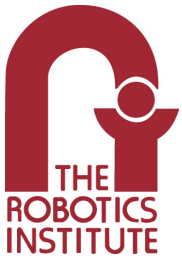
Gait Regulation Control Techniques for Robust Legged Locomotion

Galen Clark Haynes

CMU-RI-TR-08-19

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics



The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

May, 2008

Thesis Committee:
Alfred A. Rizzi, Chair
Christopher G. Atkeson
Nancy S. Pollard
Daniel E. Koditschek, University of Pennsylvania

Abstract

This thesis develops methods of control that allow a multi-legged robot to vary its stepping pattern, the gait of a robot, during locomotion. By constructing feedback control around the gaits a robot may use, we produce behaviors allowing a robot to switch amongst or return to certain gaits while performing feedback control during locomotion.

Gait regulation is one specific aspect of gait-based control, and pertains to the use of a control system to monitor and regulate the desired gaits a robot may use. While some gait-based control laws may force a robot to deviate from a nominal gait, gait regulation seeks to return to—or switch amongst—desired gaits as required. After discussing the necessary topological effects of gait regulation control, as well as noting specific constraints that are unique to legged systems, this thesis proposes methods of gait regulation control that place attractors and repellors on a high-dimensional toroidal space, a space relevant to gait timings, in order to converge upon desired gaits. The primary contribution of this thesis is an efficient algorithmic approach to gait regulation that avoids dangerous leg timings while converging to desired gaits, as specified. The system actively manages the basins of convergence for various controllers to achieve a global vector policy directing a robot to certain desired gaits.

This work is particularly applicable to four- and six-legged robots, on which a variety of interesting and useful gait timings exist. Specifically, we apply gait regulation to a climbing hexapod, on which we design a climbing behavior based upon a collection of reactive force control techniques, causing the robot to deviate from its desired gait. With gait regulation, the robot maintains use of its desired gaits, with the additional ability to actively transition amongst gaits while climbing.

Acknowledgments

I would like to thank my advisor, Al Rizzi, for his years of support for my thesis work. Al has always been helpful, supportive, and dedicated to my work, even when he has been hundreds of miles away for the final years of my graduate career. His advice, insight, and skill have all proven crucial to me throughout these many years, and it has been a pleasure to work with him.

I also would like to thank my committee members, Chris Atkeson, Nancy Pollard, and Dan Koditschek. Chris and Nancy, thank you for your insightful feedback on this thesis. Dan deserves a special note of thanks for playing a critical role in the last years of this thesis work, helping me formalize some of my ideas, and for directing me onto the exciting future work that this thesis opens up.

The Robotics Institute has been an amazing place to spend my graduate years, surrounded by brilliant people who are passionate about their research. In particular, I thank Howie Choset, Ralph Hollis, Jonathan Hurst, David Conner, Sarjoun Skaff, Hanns Tappeiner, and other members of the MSL and CFR communities.

The RiSE project funded the research in this thesis, and I'd like to thank a few members of that team in particular. Aaron Saunders at Boston Dynamics, thank you for designing amazing robots, worthy of really interesting research problems. The robot feet used for the experiments in this thesis are thanks to the design and refinement work of Alan Asbeck and Matt Spenko. Hal Komsuoglu also deserves special thanks for his tireless efforts to make cool robots go, and his interest in my work. Also, to the Boston Dynamics team, thank you for allowing me as a guest and visiting graduate student to your amazing facilities.

Noah Cowan, thank you for planting the crazy idea into my head that I should get a doctorate. Uluc Saranli, with whom I moved at the same time from Michigan to Carnegie Mellon, deserves thanks for being a mentor to me during my early graduate student years and for providing me with excellent advice on the graduate school experience.

Al Rizzi deserves thanks on a personal level as well. With my frequent trips to Boston to complete robot experiments, Al and his wife, Liz, graciously opened their house to me during several of my trips. Alex, Magic, and Quinn provided much entertainment before and after long days of robot experiments.

My friends in Pittsburgh, too numerable to name, deserve thanks for making these years so fun and exciting. You all were perfect distractions from grad school, with all of the get-togethers, summer grilling, and the many memorable road trips and backpacking trips.

I thank my mom, Gail, and my family for their words of encouragement even during the toughest of times. Their constant love and support has been amazing.

Lastly, I'd like to thank Emily—and, of course, Hayle and Cleo—without whom the last months of this thesis work would have been unbearable. You have been an amazing support to me, and I am excited for what the coming years will bring.

Contents

List of Figures

Chapter 1

Introduction

The realization of effective legged robots, capable of terrestrial locomotion rivaling even the simplest of biology's legged creatures, remains a principal challenge in mobile robotics. While legs offer significant advantages over other locomotive approaches, particular their efficiency over sparse and irregular terrain, the difficulty in designing a legged mechanism, generally composed of many degrees of freedom, and the complexity of control required has impeded progress. This thesis studies the control aspects of legged locomotion, suggesting a method of abstraction that obscures certain complexities of control while retaining desired core aspects of legged locomotion.

Wheeled vehicles operate using rolling contact between wheels and the ground, applying continuous torque to each motor to propel forward. Tracked vehicles work similarly, maintaining constant contact with a surface while increasing the friction of contact from treads. Legged locomotion is distinct, however, due to its intermittent contact with the environment. Legs periodically switch between providing support for the body, to generate propulsion, and lifting off the ground to recirculate forward. This constant switching between support and recirculation phases introduces an inherent periodicity, unlike the continuous operation of other vehicles.

1.1 Motivation

The control approach we develop takes into account the periodicity of legged locomotion when designing behaviors. Open-loop gaits, strictly periodic motions that a robot can execute, are used as a simple basis for adaptive feedback behaviors. With gaits abstracting the high-dimensional motions of a robot's legs, we study just the timing by which feet make

contact with a surface. Building feedback controllers on the timing of feet in a gait, we are able to make continuous adjustments in order to improve locomotion.

Gaited locomotion is common in biological systems, with patterns emerging for different classes of animals, depending upon the number of legs. Quadrupeds exhibit typical gaits such as the walk, trot, pace, and gallop—as just a few examples—while hexapods have the wave gait and the alternating tripod [25, 34, 50]. While biomechanics may vary amongst different animals, the timing of legs remain similar for certain gaits.

Gait patterns have been successfully utilized with robotic systems as well, particularly those with four or six legs, on which a variety of interesting gaits exist. The Adaptive Suspension Vehicle utilized prespecified collections of gait timings to achieve various forms of legged locomotion for a statically stable robot [44]. Similarly for a dynamic quadruped, the work of Raibert and his lab enabled a variety of dynamic gaits—the trot, pace, and bound—as well as the ability to perform dynamic transitions between gaits [37–39]. Furthermore, the Ambler robot built at Carnegie Mellon performed planning at the foot level after a gait controller dictated the order in which to recirculate feet [49].

Gaited locomotion continues to meet success with robotic systems, moving onto smaller and simpler legged robots, while relying upon compliance and underactuation to achieve locomotion. With success on both four and six-legged systems [2, 4, 10, 36], a current popular method is to store fully open-loop gait motions, prespecified motions for both the timing and geometry of a gait, and play back an individual gait to generate locomotion. Compared to previous systems, which used models of internal dynamics or expensive sensing of the surrounding environment, these systems often operate with little or no sensors present, often utilizing a robot's underactuation and compliance when locomoting.

The work contained in this thesis focuses on this open-loop gaited locomotion approach, in which an underactuated robot performs a task using a fully prespecified gait as its basis, in order to develop feedback controllers. Studying in particular just the timing, we develop control methods that modify a robot's gait, in order to keep the robot using certain gaits (in the presence of other feedback controllers which deviate away from such gaits) as well as switch amongst gaits while locomoting. While previous methods exist for switching between gaits, they do not address the use of underactuated legged robots which often have constraints upon their locomotion.

