory much more closely than the left toe. This is caused by the

		Both Legs			
	Simulation	Mean	Std Dev		
Peak Vertical Force (N)	630	561	58.1		
Duty Cycle	0.590	0.592	0.043		
CoM Height at VLO (m)	0.870	0.889	0.015		
CoM Speed at VLO (m/s)	0.763	0.780	0.060		
		Left Leg		Right Leg	
	Simulation	Mean	Std Dev	Mean	Std Dev
Peak Vertical Force (N)	630	615	15.4	507	18.4
Duty Cycle	0.590	0.633	0.007	0.551	0.007
CoM Height at VLO (m)	0.870	0.902	0.009	0.878	0.010
CoM Speed at VLO (m/s)	0.763	0.737	0.050	0.818	0.038

Table 3.1: Comparison of simulation and experiment. Experimental data is reported for both legs combined and for each leg separately.

asymmetry of the boom constraint, as noted in the previous paragraph.

In summary the robot successfully walked using the proposed control model. The controller did not require a detailed full body model, but instead relied on the force control capabilities and passive dynamics of the robot. The desired ground reaction force profile and duty cycle were well approximated. The mechanical cost of transport over 5 consecutive steps amounts to 0.160, which is impressive for a bipedal robot this size (compared to 1.6 for ASIMO and 0.05 for the Cornell Ranger [9]).

3.7 Future Work

The benefit of implementing the SLIP template model is that theoretical research has already been or is being pursued in the areas of torso control, uneven ground negotiation, and the extension to three dimensions.

Proposed methods for torso stabilization will be tested and implemented. Controllers to regulate the torso include a virtual spring-damper layered on top of the existing compliance controller, as proposed in [44]. A more template-driven approach to torso control lies in the concept of the Virtual Pivot Point, which has been shown to resemble trunk stabilization dynamics observed in humans and animals [28].

Rough terrain can be handled using control derived from a swing leg policy generated

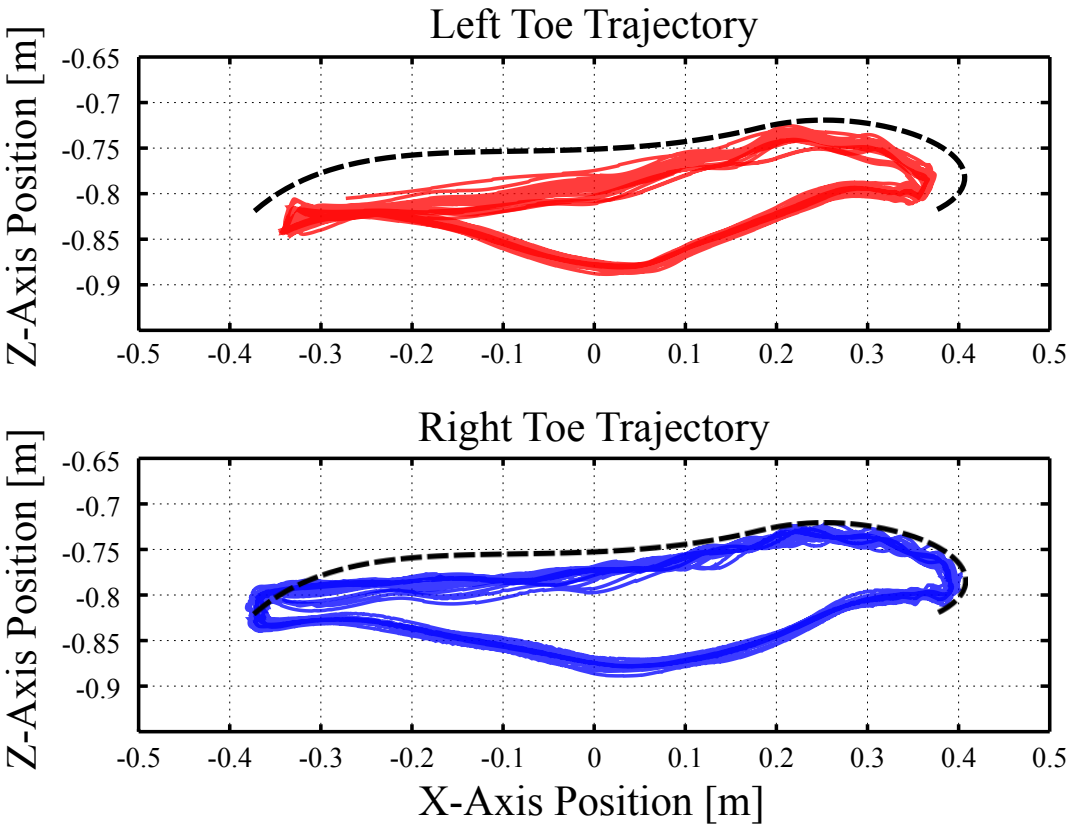

Figure 3.5: The measured trajectory of the left and right toes over many steps compared to the cubic spline with takeoff and touchdown occurring at their SLIP model simulated angles. These trajectories are represented with respect to the robot's center of mass.

by an analysis of the SLIP model according to [48] and [49]. In addition, varying ground impedances can be handled using control derived from the SLIP template in [21], which provides energetically optimal control over ground surfaces with unknown dissipation.

Another major step towards realistic application is the extension of two-dimensional control into three dimensions. Research in this area using the SLIP template model showed that aligning the swing leg with respect to the velocity vector yields stable three-dimensional locomotion and allows for controlled turning [35].

The successful application of these template based control concepts will widely extend

the locomotion capabilities of dynamic bipedal robots, and enable validation of model-based research on neuro-mechanical principles of legged locomotion.

3.8 Conclusions

Template model studies claim that natural dynamics as replicated by the spring mass model reduce control complication and lead to stable and robust locomotion. In this paper it is shown that template based robot design can simplify controller development and reproduce desired dynamics. This combination of machine and controller design proves that control derived from the SLIP template model can be used to control a robot. Developing machines such that they can enforce desired dynamics allow control strategies developed on widely researched fundamental locomotion models to be applied to robots, leading to agile and versatile legged machines.

Chapter 4: Template Based Torso Control On A Spring Mass Robot

4.1 Abstract

Bipedal robots cannot currently walk with the same robustness as humans or animals. One approach for understanding the complex dynamics of locomotion is to observe the ground reaction forces of humans and animals while they hop, walk, and run, and develop a reduced order model that exhibits similar behaviors. One such template model is the Torso Spring Loaded Inverted Pendulum (TSLIP) model when the ground reaction forces are constrained to a Virtual Pivot Point (VPP) above its center of mass. This has been proven in theory, but never applied to a robot. This paper controls the ATRIAS robot, which was designed based on a similar template model, as if it was the TSLIP model with a VPP. The resulting gait produced similar ground reaction forces and VPP locations as the reduced order model, and was able to take up to 22 steps on its own before it fell. This shows that VPP control can be implemented on a robot to produce gaits similar to those observed in the reduced order model.

4.2 Introduction

The reason there is interest in the dynamics of bipedal locomotion is to understand, help, and/or augment how humans locomote, and to build robots that exhibit the same robust locomotion that humans and animals have. The main motivation for developing robots with this capability is to allow them to get around in the same spaces humans do in similar ways, such as climbing stairs, walking around a room, walking through corridors, and being at an appropriate height in order to operate objects in a similar fashion as a human. Because of the high cost of developing such a robot, it would be useful mainly to act as a human surrogate in situations that would be dangerous for humans to be in, such as disaster zones.

The first step towards being able to create a robot that exhibits robust bipedal locomotion is to understand the dynamics of the system, and apply that knowledge

directly to a robot. One approach for understanding the complex dynamics of bipedal locomotion is to observe how humans and animals hop, walk, and run, and develop a simple physics model that exhibits similar behaviors. One of these simple templates is the Spring Loaded Inverted Pendulum (SLIP) model, which consists of a point mass and two massless spring legs. There has been much research on how it matches human and animal locomotion [6, 14, 24], as well as how to control the model to hop, walk, and run [16]. In order to apply the knowledge gained from the SLIP model to bipedal robots that could be useful in a situation like the disaster scenario above, it needs to have a power source, arms (manipulators), and sensors located on some form of upper body.

Techniques to control the torso of a bipedal walking robot tend to fall along a scale that ranges from completely passive to full-body controllers. Completely passive techniques use a mechanism to keep the torso bisecting the leg angles [10,53], or add physical springs to stabilize the torso [17]. On the other end of the spectrum there are full-body controllers, such as Hybrid-Zero Dynamics [52], Sliding Mode Control [47], Zero Moment Point [51], and Capture Point [38]; which all stabilize trajectories that the robot's links move through.