In particular, this thesis studies control issues associated with gait regulation, forcing a robot to converge to preferred gait timings. By understanding these implications of control, and constraints that exist upon the control spaces, we propose control methods that safely allow a robot to converge while avoiding dangerous leg timings.

1.2 Related Work

Legged locomotion is inherently complex, with a variety of approaches taken over the years to achieve mobility to attempt to rival biological systems. We provide an overview of basic methods for encoding legged locomotion, focusing on areas of specific relation to the work in this thesis.

1.2.1 Legged Control Strategies

Over decades of research, approaches for control of legged robots have generally fallen into one of two different camps. One approach relies upon careful planning of robot footsteps, using sensing as well as robot and world models to computationally determine routes of locomotion. A second approach is that of emergent behaviors, allowing complex networks of controller components, each component influencing other portions of the network, to produce overall locomotive behaviors.

The emergent approach for legged robot control—often termed behavior-based robotics or, more simply, reactive control—has a basis in many years of work reverse engineering the neuronal bases of locomotive control [15, 19, 50]. Many different approaches have been proposed [7, 15, 19, 20], using various networks of influences to produce robot behaviors. Complex networks of control reflexes, the behavior primitives of the network, as well as simple coordination schemes amongst legs, work together to produce very natural legged locomotion, often incorporating robot sensors and basic intuition of locomotion, with a particular application to quasi-static walking machines. The work by Cruse [15, 19] is of particular merit, developing strategies for cross-talk between distributed leg controllers as well as describing methods of incorporating sensors to adjust for body pitch and load sharing amongst legs.

A general shortcoming of this reactive approach, however, is that—beyond a set of strategies and core components—generality of the approach is hard to deduce, with the

produced behaviors sometimes occurring by accident [7]. Sometimes, furthermore, these systems may require extensive manual or automated learning of parameters to generate successful locomotion [32]. Despite these difficulties, these approaches have been successfully used to generate locomotion while requiring little computing power, yet integrating sensors to produce robust locomotion.

The opposite, historically older but still popular, approach for legged locomotion control has used the traditional robotics paradigm of "sense–think–act", building complex models of a robot's environs to plan a path to execute during locomotion. These methods often involve careful planning based upon the sensed surroundings to deliberately and carefully plan every footfall a robot makes [6, 12, 26, 43, 49]. The approach requires very accurate sensor information as well as accurate modeling of the world to plan through.[1] Using carefully sensed—often a priori—maps and information, planning algorithms compute a path by which a legged robot can navigate amongst complex terrains.

While it is possible to generate high-level behaviors using planning, this strategy is difficult to implement on small, fast, and possibly dynamic legged robots, due to the difficulties in modeling and implementing computationally-intensive algorithms on these platforms.

1.2.2 Open-Loop Gait-Based Approach

A natural progression from the two previous modes of operation is one of blind insight. Given the cyclic nature of legged locomotion, one would expect a robot to choose repetitive leg motions when presented with a uniform terrain.

This leads to the idea of using gait patterns to encode locomotion. Robots store prespecified motion patterns, replaying each when necessary [2,4,10,36]. These methods have been simple, consisting of basic trajectory tracking, and often use little or no sensing in implementation. A robot makes use of a specific gait, usually from a collection of multiple gaits, and simply plays the gait's periodic motions over and over.

Robustness is obtained generally through mechanical design, introducing unique dynamic legged robots that are truly capable of energetic running [2, 10, 36], yet compliant to maintain stability despite disturbances.

[1] It should be noted that reactive systems and behavior-based robotics grew out of a frustration over difficulties in sensing and modeling for a mobile robot [8].

While much work has been performed tuning open-loop gaits based upon desired properties [11, 29, 47, 51], little has been done to show how to create systems that are robust in their control in addition to their mechanics.

Our approach is applied to a climbing robot that follows gait-based strategies [40, 42, 45]. By relying upon a gait's open-loop characteristics, we build control systems upon a simpler set of parameters of the robot's gait, in order to apply feedback. Prior work [24] has discussed the ability to describe advanced behaviors using individual gaits connected by similarly open-loop gait transitions, however this approach was difficult to generalize to feedback scenarios. Our current work, rather than amassing gaits and transitions, builds control systems upon the space of robot gait timings, leading to adaptable behaviors.

While previous work has sought to adjust timing of legs in a gait [20, 31], it has been performed with relatively unconstrained systems. Gait-based locomotion systems, with the use of leg compliances or underactuated mechanisms, often suffer from constraints on their stance stroke, thus limiting the phase at which recirculation may begin, a constraint upon gait control that is addressed in this thesis.

1.2.3 Leg Coordination

With legged locomotion a cyclic task, one natural way of describing such systems is to use coupled oscillators to design leg coordination, for which there exists a large body of work. By specifying properties amongst a set of oscillators, through a description of the desired relationships between pairs of legs, these methods produce convergent control laws for desired gait timings.

With a foundation in the biological use of central pattern generators and neural oscillators [14, 16, 35], there exist many different approaches for applying coupled oscillator systems to legged locomotion [5, 9, 22]. We particularly focus on the work of Klavins and Koditschek [28] to build control functions that coordinate the movement of independent elements in a cyclic system. The approach taken is to coordinate legs by attracting, repelling, or simply having no communication between pairs of legs. After constructing such a network as hierarchies of relationships amongst legs of a robot, analysis tools can be used to check whether a given system, as it controls the phase velocities of each oscillator of the system, converges to a desired timing relationship amongst the oscillators. This work has previously been used to control the legs of a hexapedal running robot scrambling over

rough terrain [46], in which the coordination strategies keep the robot close to an alternating tripod gait while the robot applies feedback to locomote over the surface.

The approach taken by these prior methods focuses on limited communication amongst leg pairs, building stable controllers for specific gaits. This is compared to the work in this thesis, where we focus on controllers for a wide variety of gaits, and design controllers based upon their convergence properties, not only studying the existence of a convergent gait but also the paths taken during convergence.

1.3 Specific Contributions

The core contributions of this thesis deal with the use of gaits[2] as a basis for building robust feedback behaviors, used here to allow a legged robot to climb challenging vertical surfaces. Our approach makes explicit use of a decomposition of gaits into their temporal and spatial components. We encode the temporal portion of a gait as control upon leg phases, a dynamical system that produces various stable limit cycles at desired gait timings. The spatial characteristics of limb attachment, detachment, and body pose, typically specific to a given substrate, are encoded by a separate system of transforms from phase coordinates into the configuration space of a robot's joints. Prior use of both phase control and feedforward gait motions has been noted in the review of related work, thus this section will now establish the particular contributions this thesis makes to the field of legged locomotion.

Foremost, we show how we can exploit the temporal and spatial decomposition of gaits to build controllers that work specifically on the timing of a gait, leading to adaptable behaviors with robustness to climb difficult vertical surfaces. The utility of controllers on gait timing is obvious when we consider behavioral feedback. Specifically, we introduce proprioceptive reflexes that deviate from preferred gait timing to regulate task-level behaviors, such as ensuring robot grip on a climbing surface. These disturbances push the robot away from a desired gait limit cycle, for which we use our gait timing controller—a gait regulation controller, with its specific placement of basins of attraction surrounding preferred gaits—to regulate the gait pattern. We include multiple basins of attraction, each surrounding a different preferred gait, to provide an automated defense against disturbances during climbs.

[2] By "gait" we mean the open-loop patterns of leg and body motions that result in effective locomotion over a given substrate.