Decreasing in complexity, another approach is to calculate a desired torso angle on the fly, by specifying the torso angle based on a constant added to a linear P feedback gain on forward velocity, and stabilizing to that angle using a PD controller [32, 46]. Decreasing in complexity further, one can also command a fixed torso angle [39].

In between passive techniques and full-body controllers, there are techniques that stabilize the dynamics of a reduced-order model and then apply those techniques to the full-order model. These techniques include the Inverted Pendulum [15, 22, 25] and the Spring Loaded Inverted Pendulum (SLIP) [37], and extensions of these models. One extension of the SLIP model is the Torso SLIP (TSLIP) model, which is comprised of a torso with mass and inertia, attached to two massless spring legs. The TSLIP model has been combined with the Virtual Pivot Point (VPP) method of redirecting ground reaction forces to a fixed point relative to the torso, which stabilizes the torso and generates ground reaction forces that are similar to humans and animals [27].

ATRIAS, seen in Figure 4.1, is a bipedal robot that was designed based on the SLIP model. Specifically, the robot has lightweight legs that are driven by series-elastic actuators, and a center of mass near its leg pivot. It is human-size, weighing in at 60 kg and standing 1.7 m tall. All experiments in this paper are performed on this machine.

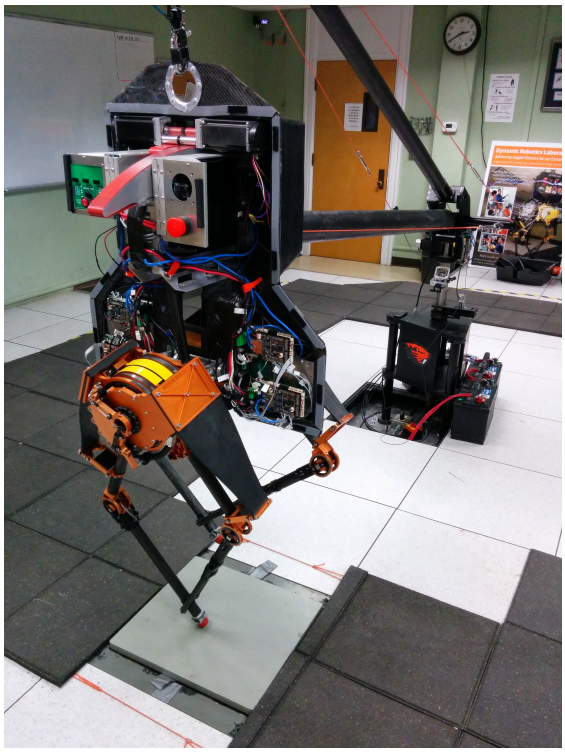

Figure 4.1: ATRIAS, a human-size bipedal robot developed in the Dynamic Robotics Lab at Oregon State University. It was designed based on the SLIP model, with lightweight legs driven by series-elastic actuators and a center of mass near its leg pivot. The robot is attached to a planarizing boom, with onboard computing and offboard power.

The VPP method has never been directly applied to a robot in experiment before, and this is a robot that is mechanically suited to the task. This paper controls the torso of ATRIAS using the VPP method. Simulations of the TSLIP reduced-order model simplification of ATRIAS are performed, control method implementation is discussed, experiments are performed on the robot, and the results are analyzed.

4.3 Simulation

The model state consists of six parameters: the torso center of mass position (x, z) and angle with respect to global coordinates (θ_T) and their derivatives. Other model parameters include the leg angle with respect to the world (θ_L), the angle between the

leg and the virtual pivot point (θ), and the distance between the leg pivot and the center of mass (r_{com}). The control is the virtual pivot point height and angle (r_{vpp}, θ_{vpp}), which indirectly controls hip torque, and the leg touchdown angle (q_{TD}). See Figure 4.2 for a visual representation of these variables. There are other ways to characterize the system, but these variables simplify the system dynamics expressions. A formal definition of the state and control is:

$$X = \begin{bmatrix} x & \dot{x} & z & \dot{z} & \theta_T & \dot{\theta}_T \end{bmatrix}^T$$
$$U = \begin{bmatrix} q_{TD} & r_{vpp} & \theta_{vpp} \end{bmatrix}^T$$

The dynamics can be derived using Lagrangian mechanics. The kinetic and potential energy of the system are

$$KE = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{z}^2 + \frac{1}{2}I\dot{\theta}_T^2$$
$$PE = mgz$$

The Lagrangian with respect to each of the system variables gives

$$\frac{\partial}{\partial t}\left(\frac{\partial KE}{\partial \dot{x}}\right) - \frac{\partial KE}{\partial x} + \frac{\partial PE}{\partial x} + \frac{\partial R}{\partial \dot{x}} = \sum F_x$$
$$\frac{\partial}{\partial t}\left(\frac{\partial KE}{\partial \dot{z}}\right) - \frac{\partial KE}{\partial z} + \frac{\partial PE}{\partial z} + \frac{\partial R}{\partial \dot{z}} = \sum F_z$$
$$\frac{\partial}{\partial t}\left(\frac{\partial KE}{\partial \dot{\theta}_T}\right) - \frac{\partial KE}{\partial \theta_T} + \frac{\partial PE}{\partial \theta_T} + \frac{\partial R}{\partial \dot{\theta}_T} = \sum M_{\theta_T}$$

Combining the previous two equation sets yields

$$m\ddot{x} = F_x$$
$$m(\ddot{z} + g) = F_z$$
$$I\ddot{\theta}_T = r_{vpp}(F_x cos(\theta_T + \theta_{vpp}) + F_z sin(\theta_T + \theta_{vpp}))$$

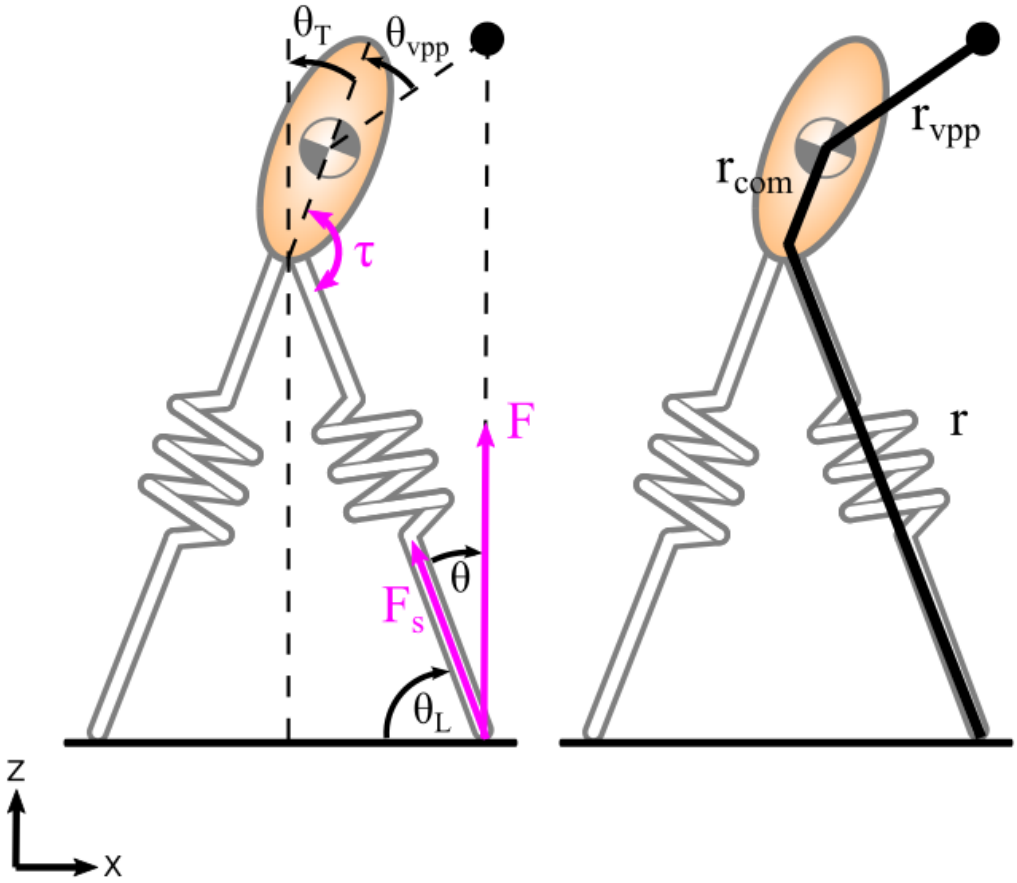

Figure 4.2: Variables that describe the model. The image on the left displays angles, forces, and torques; the image on the right displays distances. This figure also shows the virtual pivot point control concept: that the spring force (F_s) is redirected using hip torque (τ) such that the total ground reaction force (F) points towards the virtual pivot point (the black dot). This virtual pivot point is stationary in the torso reference frame.