When proprioceptive feedback terms accidentally force a system to cross basin boundaries, the robot transparently and safely switches gaits without the need for operator attention. A second form of gait change is by explicit choice of policy. As gaits differ in their locomotive characteristics—some favoring speed over stability, others cautious and slow—a robot operator can choose to switch amongst gaits on command. The basins of attraction used for our gait controllers avoid regions of unsafe operation while also guaranteeing convergence. In summary, by designing our controllers based upon desired properties of the dynamic system controlling gait timing, have proven to be crucial to the longevity of climbing over natural and man-made exterior surfaces, incorporating both automated switching of basins as well as operator-chosen transitions amongst gaits.

The second contribution is an informal but intuitively effective approach for designing control systems for underactuated legged robots, separating the control of gait timing from issues of geometry, posture control and compliance. The RiSE robot uses a highly underactuated mechanism to climb large distances, passively adapting without high-level commands. We incorporate RiSE's underactuation by representing spatial components of gaits with sets of leg motions—each tuned per surface to excite compliances found in legs, ankles, and feet—separately from temporal parameters that we use for feedback control. By decoupling these gait parameters, we discuss the readily and intuitively tunable nature from surface to surface. For a certain gait, the same timing parameters remain effective across a large variety of surfaces. For a fixed surface, a single set of geometric parameters remain effective for a variety of gait timings. A more traditional approach would be to plan a robot's motions up a challenging wall. Sensing and modeling of the surface, however, cannot provide the level of detail required for RiSE's operation, as the robot relies until tiny asperities to grasp a wall, yet may progress over tens of meters in a climb. In summary, the decomposition and parameterization of gait-based control offers a simple approach to develop broad libraries of robust climbing behaviors, suitable for a wide range of environments and tasks. In this thesis we focus on robot climbing of one specific surface, for which these components are tuned.

We have asserted critical properties of our dynamic system for control of gait timing, for which we develop a novel hybrid gait regulation control approach that uses an automated deployment of basins of attraction to design safe convergence to desired gaits. The critical goal of this system is to avoid physically dangerous phase differences amongst legs, which

we term phase space obstacles, that occur when too many legs recirculate together. Starting with safe configurations in phase coordinates, in which the robot is safely providing support for its body, our system should drive itself to a desired limit cycle gait while incurring a minimal number of incursions into these phase space obstacles. Difficulty in designing our basins of attraction, however, comes from the following topological obstruction: basins of attraction must have the same homotopy type as the attractor they surround. The homotopy type of our basins, therefore, must match that of the cycle, and are different from the space within which they are contained, the torus. Additionally, with the further possibility of multiple preferred gait cycles, each of which is an acceptable gait for steady state locomotion, we devise methods to join multiple basins together by constructing boundaries between them in a logical manner. In summary, we design an approach that meets the desired properties of our control system, using a logical deployment of basins of attraction via a cellular decomposition of the torus—with knowledge of phase space obstacles as regions to avoid—to facilitate the inclusion of many desired gaits and to design convergent basins to allow gait switching upon operator command.

1.4 Thesis Outline

In Chapter 2, we describe basic methods for gait-based locomotion, and present a component view of constructing a gait reference trajectory generator. We introduce a spatio-temporal decomposition of gaited locomotion, in which the timing of stance-flight patterns are encoded using a phase space transformation, while the spatial properties of a desired gait—the limb-level geometry of foot attachment and detachment from a surface—is a memoryless transform from phase coordinates to robot joint space. This chapter ends with a discussion of gait-based control, building robot behaviors by applying feedback control to gait parameters.

Chapter 3 builds upon the discussion of gait timing by introducing gait regulation, the use of a control system to manage the gait a robot uses. By designing convergent limit cycles with vector fields on a high-dimensional torus, we produce controllers that move a robot toward certain desired gaits. We introduce phase space obstacles, regions of the torus corresponding to unsafe simultaneous leg recirculations, and discuss the necessary topological implications of control when producing convergence to desired gaits.

Chapter 3 also presents the control algorithms of our new approach to gait regulation, in which desired limit cycles produce "safe" basins of attraction—only passing minimally through phase space obstacles. We specifically describe a hybrid control approach used to guarantee global convergence when desired, and to mediate ownership between various desired limit cycles.

Chapter 4 compares our approach for gait regulation with a variety of other possible approaches, through the use of simple numerical simulations.

The RiSE robot, our experimental platform, is introduced in Chapter 5. We highlight the application of RiSE to climbing vertical rough exterior building surfaces, such as stucco, while describing an open-loop gait used for locomotion. This chapter proceeds to describe a variety of control components that are included in an overall climbing behavior along with gait regulation control.

Chapter 6 presents a wealth of climbing data gathered using the RiSE platform. We present empirical results that show the addition of robustness for our climbing machine through our feedback approach, as well as show the utility of gait switching while safely avoiding phase space obstacles.

The appendices of this thesis provide a detailed description of the combinatorics involved with legged gaits, an explanation of the cellular decomposition used for our control algorithms, as well as a listing of example Matlab© code that simulates basic gait regulation experiments.

Chapter 2

Gaits: Robust Locomotion with Minimal Sensing and Control

In designing a control approach for legged systems, we have constructed methods to abstract and decompose locomotion into a collection of more facile systems. Using these constructs, we separate the spatial from temporal components of locomotion, and mix feedforward with feedback control, in order to create robust behaviors. This section highlights design decisions made during this process.

Legged systems necessarily involve complex foot-surface interactions to locomote; our use of gaits as abstractions allows us to capture the core aspects of locomotion, while not precisely modeling these interactions. A decomposition of spatial from temporal components allows us to express gaited locomotion as a composition of a central pattern generator with a kinematic mapping. With an effective method of parameterizing these systems, we introduce methods of designing feedback control, to realize robust behaviors, using feedforward gaits as a basis.

2.1 Behaviors and Gaits

The dominant characteristic of legged locomotion, compared to other forms of terrestrial locomotion, is the discreteness of contact that legs have with the ground. As a robot moves forward to locomote, the set of supporting legs must change as individual legs recirculate. Broadly speaking, legged locomotion includes any possible sequence of discrete foot contacts. In this section, we study how gaits form a unique subset of such behaviors.

Static walking robots maintain body support continuously, the robot keeping static support at all times even though the set of supporting legs may change. With each stance phase of a leg, there is a finite stride length possible, after which a leg protracts to recover that distance and begin a new stride. This constant need to recirculate each leg forces the inherent discontinuous nature of legged locomotion.

We use this discreteness as our first method of trying to understand locomotion. Assuming a given legged behavior can be defined by the time-varying trajectories of a robot's joints, we first consider a function, $b(t)$, whose codomain is the robot's joint configuration space, $\mathbb{Q}$. This function simply describes how a specific behavior moves the robot's joints (currently ignoring sensor information or any decision making processing). We consider a second function, $s(t)$, to describe only whether or not individual legs are currently in stance and supporting the body, mapping from time to the set $(0,1)^N$ (for an N-legged robot).

$$b(t) \quad : \quad \mathbb{R} \rightarrow \mathbb{Q} \tag{2.1}$$

$$s_i(t) \quad = \quad \begin{cases} 0 & \text{if leg } i \text{ is recirculating at time } t \\ 1 & \text{if leg } i \text{ is in contact with surface} \end{cases} \tag{2.2}$$

$$s(t) \quad = \quad s_1(t) \times \ldots \times s_N(t) \tag{2.3}$$

The rising edges of (2.3) indicate the times at which legs make initial contact, per stride, with the surface. Each leg repeats the tasks of stance and recirculation, and we can study the relative timings of these events to determine whether legs act together or singly, and what sets of legs work together to provide body support.