r_{vpp} is the distance between the torso's center of mass and virtual pivot point. θ_T is the torso angle with respect to the world. F_x and F_z are a combination of spring force (F_s) and hip torque (τ). The spring force is derived using a static analysis of one

ATRIAS leg:

$$F_s = \frac{2 * k_s * (acos(r_0) - acos(r))}{(1 - r^2)^{\frac{1}{2}}} \tag{4.1}$$

where k_s is the radial spring constant, r_0 is the rest leg length, and r is the current leg length. The reaction force generated at the toe by hip torque (F_τ) is perpendicular to the leg, and is described by $\tau = rF_\tau$. Given that the resultant vector of the spring force and hip force points in the direction of the virtual pivot point, the spring force and hip force are related by $F_s tan(\theta) = F_\tau$. Once all of this is known, F_x and F_z can be solved for using geometry. The force applied by one leg can be expressed by:

$$F_x = -cos(\theta_L + \theta)\sqrt{F_s^2 + F_\tau^2}$$
$$F_z = sin(\theta_L + \theta)\sqrt{F_s^2 + F_\tau^2}$$

Which, when substituted into the equations from the Lagrangian, makes the dynamics during single support:

$$\ddot{x} = \frac{-F_s\ cos(\theta + \theta_L)}{m\ cos(\theta)}$$
$$\ddot{z} = \frac{F_s\ sin(\theta + \theta_L)}{m\ cos(\theta)} - g$$
$$\ddot{\theta}_T = \frac{-r_{vpp}\ F_s\ cos(\theta + \theta_L + \theta_T + \theta_{vpp})}{I\ cos(\theta)}$$

The dynamics during double support can be derived in a similar fashion, except when calculating F_x and F_z there is a duplicate second term that accounts for forces applied by the second leg. Now that all of the dynamic equations have been derived, they need to be numerically integrated.

All simulations are performed in MATLAB (2013b, The Mathworks, Natick, MA, USA), numeric integration is performed using ode45 (rel. and abs. tolerance 1e-8), and optimizations are performed using the *fminsearch* function in MATLAB.

Given the VPP dynamic model, a walking limit cycle needs to be found to test on

the robot. The initial conditions for the simulation started with the known working configuration of the SLIP walking paper (See Section 3.3). Leaving all of those variables as before ($z = 0.990\ m,\ \dot{x} = 0.763\ m/s,\ q_{TD} = 1.144\ rad$), there are also now two more degrees of freedom and two more control inputs: torso pitch (θ_T), torso pitch rate ($\dot{\theta}_T$), VPP radius (r_{vpp}), and VPP angle (θ_{vpp}). The torso pitch is set to zero at VLO in order to reduce the solution space. The remaining degrees of freedom ($\dot{\theta}_T,\ r_{vpp},\ \theta_{vpp}$) are optimized to yield a limit cycle. They become $\dot{\theta}_T = 0.0180\ rad/s,\ r_{vpp} = -0.0083\ m$, and $\theta_{vpp} = 0\ rad$.

While perfectly valid in simulation, this fixed point is undesirable for testing on the robot because the VPP is placed below the center of mass; stable simulation VPPs occur with a r_{vpp} greater than 0.01 m [28], and stable human VPPs occur around a r_{vpp} of 0.05 m to 0.70 m and a θ_{vpp} of 0 rad [26]. Increasing r_{vpp} can be accomplished by increasing the forward speed of the simulation - a 0.9 m/s forward speed was chosen because it had already been demonstrated in locked torso hardware tests on the robot, and that fixed point became the simulation baseline and the starting point for experiments: $x = 0\ m,\ z = 0.990\ m,\ q_T = 0\ rad,\ \dot{x} = 0.9\ m/s,\ \dot{z} = 0\ m/s,\ \dot{q}_T = -0.45\ rad/s$, control parameters of $q_{TD} = 1.144\ rad,\ r_{vpp} = 0.198\ m,\ \theta_{vpp} = 0\ rad$, and a resting leg length of 0.9 m. The resulting ground reaction forces can be seen in 4.3. How the control parameters relate to the model can be seen in Figures 4.2 and 4.3.

4.4 Control

The robot control consists of two main dynamic regimes: stance control and flight control. During stance control, the robot regulates the axial leg force to the force it would apply given a virtual constraint of a fixed leg length (Equation 4.1). It also regulates hip torque to redirect the nominal axial leg force to pass through the virtual pivot point. The torque to redirect the axial spring force was derived in Section 4.3, and is:

$$\tau = rF_s tan(\theta)$$

In order to translate the forces from end effector space to motor space, a Jacobian is used [31]:

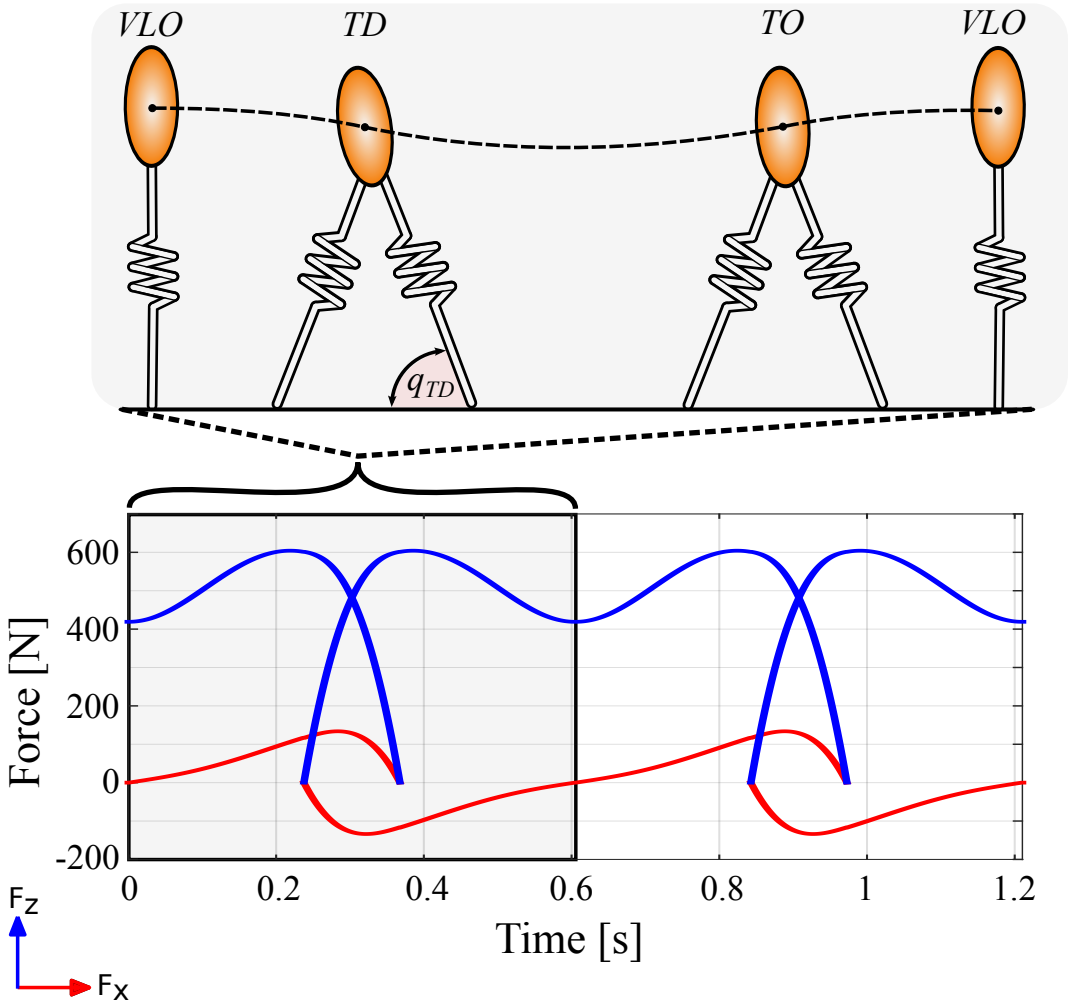

Figure 4.3: The simulated ground reaction forces for a gait using virtual pivot point control. The robot will not try to force its ground reaction forces to be like this; instead, it will use the parameters $(q_{TD}, r_{vpp}, \theta_{vpp})$ that were used to create this gait to control itself. This results in a new, emergent gait that is similar but not identical to the simulated gait.

$$\tau = J^T F \tag{4.2}$$

Where F is the force in the end effector space, J is the Jacobian, and τ is motor

torque. The Jacobian for the ATRIAS leg is:

$$J = \begin{matrix} \frac{\partial x}{\partial q_A} & \frac{\partial x}{\partial q_B} \\ \frac{\partial y}{\partial q_A} & \frac{\partial y}{\partial q_B} \end{matrix}$$

Where x, y are the cartesian coordinates of the end effector, and q_A, q_B are the motor coordinates. Now that we have the nominal torques in motor space they are used as feed-forward terms, and PD control is used to drive the current force to the nominal force. Now that stance control is complete, there needs to be a policy to recircluate the legs during the swing phase of the controller.

During swing control, the motors track a trajectory given by two cubic splines that dictate the position of the virtual toe with respect to the world. "Virtual" indicates that it is not the true position of the toe, but what it would be if there were no springs in the leg. The true toe position is not controlled directly, because while having soft springs in series with the motors makes force control possible, it makes position control of anything past the springs unrealistic.

Cubic splines define the virtual leg length and leg angle. The spline formulation used for this controller is:

$$\begin{aligned}
a_0 &= 2 * (y_1 - y_2) + (\dot{y}_1 + \dot{y}_2) * (x_2 - x_1) \\
a_1 &= y_2 - y_1 - \dot{y}_1 * (x_2 - x_1) - a_0 \\
a_2 &= \dot{y}_1 * (x_2 - x_1) \\
a_3 &= y_1 \\
s &= (x - x_1)/(x_2 - x_1) \\
y &= a_0 * s^3 + a_1 * s^2 + a_2 * s + a_3 \\
\dot{y} &= \dot{x} * \ -3 * \frac{a_0 * (x - x_1)^2}{(x_1 - x_2)^3} + 2 * \frac{a_1 * (x - x_1)}{(x_1 - x_2)^2} - \frac{a_2}{(x_1 - x_2)}
\end{aligned}$$

Where the swing leg's progression through the spline $(y, \dot{y})$ is dictated by the stance leg angle (x). The stance leg angle ranges between when the flight leg takes off (x_1), and an angle that is experimentally found to timely recirculate the leg (x_2). The flight leg angle ranges between the state it was in when it took off $(y_1, \dot{y}_1)$, and the simulated touchdown angle $(y_2 = \theta_{TD}, \dot{y}_2 = -0.1)$. The constant negative final velocity is tuned to

start retracting the leg before touchdown in order to smooth the state transition.

A similar spline is used to determine the swing leg length, except it is broken into two pieces: one for retraction, and one for extension. The retraction spline ranges between the state it was in when it took off $(y_1, \dot{y}_1)$, and a retraction length halfway through stance $(y_2, \dot{y}_2)$. The extension spline is then used, starting at the retraction state $(y_1, \dot{y}_1)$ and finishing at the resting leg length $(y_2 = r_0, \dot{y}_2 = 0)$. This combination of splines for the leg angle and leg length generate a trajectory that timely recirculates the leg. Now that stance and flight controllers have been developed, there needs to be a policy for when to switch between them.

A state machine controls if the legs are in stance or flight. The touchdown switch occurs based on when a load cell in the toe crosses an experimentally determined threshold force. The takeoff switch occurs based on when spring force in the leg crosses zero. Spring force cannot be used for touchdown as well because the force continually oscillates during swing (it is a spring-mass system with very little damping). Since the load cell is physically located past this oscillating force, it can detect touchdown before leg force.

While the simulation is an energetically conservative system, the robot is not. Energy can be injected into the system based on the location of the virtual pivot point; a virtual pivot point that is tilted backwards will continually inject energy [28]. This strategy provides an energy injection policy that does not create conflicting control objectives. The virtual pivot point tilt is experimentally determined in order to develop sustained walking.

4.5 Experimental Setup

All experiments were carried out on ATRIAS: a bipedal, spring-mass, human-size robot. ATRIAS weighs about 60 kg, is about 1.7 m tall, and is attached to a planarizing boom that is about 2 m in radius (Figure 4.1). All of the power used on the robot is located offboard in 3 lead-acid car batteries connected in series.

Robot control is done at a frequency of 1kHz using a combination of open-source Linux components. All control computation is done on a nettop onboard the robot, and is wirelessly connected to a graphical user interface located on a laptop separate from the robot (see [34] for details)[1].

[1]The full system and control code are available here: https://code.google.com/p/atrias/

Force control on ATRIAS is achieved mechanically via a four bar linkage leg connected to two series elastic actuators [20]. The springs in these series elastic actuators have a spring constant of 1600 N*m/rad. The legs on ATRIAS also have another degree of freedom laterally; in total, each leg has three degrees of freedom.

The boom constrains a point on ATRIAS' torso to the surface of a sphere, which is a reasonable estimate of planar walking. In order to better approximate planar walking, the lateral degree of freedom of each of ATRIAS' legs is constrained to a cylinder centered around the boom's base. Rubber mats are placed around the boom to increase friction between the ground and the toes of the robot in order to reduce slipping.

For the gaits in this paper, the robot has a torso that is free to pitch forwards and backwards. It starts at rest, is pushed up to speed by an experimenter, and is let go to continue walking without external influence.

4.6 Results

The control inputs derived from simulation are touchdown angle (q_{TD}) and the Virtual Pivot Point position (r_{vpp}, θ_{vpp}). During experiments on the robot, the touchdown angle did not need to be modified; however, the target Virtual Pivot Point did due to imperfect force tracking. The target Virtual Pivot Point had to be higher and in front of the robot ($r_{vpp} = 0.4m, \theta_{vpp} = 0.25rad$) in order to make the true forces create a pivot point that would generate walking for a number of steps.

Out of 19 walking tests, the median number of steps the robot was able to take under its own power was 8, the minimum was 4, and the maximum was 22. The common failure mode was the torso falling forwards or backwards as the robot's energy became too high or low, because there was no position control keeping the torso upright and no energy regulation. This lack of regulation may be surprising and counter-intuitive, however it was done intentionally so there would be fewer variables to account for when trying to observe the influence of the reduced order model control when implemented on the robot. That it was able to walk at all shows that the robot behaves somewhat like the reduced order model. The main question that remains is: how close did it get?

The best experiment for showing how the dynamics of the robot and controller behave is the test where the robot took 22 steps, because this is the most steady-state gait that was achieved. This is the dataset that is analyzed for the rest of this section. The gait

metrics that will be shown are the location of the Virtual Pivot Point, the magnitude and direction of the ground reaction forces, the duty cycle of the gait, the center of mass height of the robot, and the center of mass speed of the robot. All of these metrics will be compared between the reduced-order model simulation and experimental data. The first metric, the Virtual Pivot Point, shows how the forces were applied to the torso.

The accuracy of the created Virtual Pivot Point (VPP) during walking is shown by Figure 4.4. The black arrow indicates the direction of motion of the robot, and the blue and pink lines indicated the sensed and desired forces, respectively. All forces are plotted with respect to the torso, with the origin located at the center of mass of the robot.

The experimental VPPs were calculated by finding the shortest distance between a point and all force vectors, scaling those distances by the magnitude of the force being applied, and taking the sum of squares of those values. This value was then minimized using fminsearch in MATLAB (2013b, The Mathworks, Natick, MA, USA), with input and function tolerance at 1e-18, to find the VPP.

The black dot shows the VPP that generates a limit cycle in simulation. This is close to the VPP that was generated by the forces applied by the controller (0.04 m and 0.15 m away for the left and right legs, respectively), represented by the blue dot. The pink dot that shows the desired VPP is the further away still (0.21 m away from the simulated VPP), tilted forward and positioned higher than the generated and simulated VPP. It is positioned like this because when the leg touches down, the initial conditions drive the force away from the VPP. Over the duration of the gait, the force gradually points more accurately at the VPP. Overall, this results in a force that lags behind the desired VPP initially, and catches up to the desired VPP over the duration of the gait.