The space of functions, $b(t)$ and $s(t)$, covers all possible legged behaviors. To simplify our discussion, however, we focus on those behaviors exhibiting periodic motion, the space of open-loop gaits, $\mathbb{G}$. A gait, $g \in \mathbb{G}$, is a periodic mapping from a phase coordinate, $\phi \in S^1$ to $\mathbb{Q}$. Phase is a concept similar to time, in that both monotonically increase, however phase is cyclic and often thought best of as a winding clock. S^1 is the space of the unit circle, a differentiable manifold we represent using the interval $\left[0 \quad 1\right]$ with endpoints identified. We describe a behavior using an open-loop gait as follows.

$$g(\phi) \quad : \quad S^1 \rightarrow \mathbb{Q} \tag{2.4}$$

$$s_i(\phi) \quad = \quad \begin{cases} 0 & \text{if leg } i \text{ is recirculating at phase } \phi \\ 1 & \text{if leg } i \text{ is in contact with surface} \end{cases} \tag{2.5}$$

$$s(\phi) \quad = \quad s_1(\phi) \times \ldots \times s_N(\phi) \tag{2.6}$$

Compared to $b(t)$ and $s(t)$, these new functions, $g(\phi)$ and $s(\phi)$ can only represent cyclic behaviors, a much tighter set of possible functions.

2.2 Specification of Robot Gait

As the space of all possible gaits, $\mathbb{G}$, includes any periodic mapping from phase to $\mathbb{Q}$, we are interested in methods to simplify and parameterize our definition in ways that are well-suited for the domain of legged locomotion. By restricting discussion to a certain class of gaits, and introducing memoryless transformations in both phase coordinates and joint coordinates, we build a decomposition for a basic set of robot gaits.

As discussed, a gait is a cyclic function, encoding the open-loop timing by which stance and recirculation occur. Our first restriction is to consider only those open-loop gaits that recirculate each leg exactly once per stride. This simply states that $s(\phi)$ has only two distinct regions, one for stance, during which $s(\phi) = 1$, and one for recirculation, $s(\phi) = 0$. With this assumption, we move onto discussions of gait decompositions.

We elucidate these two portions of a gait function with the following decompositions. Foremost, we separate a gait function into the motions for each leg, a mapping from phase to $\mathbb{Q}_i$, the configuration space for leg i. This is under the assumption that the robot's configuration space is simply the Cartesian product of the individual leg configuration spaces, $\mathbb{Q} = \mathbb{Q}_1 \times \ldots \times \mathbb{Q}_N$.

$$g_i(\phi) \quad : \quad S^1 \rightarrow \mathbb{Q}_i \tag{2.7}$$

$$g(\phi) \quad = \quad g_1(\phi) \times \ldots \times g_N(\phi) \tag{2.8}$$

A second decomposition considers each of these leg functions as having distinct temporal and spatial aspects that can be separated further. If $g_{i,t}$ and $g_{i,s}$ are the temporal and spatial components of g_i, we construct the following:

$$g_i(\phi) = g_{i,s} \circ g_{i,t}(\phi) \tag{2.9}$$

The temporal portion, a *clock mapping*, $g_{i,t} : S^1 \rightarrow S^1$, describes the relative speeds of legs in stance versus in flight, as well as the phasing amongst various legs. The spatial portion of a gait, a *leg trajectory*, $g_{i,s} : S^1 \rightarrow \mathbb{Q}_i$, maps stance and flight into a desired geometric foot path trajectory. The clock mapping functions serve as a type of central pattern generator, dictating the rhythmic occurrence of stance and recirculation for each individual leg, while the leg trajectories are basic kinematic mappings.

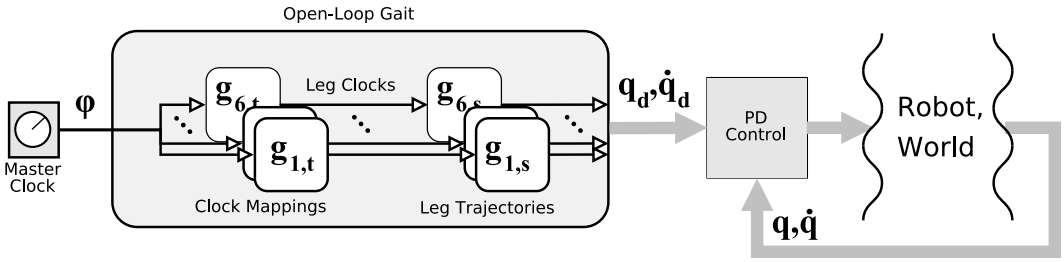


Figure 2.1: Open-loop gaits and their decomposition into temporal and spatial portions

With this decomposition of a gait, and our assumption that each leg recirculates only once per stride, we make a distinction of the phase angle at which each given leg begins stance, the *stance phase offset*, $\rho_i \in S^1$. This parameter is an open-loop parameter, and reflects the phase at which a leg should ideally touch the ground, and is distinct from whether or not a leg is actually in contact. Phase offsets may differ for legs in a gait, and their ordering indicates the timing of legs. As such, each phase offset, ρ_i, corresponds to the rising edge of the $s_i(\phi)$ function in (2.6). The space of a set of phase offsets, $\mathbf{x} = \begin{bmatrix} \rho_1 & \dots & \rho_N \end{bmatrix}^T$, is the N-torus, $\mathbb{T}^N$:

$$\mathbb{T}^N = S^1 \times \ldots \times S^1 \tag{2.10}$$

Phase offsets describe the order by which legs contact the surface. Duty factors, another temporal parameter described by $\delta_i \in S^1$, reflects the percentage of phase spent in stance

for a given leg. Just as ρ_i is the phase angle at which a leg contacts the surface, $\rho_i + \delta_i \in S^1$ is the phase angle at which stance ends. Thus, $s_i(\rho_i + \delta_i)$ is the falling edge of s_i. We define $\mathbf{d} = \begin{bmatrix} \delta_1 & \dots & \delta_N \end{bmatrix}$ as the vector of a gait's duty factors.

These temporal parameters define the clock mapping functions, $g_{i,t}$, and are used to define a local change of coordinates to describe the system in terms of individual leg phases, $\bar{\mathbf{x}} = \begin{bmatrix} \psi_1 & \dots & \psi_N \end{bmatrix}$:

$$\bar{\mathbf{x}} = \phi\vec{\mathbf{1}} - \mathbf{x} \tag{2.11}$$

In this set of coordinates, stance always begins at $\psi_i = 0$ and ends as $\psi_i = \delta_i$. The combination of phase offsets, duty factors, and the change of coordinates to leg phases allows us to fully characterize (2.6), by specifying precisely the portions of S^1 dedicated to stance versus flight, in terms of phase.

$$\psi_i = \phi - \rho_i \tag{2.12}$$

$$s_i(\phi) = \begin{cases} 1 & \text{if } \psi_i \in [0, \delta_i) \\ 0 & \text{if } \psi_i \in [\delta_i, 1) \end{cases} \tag{2.13}$$

2.3 Examples of Gait Timings

With phase offsets and duty factors as methods of parameterizing the clock mappings of gaits, we discuss basic examples of gaits for quadrupedal and hexapedal robots.

In designing an open-loop gait for a legged robot, we follow two design steps. Foremost, we choose the desired timing of the gait, in terms of stance and recirculation, by selecting phase offsets that produce a certain recirculation order, and duty factors the provide overall stability for the body, all of which is stored in the $g_{i,t}$ functions. Second, we tune the geometric motions a individual foot makes, designing the $g_{i,s}$ functions. We have chosen to implement these functions as simply parameterized piecewise cubic functions, stored on a robot and composed together to realize a gait function with little computational cost.