The fact that the applied forces (and resulting VPPs) are asymmetrical between the left and right legs is consistent with the rest of the data. This is because the planarizing boom that the robot is attached to does not perfectly represent planar walking - and in fact has a relatively large impact on experimentation. This generally makes finding a gait that is suitable for the robot a balance between getting one or the other leg closer to a stable gait, with the best resultant gait being the best compromise. This asymmetry can also be seen in the plot of the ground reaction forces.

The ground reaction forces show the movement of the robot from a perspective that is abstracted away from any one particular robot configuration and controller, by showing how forces are applied to the robot in order to produce motion. In Figure 4.5, the walking

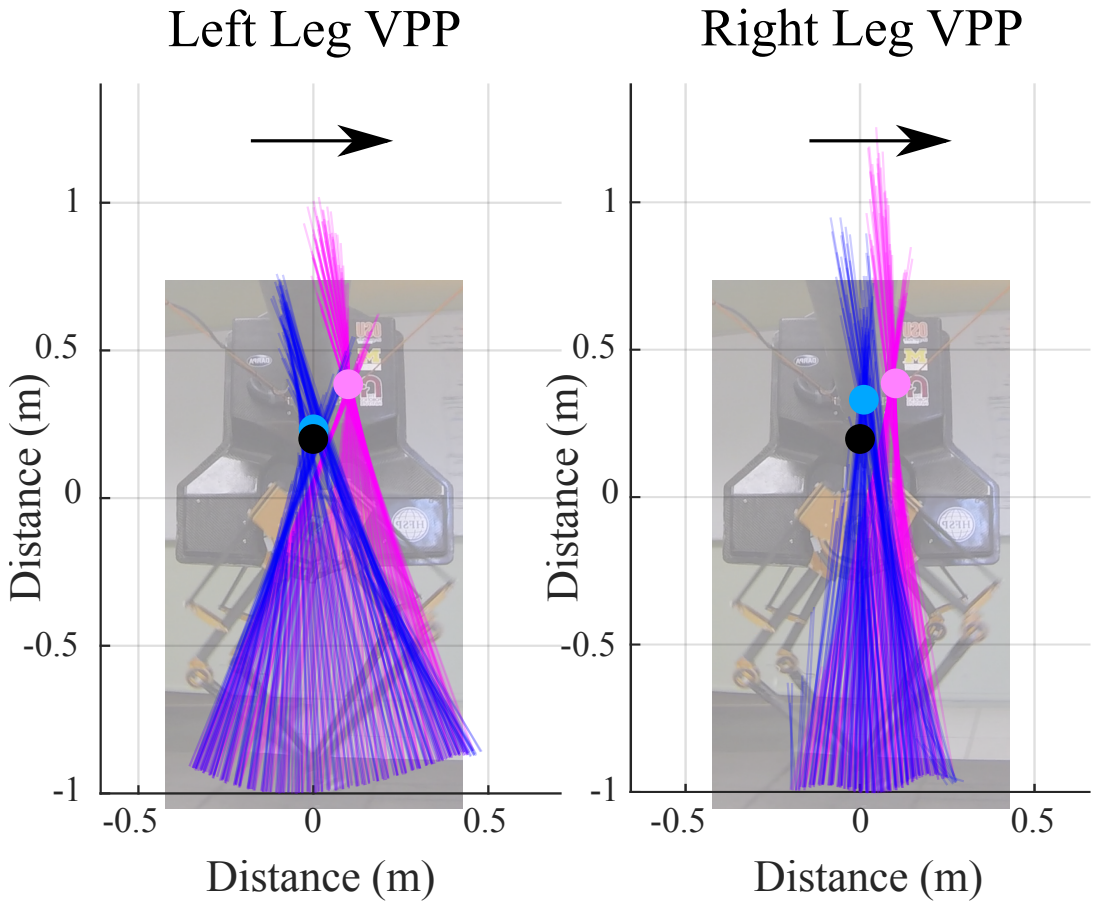

Figure 4.4: This compares the simulated (black), desired (pink), and actual (blue) Virtual Pivot Points (VPPs). The black arrow shows the direction the robot is moving, and the origin of the plot is located at the center of mass of the robot. All forces (blue and pink lines) are plotted with respect to the torso. This shows that while the desired VPP is far away from the simulated VPP (0.21m), the sensed VPP is actually much closer to the simulated VPP (0.04m and 0.15m for the left and right legs, respectively).

ground reaction forces are presented in a similar manner as the reduced order model, where the robot can be seen in the top panel of the figure, and the ground reaction forces

can be seen in the lower part of the figure. This puts the data in context with the robot's motion, shows how quickly the forces progress over time, and how much time is spent in double support (with two legs on the ground). This figure shows two stance phase duration forces and snapshot images. When compared with Figure 4.3 several similarities can be drawn. At Vertical Leg Orientation (VLO), the torso is vertical and the forces are (almost) at a minimum, indicating that the robot has no vertical velocity since the leg has a fixed resting length and the spring is the least compressed. At touchdown (TD), the images show that the robot is leaned back and ready for the impact that will rotate the torso forward. At takeoff (TO), the torso is slightly leaned backwards, and at VLO, the torso is vertical once again. The duration of these two steps is of similar scale in simulation and on the robot (1.2 and 1.1 seconds respectively), with double support taking slightly longer and single support slightly less time in experiment when compared to simulation. The largest difference between these two force profile sets is that while the first peak of the vertical force is almost identical between simulation and experiment, the second peak is not nearly as large. This can be explained by an off-center VPP. When the the VPP is centered, as in the simulation, the resulting force profile is symmetric and no energy injection occurs. However, since the real system has damping, the VPP is placed off-center, injects energy, and results in an asymmetric force profile.

In simulation, the robot walked with force profiles seen in Figure 4.3. In experiment, the horizontal component of the force profile matched reasonably well while the vertical force profile was asymmetric (Figure 4.6). This Figure shows the mean of the force profiles of the right leg over all of the steps, and directly compares this with simulation. The main difference of note is the asymmetry between the first and second maximum forces, which was discussed in the previous paragraphs, but to recap this is due to the fact that the experimental system has damping and needs to inject energy in order to keep walking. A centered VPP has symmetric force profiles, and an off-center VPP has asymmetric force profiles that inject energy into the gait.

The duty cycle of the gait shows how quickly the ground reaction force impulse is delivered to the robot (a duty cycle less than 0.5 indicates running). The center of mass speed and height of the robot give a physical metric that can be easily compared to the reduced order model, showing how close the robot came to matching the state of the reduced order model simulation over several steps (Table 4.1). This shows that the mean of the peak force for both legs came within one standard deviation of the simulation's