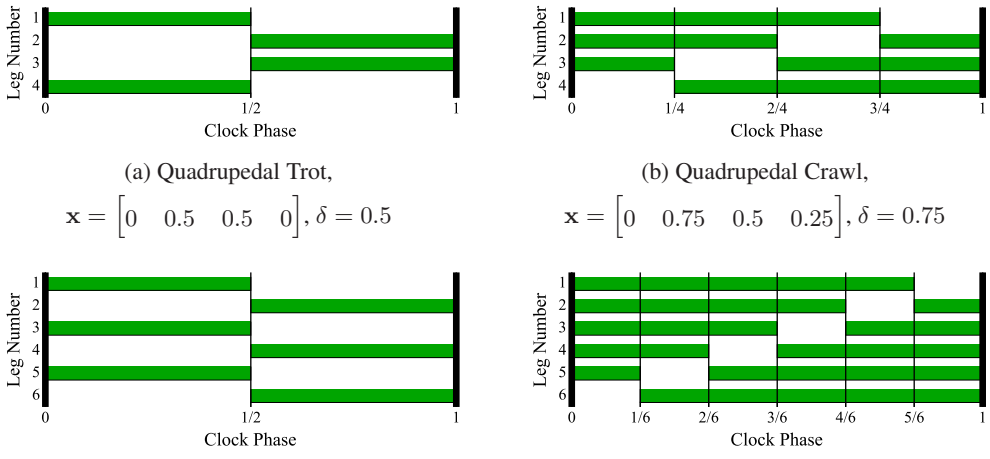
Discussing gait examples, we are interested mainly in the timing of different styles of locomotion. A variety of gait timings are illustrated in Fig. 2.2. For each of these, $N \times 2$

temporal parameters determine the clock mappings of the gait. In these examples the duty factor elements all have the same value.

These examples simply describe the patterns of stance and recirculation for a single stride of a gait, and ignore the kinematic mapping. Gaits designed for higher duty factors (such as the crawl and wave gaits in Fig. 2.2) keep more legs in contact with the surface compared to others. Due to speed limitations of recirculating legs, it is common for gaits like the trot and tripod to locomote significantly faster. These temporal patterns remain similar across robots with different morphologies, with similarly varying ranges of stability and speed.

2.4 Gait-Based Feedback Control

We have introduced a basic method of using a central pattern generator—the clock ϕ—to drive a series of memoryless transforms, resulting in feedforward locomotion. We are interested, however, in designing feedback behaviors of greater complexity that are not necessarily limited to the purely cyclic patterns of gaits. The approach we have chosen in building these behaviors uses open-loop gaits as a basis for locomotion, while constructing



(a) Quadrupedal Trot, $\mathbf{x} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \end{bmatrix}, \delta = 0.5$

(b) Quadrupedal Crawl, $\mathbf{x} = \begin{bmatrix} 0 & 0.75 & 0.5 & 0.25 \end{bmatrix}, \delta = 0.75$

(c) Hexapedal Tripod, $\mathbf{x} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}, \delta = 0.5$

(d) Hexapedal Wave Gait, $\mathbf{x} = \begin{bmatrix} 0 & \frac{5}{6} & \frac{4}{6} & \frac{3}{6} & \frac{2}{6} & \frac{1}{6} \end{bmatrix}, \delta = \frac{5}{6}$

Figure 2.2: Examples of gaits for 4- and 6-legged robots, shown as visual representations of the $s(\phi)$ function for each gait. Green shaded regions note, for each leg, the occurrence of stance. Stance phase offsets determine the clock phase at which stance begins, while duty factors correspond to the length of stance, versus a stride.

dynamical systems using the parameterizations of gaits introduced in §2.2. As each unique gait describes a basic style of locomotion, a behavior making use of a specific gait will build upon that style, while layering simple feedback controllers to add adaptability.

The parameterizations of the temporal aspects of gaits—phase offsets and duty factors—allow us to write a gait as a function of more than just the phase of the clock ϕ. By designing control laws to modify these parameters while locomoting, we can develop behaviors as time-varying gaits. By modifying temporal parameters of a gait, these controllers perform phase control to the individual legs.

We interpret a complete legged behavior, as shown in (2.1), as a gait augmented with time-varying gait parameters, further making use of our spatio-temporal decomposition of a gait (2.9). We modify temporal parameters to affect the timing of our central pattern generator without changing spatial characteristics, modifying the ordering and/or relative lengths of stance versus recirculation. A separate set of equations can modify the spatial properties of a gait, to generate behaviors with adaptable geometries.

There exist three ways in which we change temporal properties of a gait. The simplest method changes the stride rate of locomotion by speeding up or slowing down the central pattern generator, making our clock phase a time-varying parameter with varying speeds, $\phi(t)$. With higher values of $\dot{\phi}$, stride times decrease and recirculation rates increase. The other two methods of changing gait timing involve modifying the phase offsets of a gait as time-varying parameters, $\mathbf{x}(t)$, and similarly changing duty factors, $\mathbf{d}(t)$.

Modifications to the spatial properties of a gait are made via offsets in commanded position, adjusting the output joint commands of a gait with a gait offsets, $g_o(t) \in \mathbb{Q}$. The definition of a complete behavior, using these many forms of control, follows:

$$b(t) = g(\phi(t), \mathbf{x}(t), \mathbf{d}(t)) + g_o(t) \tag{2.14}$$

With a gait's base components, the clock mappings and leg trajectories, as a basis, we can then develop simple control laws that command how to affect the values of the phase offsets, duty factors, and gait offsets. In particular, this thesis focuses on the control of primarily the phase offsets.

2.5 Equivalence to Coupled Oscillators

We have chosen to use a master clock, ϕ, along with a set of phase offsets, $\mathbf{x}$, as our state-space representation of a robot using a gait. It is worth noting, however, that this system can be represented using coupled oscillators, and thus can be directly related to the research found in §1.2.3.

While the notion of a master clock is a centralized concept, coupled oscillators do away with this central clock, giving each oscillator its own phase angle. Methods of coordination are constructed between oscillators in a decentralized way.

Eqn. (2.11) shows the relationship between a centralized clock representation and a set of individual clocks, one per leg. In essence, the centralized clock in our representation only adds forward progress to each individual clock, from which we offset with our chosen parameters of phase offsets. The set of individual leg clocks, $\bar{\mathbf{x}}$, is equivalent, however, to our use of phase offsets and a master clock.

The reason for our specific choice of parameterization comes in our ability to discuss the action of our controllers. In the case of a fixed gait, where a robot operates in an open-loop manner, the phase offsets of the gait remain constant, $\dot{\mathbf{x}} = 0$. Gait-modifying behaviors, as developed in this thesis, result from time-varying phase offsets. This distinction between open-loop and closed-loop behaviors is why we choose to use phase offsets over simply using coupled oscillator phase angles.

2.6 Concluding Remarks

Our use of open-loop gaits, storing essential characteristics of feedforward locomotion, allow us to base feedback behaviors upon specific modes of locomotion. More specifically, our separation of gait timing from gait geometry is extremely useful in our approach to designing complex feedback behaviors.

In the remaining chapters of this thesis, we will discuss behaviors that modify a robot's gait timing, realized by applying control laws to the phase offsets of a gait. As the phase offsets are each members of the cycle, S^1, the relevant space of gait timing is the N-torus, $\mathbb{T}^N$, with our controllers operating as vector fields on the torus.

Chapter 3

Gait Regulation: Motivation and Technical Approach

In this chapter, we discuss *gait regulation*, the use of a control system to converge upon preferred gaits while simultaneously avoiding dangerous regions of the configuration space of gaits. As we describe basic gait properties using temporal aspects of gaits as a chosen parametrization, gait regulation focuses on control of these timing parameters. As a control system, this means describing a dynamic system on the N-torus, the phase space of gait timing, that converges upon a certain limit cycle gait.