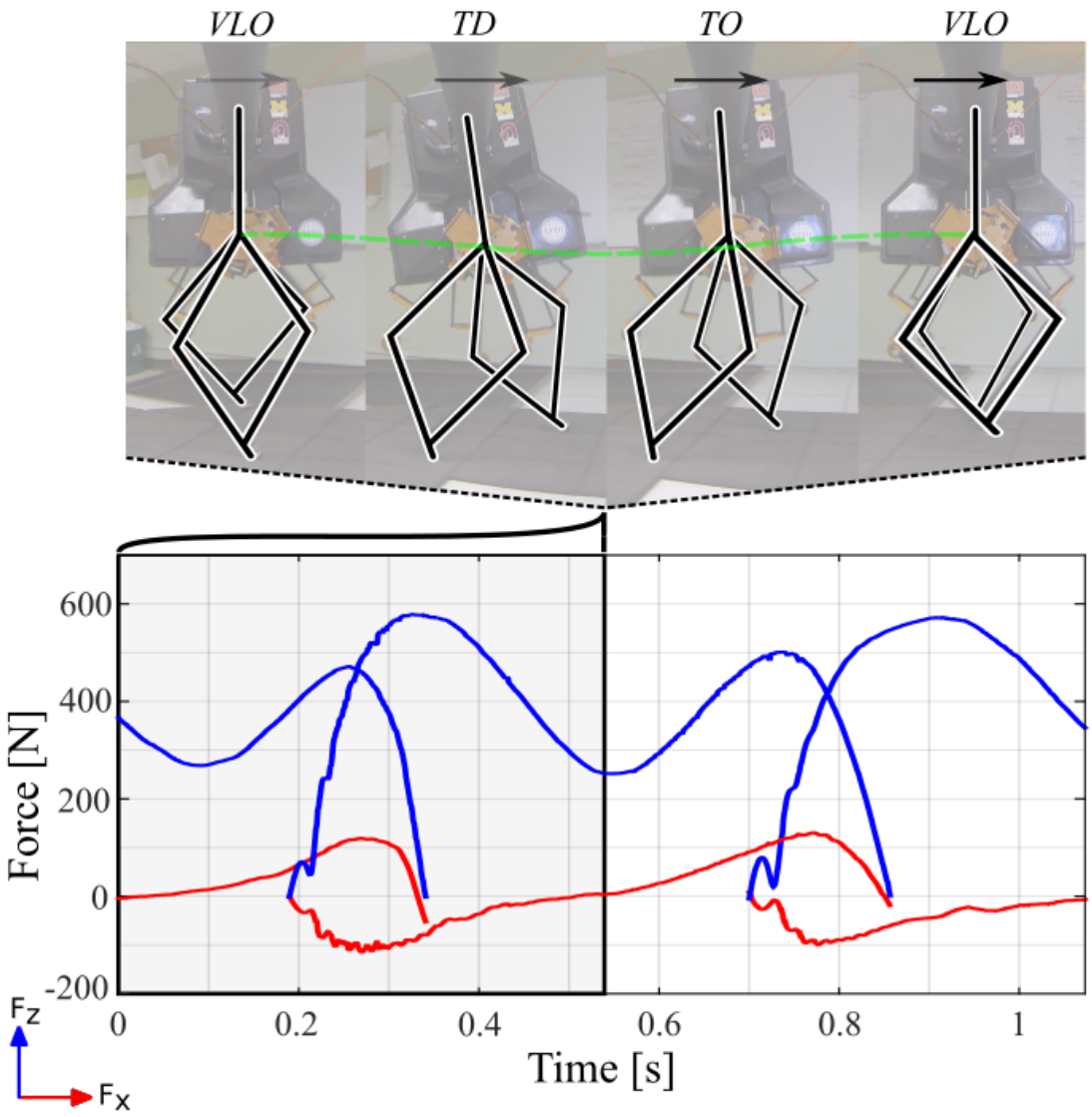

Figure 4.5: This shows the experimental ground reaction forces over two stance phases, as well as snapshots of the robot at particular events (vertical leg orientation, touchdown, and takeoff). This Figure can be directly compared with Figure 4.3, which shows the simulation in a similar manner. This allows for a qualitative comparison between the experiment and simulation, showing robot posture, force profile, gait timing. Overall, the robot qualitatively comes close to matching its ideal template model.

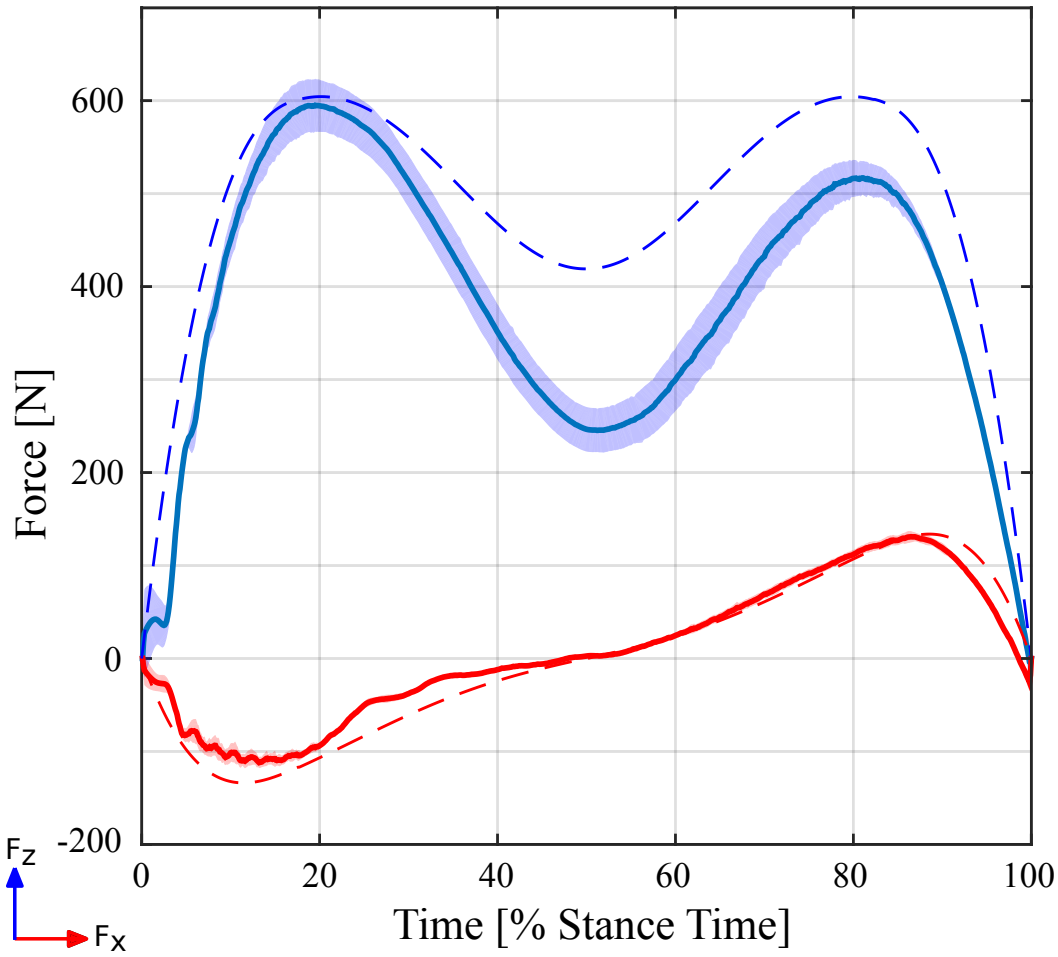

Figure 4.6: The mean experimental (solid line) and simulated SLIP (dashed line) force profiles of walking. Standard deviations for the experimental data are shown as a shaded region around the solid line, and the statistics have been taken from the right leg over 22 steps. The horizontal force profiles (red) match closely, and the vertical force profiles (blue) match for the first peak but deviate later on. This is due to damping and associated energy injection in the full order model that is not captured in the reduced order model.

peak force. This is expected, because the mass of the robot should be similar to the mass of the simulation, and if the dynamics are reasonably similar these should match

up. The duty cycle of the simulation was outside two standard deviations from the mean of the experiment. This can be explained because the state transitions between stance and flight was not regulated, so the dynamics of the robot gait fell into something close but not quite the same as the reduced order model. This is the same for the the center of mass height at Vertical Leg Orientation (VLO). The center of mass speed at VLO was not regulated either, but because of its large standard deviation the simulation's speed does fall inside one standard deviation. The Virtual Pivot Point (VPP) radius falls within two standard deviations of the simulation, and the VPP angle falls within one standard deviation. The asymmetry of the gait due to the boom can be most clearly seen when the standard deviations of both legs are compared with the standard deviations of the individual legs. Every single statistic except one has a smaller standard deviation when broken out by leg than when combined. This is mostly a function of the radius of the boom - if the robot had a longer boom, then this asymmetry would not be as noticeable. However, given these constraints, the robot matched the simulation well enough to walk without the regulation of these parameters.

Flight phase control performance is dictated by how well the motors track their trajectories. This can be displayed in terms of the position of the virtual toe (where the toe would be if there was no spring deflection) with respect to the torso. The desired and true trajectories for 22 steps can be seen in Figure 4.7. Given that the robot did not regulate its stride length, the toe trajectories were remarkably consistent from step to step. The toes did not follow their desired trajectories perfectly, however this was never a critical metric. What was required is a consistent, timely recirculation of the legs that touches down at a particular location with respect to the torso, which this does.

The main differences between the experiment and the reduced order model were that the virtual pivot point (VPP) was higher on the robot (Figure 4.4), and the force profile of the experiment was asymmetrical while the simulation force profile was symmetrical (Figure 4.6). The first can be explained because asymmetries in the gait caused by the boom forced the robot to find a new stable VPP, which was slightly higher than the simulated VPP. The asymmetrical force profile is due to a tilted VPP. When the VPP is directly above the center of mass the resulting force profile is symmetric. However when the VPP is moved off center, an asymmetric force profile results that injects energy into the gait. This is necessary for the energetically lossy robot, but not the reduced order model, hence the difference.