After discussing specific challenges of gait regulation, we introduce our hybrid control approach, in which a collection of control policies are algorithmically composed in order to produce properties of gait convergence we desire.

3.1 Challenges of Gait Regulation

We first discuss challenges in the form of realizable obstacles that exist in the phase space of gaits, as well as properties of convergence that are required of a gait regulation controller.

3.1.1 Physicality of Obstacles in Phase Space

A robot that attempts to recirculate too many of its feet at once is in danger of losing static stability. Conversely, a statically stable legged robot must keep some set of its feet in contact with the ground surface at all times. Translated as constraints on the gaits a robot may use, we describe an obstacle set within the configuration space of gaits. Unlike familiar notions of geometric obstacles in a robot's workspace, these obstacles exist in the space of possible gaits, and coincide with those gaits that risk the robot's loss of static stability. Existing on an

N-torus, these so-called $\mathcal{P}$-obstacles are important to understand, as well as to avoid when applying gait regulation control. Control policies that avoid these regions while converging to a desired gait are preferable.

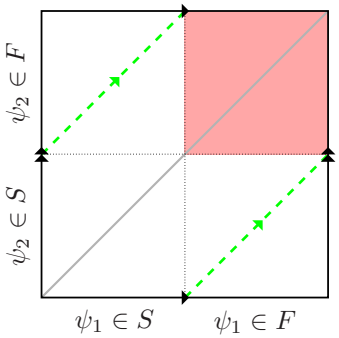
With each leg oscillator recirculating once per stride, there exist regions for each axis of phase space corresponding to stance and to flight. With a gait winding around the N-torus, recirculating each leg once per stride, each leg will recirculate for some portion of the stride. Depending upon the phase offsets of a gait, it may pass through these $\mathcal{P}$-obstacles, in which too many legs recirculate at the same time.

With two legs, the corresponding phase space, the 2-torus, admits a simple obstacle set, the size of which is based upon gait duty factor. When we assume a duty factor of $\delta = 0.5$, there exists a single admissible gait[1] (while, with a slight larger duty factor, a small range of admissible gaits exist). Fig. 3.1a shows the torus with this specifically-sized obstacle and single gait identified. If both legs were to recirculate at the same time, i.e. the gait configuration falling within the upper right quadrant of the torus, a static robot would have no support for its body. In this example, only one useful gait exists, the familiar bipedal walking gait that keeps the two legs out of phase with one another.

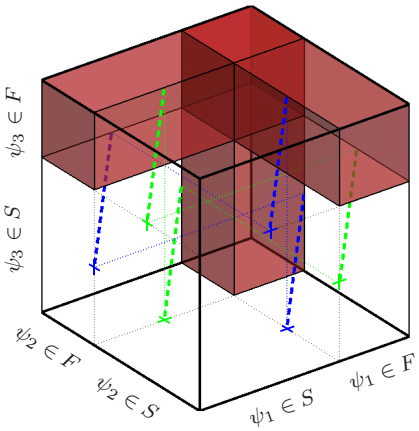
As we consider systems with a greater number of legs, the dimensionality of the phase space increases, as does the number of admissible gaits. Consider the three legged example shown as a 3-torus in Fig. 3.1b. Instead of a mere quadrant of the torus, we see that the shape of the $\mathcal{P}$-obstacle set becomes more complex—wrapping around the 3-torus in multiple directions—and that there exist two distinct and different admissible gaits, corresponding to different footfall patterns, that both avoid the obstacle entirely when the gait duty factor is precisely $\delta = \frac{2}{3}$.

Moving to higher-dimensional phase spaces, as are encountered with 4- and 6-legged systems, the topology of the obstacle continues to change, and we note differing layers of the obstacle. In the case of the 3-torus, two separate classes of obstacle exist, one corresponding to three legs all recirculating at once (a small cube in one corner of our representation of the torus), and another type of obstacle corresponding to two legs recirculating

[1] A gait in this arrangement is a cycle that uniformly increases on both axes of the torus at the same rate, thus preserving duty factor of the gait as the percent of each stride spent in stance. This follows also from our use of phase offsets to define basic properties of a gait's timing.



(a) With a stance duty factor of 50%, there exists a single gait (dashed line) not passing through the obstacle set.



(b) Three distinct regions of the $\mathcal{P}$-obstacle correspond to two legs recirculating, with a single region in which all three legs recirculate. Two distinct gaits are capable of keeping exactly two legs in stance, with a duty factor of $\frac{2}{3}$.

Figure 3.1: Phase spaces of low-dimensional gaits

(the three distinct "arms" of the obstacle set). The obstacle set for the 4- and 6-legged systems cannot be visualized but consist of similar obstacle "layers". Furthermore, some gaits preferably recirculate multiple legs together, while others do not. The alternating tripod gait is a prime example, as it allows three legs to recirculate together—an obvious incursion into the obstacle set as half of the robot's are recirculating—yet is a perfectly good, statically-stable gait. Other gaits that attempt to recirculate three legs together, however, have a high likelihood of losing static stability.

This thesis does not focus specifically on understanding the precise shape, topology or geometry of the obstacle set, but instead looks for control methods that may avoid obstacles and can be easily adapted based upon local definitions of the obstacle, i.e. which and how many legs should be allowed to recirculate together, based upon a gait's description.

3.1.2 Considerations of Controller Convergence

We now discuss another particular challenge of gait regulation control, that of how controllers must converge to desired gaits, and the implications this has upon controller properties. As we have discussed, a gait regulation controller must apply control to the space of gait timings, a high-dimensional torus, while attempting to converge onto a desired gait, a limit cycle within that torus. Preferably, following our discussion in the previous section, the controller should also avoid the $\mathcal{P}$-obstacle set, a non-trivial set embedded within the torus. Considering the topology of each of these spaces, each is of a different homotopy class (one measure of a space's topological shape). For a gait regulation controller to be successful, it must somehow overcome these differences in homotopy—between the torus, a single gait cycle, and the obstacle set—when converging, the second major challenge of gait regulation control.

While it is of utmost important to avoid the obstacle set, we will temporarily omit it from discussion, and focus directly on the homotopy mismatch between the N-torus and the cycle, and the implications this mismatch has upon a gait regulation controller.

A gait regulation controller that has global convergence to a desired gait is, in essence, a policy that directs any point within the control space down to a single limit cycle. Topologically speaking, this can be interpreted as a deformation retraction from the control space to a cycle, the retraction *squeezing* the control space down to a single limit cycle. A deformation retract is only possible, however, between spaces whose homotopy classes are

equivalent, a property not true between a single cycle and a high-dimensional torus (a product of a collection of cycles). Thus, informally speaking, a gait regulation controller must not only direct the space to converge to a single cycle, but must also place *boundary sets* on the torus, cutting the torus apart in order to match the homotopy of the cycle. In terms of a control system, these boundaries become discontinuities in control policy, often unstable equilibria from which the control system flows in multiple directions.

This idea of the homotopy mismatch between a cycle and a torus is best explained with a simple example. Consider the placement of a pointwise attractor on a cycle. According to the Poincaré-Hopf Index theorem from differential topology, a smooth vector field on the cycle cannot place a stable critical point (our pointwise attractor), without also introducing a repellor elsewhere on the cycle, an unstable equilibrium point. This unstable critical point creates a *boundary cut* on the cycle, left of which the system flows in one direction, and right of which the other, resulting in both clockwise and counter-clockwise flows to the attractor. In this case, with the unstable critical point cutting the cycle apart, the control system is performing control on an interval, rather than a cycle, thus matching the homotopy of a point. Global convergence is obtained (except for at the unstable critical point), but only with this necessary condition. This is similar to the necessity that gait regulation controllers must introduce boundary cuts on the high-dimensional torus in order to produce attractive cycles with global convergence.

3.2 Design of Attractors and Basins for Gait Regulation Control

Gait regulation controllers must necessarily place boundaries on a torus, for which we study methods using a potential field approach. As current methods for designing repellors are not capable of encapsulating the precise geometry and topology of $\mathcal{P}$-obstacles, we research the use of simple control laws that may approximate the obstacle set when laying down repellors and attractors on the torus, in order to produce convergent results while also avoiding the obstacle set.

We make use of potential functions in designing gait regulation controllers. Potential functions are common for robot navigation tasks [13], in which obstacles and goals are placed as maxima and minima of a real-valued function on the configuration space of the robot. In our case, the configuration space is the N-torus, while we attempt to place global

minima at desired gait limit cycles, and maxima to coincide with $\mathcal{P}$-obstacles. Navigation of gait phase space is performed by following the negative gradient of a potential function, Φ:

$$\Phi \quad : \quad \mathbb{T}^N \rightarrow \mathbb{R} \tag{3.1}$$

$$\dot{\mathbf{x}} \quad = \quad -\nabla_{\mathbf{x}} \Phi(\mathbf{x}) \tag{3.2}$$

The design of a potential function that satisfies all of our desired properties, such as global convergence to a preferred gait while avoiding $\mathcal{P}$-obstacles, is a challenging problem. Rather than focus on designing a complete potential function for the N-torus, we follow an approach to build pairwise functions, each a potential on the 2-torus, that we compose together to make an overall potential. By focusing on the constraints that exist within each pairwise subspace, we seek to respect constraints on the N-torus. In the case of legged robot gaits, common specifications exist between pairs of legs, such as pairs that should be in-phase with one another, or out-of-phase, and we encode these constraints using simple potential functions. These subfunctions place repellors and attractors on the lower-dimensional subspaces, with the assumption that the properties will hold when incorporated into a potential function on the N-torus.

The basic approach we follow is that of [28]: we construct a network of connections between pairs of legs, specifying whether individual legs attract or repel one another, or simply are not connected. The overall potential function on the N-torus is then defined as the summation of a collection of functions on $\mathbb{T}^2$ subspaces. Let f_a and f_r be pairwise methods of either attracting or repelling between two legs. We define binary vectors, r_a and r_r, containing the possible projected subfunctions, and activate them using binary vectors, as follows, to produce the overall potential, Φ:

$$\Phi(\mathbf{x}) \quad = \quad \begin{bmatrix} c_a^T & c_r^T \end{bmatrix} \begin{bmatrix} r_a(\mathbf{x}) \\ r_r(\mathbf{x}) \end{bmatrix} \tag{3.3}$$

$$r_a(\mathbf{x}) \quad = \quad \begin{bmatrix} f_a(\rho_1, \rho_2) & \dots & f_a(\rho_{N-1}, \rho_N) \end{bmatrix}^T \tag{3.4}$$

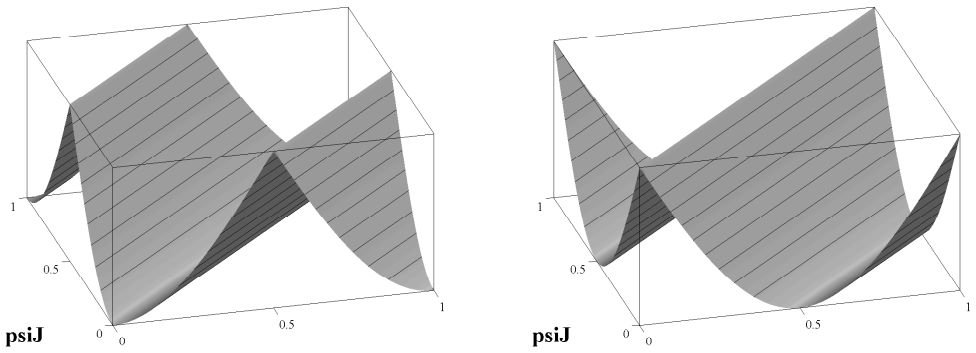
$$r_r(\mathbf{x}) \quad = \quad \begin{bmatrix} f_r(\rho_1, \rho_2) & \dots & f_r(\rho_{N-1}, \rho_N) \end{bmatrix}^T \tag{3.5}$$

We differ in two ways from the specific approach of [28]. Foremost, the component functions we use—$f_a(\rho_i, \rho_j)$ (3.6) and $f_r(\rho_i, \rho_j)$ (3.7) for attraction and repulsion, respectively—place maximal ridges and minimal valleys appropriately, but have sharper peaks than a traditional coupling function. This sharpness can be seen in Fig. 3.2, and compared to a more traditional set of attraction and repulsion functions in Fig. 3.3. While our choice of f_a and f_r makes the system non-differentiable at maximal ridges, the sharpness greatly improves convergence, as systems move rapidly away from ridges.

$$f_a(\rho_i, \rho_j) = 1 - \| \sin(\pi (\rho_i - \rho_j + 0.5)) \| \tag{3.6}$$

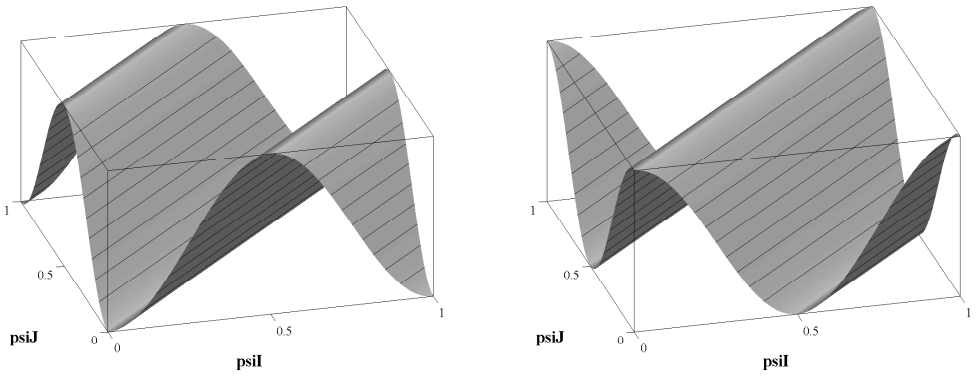
$$f_r(\rho_i, \rho_j) = 1 - \| \sin(\pi (\rho_i - \rho_j)) \| \tag{3.7}$$

The second way in which we differ from [28] is in our use of binary vectors, $c_a, c_r \in \{0,1\}^{\binom{N}{2}}$, to specify the activation of either attraction or repulsion between pairs of legs (as there are $\binom{N}{2}$ possible pairings of legs). While there is the assumption that two legs can either attract or repel each other, but not both, we can transform this representation into the connection matrices of [28] as follows. Let M be the symmetric connection matrix defining



(a) (b)

Figure 3.2: As in (3.3), potential functions are defined as a summation of subfunctions, each on the 2-torus. The two functions shown here perform attraction and repulsion. Minima appear along limit cycles at desired relationships between legs. While non-differentiable exactly along the maximal ridges, these function produce smooth gradients that quickly move away from the ridges and converge upon the limit cycle.