		Both Legs			
	Simulation	Mean	Std Dev		
Peak Vertical Force (N)	604	583	26.7		
Duty Cycle	0.607	0.587	0.007		
CoM Height at VLO (m)	0.990	0.899	0.014		
CoM Speed at VLO (m/s)	0.900	0.911	0.148		
VPP Radius (m)	0.198	0.285	0.057		
VPP Angle (deg)	0.000	0.010	0.023		
		Left Leg		Right Leg	
	Simulation	Mean	Std Dev	Mean	Std Dev
Peak Vertical Force (N)	604	569	15.4	596	28.6
Duty Cycle	0.607	0.583	0.005	0.591	0.005
CoM Height at VLO (m)	0.990	0.908	0.009	0.889	0.013
CoM Speed at VLO (m/s)	0.900	0.804	0.014	1.030	0.135
VPP Radius (m)	0.198	0.233	0.004	0.343	0.008
VPP Angle (deg)	0.000	-0.010	0.004	0.033	0.003

Table 4.1: Comparison of simulation and experiment statistics. Experimental data is reported for both legs combined and for each leg separately, and calculated over 22 steps. The only quantities that were being controlled in this table were VPP radius and angle; the rest were left to vary based on whatever gait the robot settled into. Given this, the robot displays remarkably similar characteristics through these unregulated parameters.

Overall, the robot and controller acted similarly to the reduced order model. This is shown by the similar force profiles in Figure 4.6, similar gait timing shown in Figure 4.5, and a similar VPP shown in Figure 4.4.

4.7 Future Work

Progress that can improve upon this work can focus on improved force control in order to bring the applied force closer to the desired force. This would allow for a more accurate Virtual Pivot Point (VPP), and less tuning would be needed in order to work a gait onto the robot. Force control can be improved in hardware and software. The hardware can be upgraded by using better amplifiers, which will reduce the delay between commanding a current and having it be applied. It can also be improved in software, by taking more of the dynamics into account than a single feed-forward term with a technique like feedback

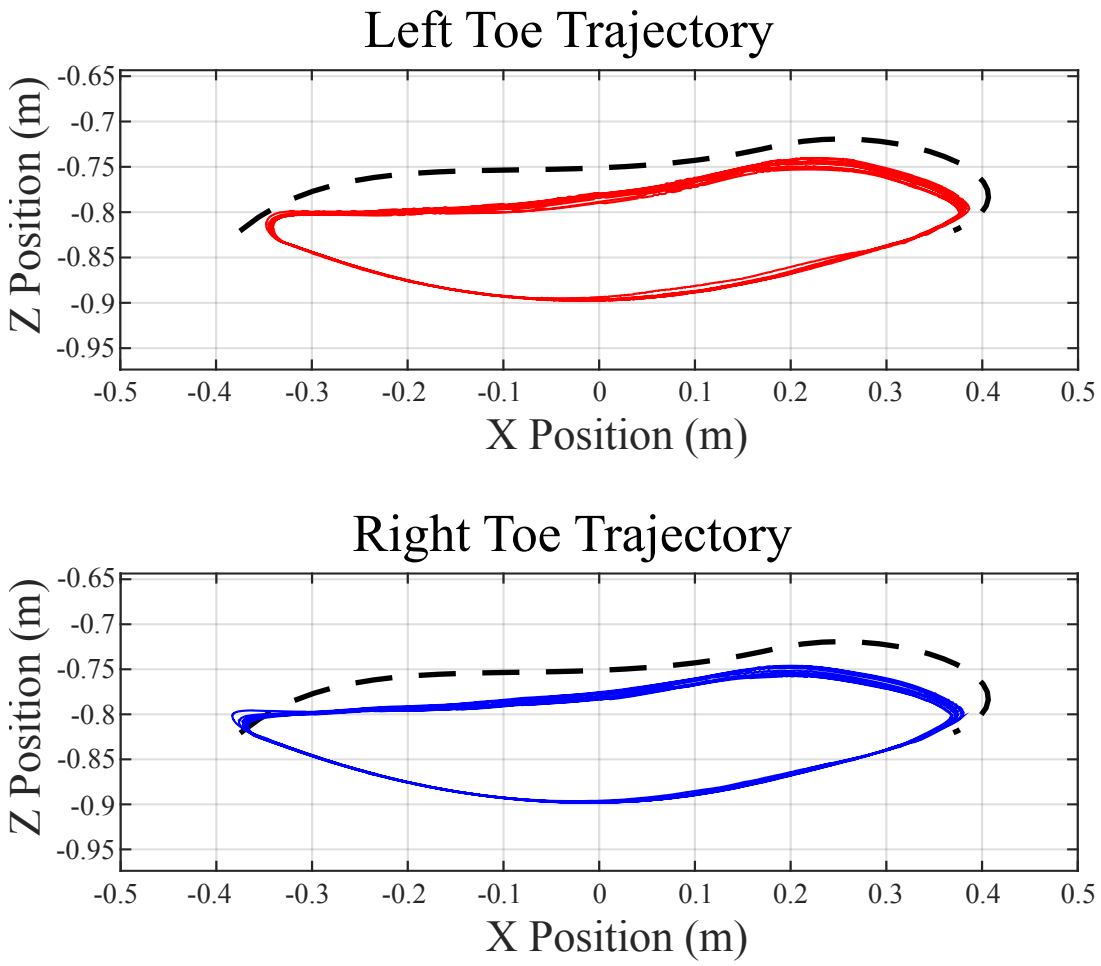

Figure 4.7: The trajectories of the virtual toe (the toe's location given zero spring deflection) with respect to the torso, taken over 22 steps (solid lines), in comparison with the desired trajectories (dashed lines). The trajectories show a consistent leg recirculation that ends close to the desired endpoint of the trajectory, which is what is required of the flight controller.

linearization.

On a larger scope, the goal is to achieve stable walking with these techniques. In order to accomplish this, explicit energy and torso regulation needs to be added so that

the gait can be stabilized and the robot continue to walk past the current average of 8 steps. Torso stabilization can either be accomplished by moving the VPP around in response to disturbances, or by adding a loose PD controller that commands the torso to be vertical. Energy regulation can be performed by changing the VPP position, and/or changing the rest leg length during stance in a predefined manner.

4.8 Conclusions

The objective of this paper was to show that the dynamics of the robot and controller work together to impose a virtual pivot point (VPP) on a TSLIP reduced-order template model. The reduced-order model was implemented by using a minimal number of constraints: the control inputs derived from the reduced order model are touchdown angle (q_{TD}) and the VPP position (r_{vpp}, θ_{vpp}). Besides these parameters, the dynamics evolve during stance as if the robot is the reduced-order model, and during flight the leg is smoothly and timely recirculated. Even though there was no explicit energy regulation or torso stabilization, the robot was able to take up to 22 steps and produced force profiles and VPPs that were similar to the reduced order model. These results give validation that reduced-order model studies can provide insights that can be applied to robots in the real world.

Chapter 5: Conclusion

In conclusion, this thesis gives a description of an open-source, real-time software stack for robot operation and control, as well as a description of two template controllers and their experimental results. The software stack is important because it is the virtual machinery that attaches the control theory to the electromechanical system. The first template controller paper describes how a Spring Loaded Inverted Pendulum (SLIP) controller is implemented on the robot along with experimental results, and is significant because this is the template model that the robot is mechanically designed to behave like. The last paper describes a template model that adds a Torso to the SLIP model (a TSLIP model), and provides a control method that regulates the torso on the reduced-order model (the virtual pivot point method). This method is tested in experiments on the robot.

The software stack paper provides an open-source solution that achieved sufficiently fast real-time operation (1ms control loop), was implemented on the robot, and was used for all subsequent experiments in this thesis. The SLIP model template controller was able to reproduce the desired dynamics, by creating ground reaction forces that approximated the forces generated by the reduced-order model. The TSLIP model template controller was also able to reproduce the desired dynamics, again by creating ground reaction forces that approximated the reduced-order model.

This work is important because it shows that SLIP derived reduced-order model control can be applied to robots. It also shows a path forward for testing other SLIP-like model theories about energy injection, gait stabilization, and torso stabilization. The goal is that this will help develop more robust and efficient bipedal robots, which will lead to their use in environments that are naturally suited to bipedal walkers. A few examples are human environments that have become dangerous due to fire, structural damage, or radiation. In these cases a bipedal platform provides a leg up when navigating spaces designed for bipeds.

Bibliography

[1] Atmel. Atxmega128a1-au datasheet. [Retrieved 1/22/13].

[2] Pranav A. Bhounsule, Jason Cortell, and Andy Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Proc. CLAWAR*, pages 441–448, 2012.

[3] R Blickhan. The Spring Mass Model for Running and Hopping. *Journal of Biomechanics*, 22(11-12):1217–1227, 1989.

[4] Yvonne Blum, Aleksandra Birn-Jeffery, Monica A. Daley, and Andre Seyfarth. Does a crouched leg posture enhance running stability and robustness? *Journal of theoretical biology*, 281(1):97–106, 2011.

[5] J.H. Brown and B. Martin. How fast is fast enough? choosing between xenomai and linux for real-time applications. Technical report, Open Source Automation Development Lab, [Retrieved 1/22/13].

[6] Sharon R. Bullimore and Jeremy F. Burn. Ability of the planar springmass model to predict mechanical parameters in running humans. *Journal of theoretical biology*, 248(4):686–695, 2007.

[7] J. Butterfaß, M. Grebenstein, H. Liu, and G. Hirzinger. DLR-hand II: Next generation of a dextrous robot hand. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 109–114. IEEE, 2001.

[8] Zhaopeng Chen, Neal Y. Lii, Thomas Wimböeck, Shaowei Fan, and Hong Liu. Experimental evaluation of cartesian and joint impedance control with adaptive friction compensation for the dexterous robot hand dlr-hit ii. *International Journal of Humanoid Robotics*, 08(04):649–671, 2011.

[9] S. H. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[10] Steven H Collins and Andy Ruina. A bipedal walking robot with efficient and human-like gait. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1983–1988. IEEE, 2005.

[11] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi. An atlas of physical human–robot interaction. *Mechanism and Machine Theory*, 43(3):253–270, 2008.

[12] Michael Ernst, Hartmut Geyer, and Reinhard Blickhan. *Spring-legged locomotion on uneven ground: a control approach to keep the running speed constant*, pages 639–644. 2009.

[13] Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29 – 45, 2008.

[14] R.J. Full and D.E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 1999.

[15] J Furusho and M Masubuchi. A theoretically motivated reduced order model for the control of dynamic biped locomotion. *Journal of Dynamic Systems, Measurement, and Control*, 109(2):155–163, 1987.

[16] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society B: Biological Sciences*, 273(1603):2861–2867, 2006.

[17] Mario Gomes and Andy Ruina. Walking model with no energy cost. *Physical Review E*, 83(3):032901, 2011.

[18] Jesse Grimes and Jonathan W. Hurst. The design of atrias 1.0 a unique monopod, hopping robot. In *International Conference on Climbing and Walking Robots*, pages 548–554, July 2012.

[19] Masato Hirose and Kenichi Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1850):1119, 2007.

[20] Christian Hubicki, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. ATRIAS: enabling agile biped locomotion with a template-driven approach to robot design. *in preparation*, 2014.

[21] Christian Hubicki and Jonathan W Hurst. Running on soft ground: simple, energy-optimal disturbance rejection. In *CLAWAR 2012*, pages 543–547, 2012.

[22] S Hyon and Takashi Emura. Symmetric walking control: Invariance and global stability. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1443–1450. IEEE, 2005.

[23] Kenji Kaneko, Fumio Kanehiro, Mitsuharu Morisawa, Kazuhiko Akachi, Gou Miyamori, Atsushi Hayashi, and Noriyuki Kanehira. Humanoid robot HRP-4-humanoid robotics platform with lightweight and slim body. In *IEEE/RSJ IROS*, pages 4400–4407, 2011.

[24] Susanne W. Lipfert, Michael Günther, Daniel Renjewski, Sten Grimmer, and Andre Seyfarth. A model-experiment comparison of system dynamics for human walking and running. *Journal of Theoretical Biology*, 292:1117, 2012.

[25] Thijs Mandersloot, Martijn Wisse, and Christopher G Atkeson. Controlling velocity in bipedal walking: A dynamic programming approach. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 124–130. IEEE, 2006.

[26] H.-M. Maus, S.W. Lipfert, M. Gross, J. Rummel, and A. Seyfarth. Upright human gait did not provide a major mechanical challenge for our ancestors. *Nature Communications*, 1(6):1–6, September 2010.

[27] H.M. Maus, SW Lipfert, M. Gross, J. Rummel, and A. Seyfarth. Upright human gait did not provide a major mechanical challenge for our ancestors. *Nature Communications*, 1:70, 2010.

[28] Horst-Moritz Maus, Jürgen Rummel, and André Seyfarth. Stable upright walking and running using a simple pendulum based control scheme. In *Advances in Mobile Robotics: Proc. 11th Int. Conf. Climbing and Walking Robots. Coimbra, Portugal: World Scientific*, pages 623–629, 2008.

[29] Tad McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

[30] Johannes Meyer and Armin Strobel. A flexible real-time control system for autonomous vehicles. *41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, pages 1 –8, June 2010.

[31] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[32] Neil Neville, Martin Buehler, and Inna Sharf. A bipedal running robot with one actuator per leg. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 848–853. IEEE, 2006.

[33] K. Okumura, H. Oku, and M. Ishikawa. Lumipen: Projection-based mixed reality for dynamic objects. In *Multimedia and Expo (ICME), IEEE International Conference on*, pages 699–704. IEEE, 2012.

[34] Andrew Peekema, Daniel Renjewski, and Jonathan W. Hurst. Open-source real-time robot operation and control system for highly dynamic, modular machines. In *ASME 2013, International Design Engineering Technical Conferences & International Conference on Multibody Systems, Nonlinear Dynamics, and Control*, 2013.

[35] Frank Peuker, Christophe Maufroy, and André Seyfarth. Leg-adjustment strategies for stable running in three dimensions. *Bioinspiration & Biomimetics*, 7(3):036002, September 2012.

[36] F. Pfeiffer, K. Loffler, and M. Gienger. The concept of jogging Johnnie. In *IEEE ICRA*, Washington, DC, May 2002.

[37] Ioannis Poulakakis and Jessy W Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *Automatic Control, IEEE Transactions on*, 54(8):1779–1793, 2009.

[38] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207. IEEE, 2006.

[39] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[40] Alireza Ramezani, Jonathan W. Hurst, Kaveh Akbari Hamed, and J. W. Grizzle. Performance analysis and feedback control of ATRIAS, a three-dimensional bipedal robot. *Journal of Dynamic Systems, Measurement, and Control*, 136(2):021012, 2014.

[41] Daniel Renjewski, Alexander Sprwitz, and Jonathan W Hurst. Exciting passive dynamics in a versatile bipedal robot. *IEEE Transactions on Robotics*, submitted, 2014.

[42] Juergen Rummel, Yvonne Blum, H. Moritz Maus, Christian Rode, and Andre Seyfarth. Stable and robust walking with compliant legs. In *IEEE ICRA*, pages 5250–5255, 2010.

[43] Harold Roberto Martinez Salazar and Juan Pablo Carbajal. Exploiting the passive dynamics of a compliant leg to develop gait transitions. *CoRR*, 83, 2011.

[44] Z. H. Shen and J. E. Seipel. A fundamental mechanism of legged locomotion with hip torque and leg damping. *Bioinspiration & Biomimetics*, 7(4):046010, 2012.

[45] ISO Standard. Iso 11898, 1993. *Road Vehicles, Interchange of Digital Information-Controller Area Network (CAN) for High Speed Communications*, 1993.

[46] Timothy Sullivan and Justin Seipel. 3d dynamics of bipedal running: Effects of step width on an amputee-inspired robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 939–944. IEEE, 2014.

[47] Spyros Tzafestas, Mark Raibert, and Costas Tzafestas. Robust sliding-mode control applied to a 5-link biped robot. *Journal of Intelligent and Robotic Systems*, 15(1):67–133, 1996.

[48] H R Vejdani, Y Blum, M A Daley, and J W Hurst. Bio-inspired swing leg control for spring-mass robots running on ground with unexpected height disturbance. *Bioinspiration & Biomimetics*, 8(4):046006, 2013.

[49] Hamid R. Vejdani and Jonathan W. Hurst. Swing leg control for actuated spring-mass robots. In *CLAWAR 2012*, pages 536–542, July 2012.

[50] M. Vucobratovìc and B. Borovac. Zero-moment point–thirty five years of its life. *Intl. J. of Humanoid Robotics*, 1:157–73, 2004.

[51] Miomir Vukobratović and Branislav Borovac. Zero-moment pointthirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.

[52] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. Hybrid zero dynamics of planar biped walkers. *Automatic Control, IEEE Transactions on*, 48(1):42–56, 2003.

[53] Martijn Wisse, Daan GE Hobbelen, and Arend L Schwab. Adding an upper body to passive dynamic walking robots by means of a bisecting hip mechanism. *Robotics, IEEE Transactions on*, 23(1):112–123, 2007.