(a) (b)

Figure 3.3: Traditional subfunctions to enable repulsion and attraction between leg pairs. Without the sharpness we introduce in Fig. 3.2, the gradients of these functions flatten out near the ridges, resulting in slow rates of convergence when close to the ridges.

the system, with element $M_{i,j}$ one of three values: 0, no connection between leg i and j; -1, repulsion between legs i and j; or $+1$, attraction.

$$c_k = c_a - c_r = \begin{bmatrix} k_1 & k_2 & \dots & k_{\binom{N}{2}} \end{bmatrix} \tag{3.8}$$

$$M = \begin{bmatrix} 0 & k_1 & k_2 & \dots & k_N \\ k_1 & 0 & k_{N+1} & \dots & k_{2N-2} \\ k_2 & k_{N+1} & \ddots & & \vdots \\ \vdots & \vdots & & 0 & k_{\binom{N}{2}} \\ k_N & k_{2N-2} & \dots & k_{\binom{N}{2}} & 0 \end{bmatrix} \tag{3.9}$$

Our choice of a terser representation comes from the fact that our proposed algorithmic approach works as a function over possible values of the binary vector, c_r, studying the ways in which the $\binom{N}{2}$ functions of $f_r(\rho_i, \rho_j)$ affect controller convergence to a variety of gaits.

3.3 A Hybrid Approach for Gait Regulation

We develop an approach for gait regulation that allows us to design convergence properties for various desired gaits, while also attempting to avoid phase space obstacles. The core of the approach comes from utilizing repulsion between leg pairs as a means of approximating the obstacle set, while also using a cellular decomposition of phase space to algorithmically reconnect regions of phase space to converge to desired gaits. The end result of our approach is a vector field control policy that produces global convergence to certain desired gaits.

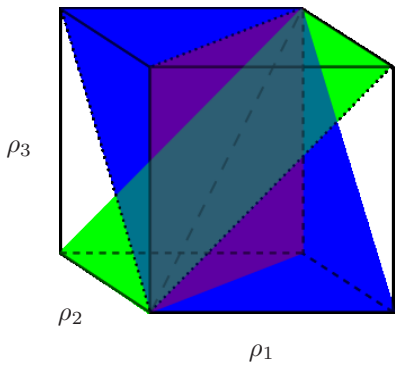
The principal contribution of this section is the hybrid control algorithm, which, by noting the connectedness of the cells making up our decomposition of the torus, allows us to design convergence properties for various gait types, preventing local minima while also attempting to avoid phase space obstacles.

3.3.1 Leg Repulsion as Approximation of Phase Space Obstacles

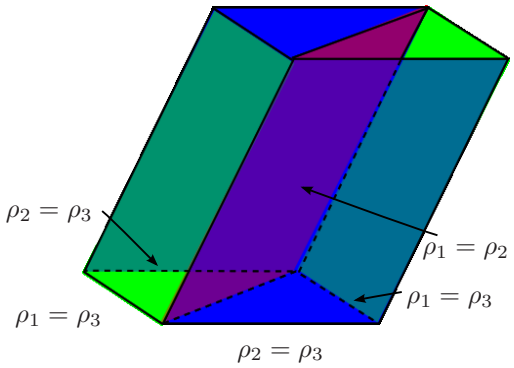
While we present both pairwise attraction and repulsion as recipes for designing gait regulation controllers, we argue that leg repulsion is the more useful of the two. By placing repulsive sets along the diagonal subspace of the torus—the subset of the torus in which two or more legs are in phase with one another—we also avoid regions where multiple legs recirculate together, the $\mathcal{P}$-obstacle set. While not a precise definition of the obstacle set, we show that this strategy is a suitable approximation for our control tasks: two legs in phase with one another will inevitably lead to two legs recirculating together. Thus, avoiding the former will aid in also avoiding the latter.

We study the 3-torus, as previously shown in Fig. 3.1b, in a slightly different perspective to show this approximation. Fig. 3.4 shows this alternative view. A repulsion controller, placing ridge lines using the $f_r(\rho_1, \rho_2)$, $f_r(\rho_1, \rho_3)$ and $f_r(\rho_2, \rho_3)$ functions, repels from these diagonal planes.

Gaits, with evenly increasing flow as represented in Fig. 3.1b, are cycles parallel to the vector $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$. We study a 2-torus subspace, a Poincaré section perpendicular to this vector to analyze the intersection of various gaits with the $\mathcal{P}$-obstacle set. We furthermore compare against values of a simple potential function designed from pairwise leg repulsion. Fig. 3.5a show a numerical integration of the obstacle set along the vector $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$. Note



(a) The 3-torus, with diagonal set noted. The three planes making up the diagonal set, $\rho_1 = \rho_2$, $\rho_1 = \rho_3$, and $\rho_2 = \rho_3$ are the repellors in an all-repulsive control arrangement.



(b) The cube is just one possible representation of the 3-torus. This figure shows a different arrangement, using the diagonal planes $\rho_2 = \rho_3$ and $\rho_1 = \rho_3$ to make up four of the identified faces of a torus representation

Figure 3.4: The 3-torus, visualized using the diagonal space of the torus

that two minima occur at points in the center of each "triangle"; these minima are the same as the two non-intersecting gaits shown in Fig. 3.1b.

Fig. 3.5b shows a similar plot, the values of the potential function:

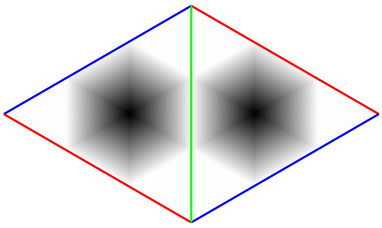
$$\Phi_{crawl3} = c_r^T r_r(\mathbf{x}) \tag{3.10}$$

$$\Phi_{crawl3} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_r(\rho_1, \rho_2) \\ f_r(\rho_1, \rho_3) \\ f_r(\rho_2, \rho_3) \end{bmatrix} \tag{3.11}$$

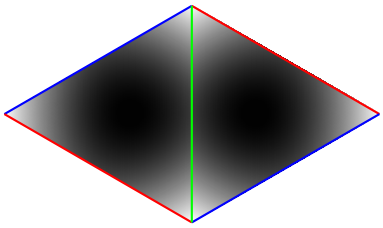
$$\Phi_{crawl3} = f_r(\rho_1, \rho_2) + f_r(\rho_1, \rho_3) + f_r(\rho_2, \rho_3) \tag{3.12}$$

This potential function seeks to avoid the diagonal set of the 3-torus, thus avoiding the exterior walls of this 2-torus subspace. Just as the left plot shows two minima located at the two gaits that do not intersect the obstacle set, Φ_{crawl3}, produces the same local minima at these gaits, with all legs out-of-phase. While the numerical integration of the obstacle set is an expensive process, the potential function is easily calculated.

As we move to higher dimensions, our use of repulsion between leg pairs provides us with an easy means of approximating the complex $\mathcal{P}$-obstacle set. As we discuss gait regulation for gaits such as the trot or alternating tripod, in which some legs are preferably



(a) Numerical integration of the obstacle set of the 3-torus, projected onto a 2-torus subspace. Light areas indicate strong intersection with the obstacle set, while dark indicate no intersection.



(b) Repulsion between pairs of legs, a gait regulation potential function. This function attempts to avoid the diagonal (the "walls" of the triangles), thus producing minima in the centers of each triangle.

Figure 3.5: A comparison of the $\mathbb{T}^3$ obstacle set, as projected onto a 2-torus subspace, with a simple fully repulsive potential function.

in phase with one another, we use leg repulsion to define which legs should explicitly not be in phase in order to approximate the phase space obstacle.

3.3.2 Phase Space Connectivity via Cellular Decomposition

The diagonal subspace, the set of repellors when we activate repulsion between all leg pairs, induces a cellular decomposition of our high dimensional torus. We use this cellular decomposition to discretely compute distances between gaits of interest, via a related graph representation. Precomputation of distances between cells informs us of potential local minima when performing gait regulation tasks for certain gait types.

Appendix B introduces the cellular decomposition we use throughout this thesis, as well as a simple graph representation of the cellular complex. As discussed in the appendix and empirically shown in simulation (§4.2), the cells of this complex surround the $(N-1)!$ crawl gaits for an N-dimensional system. The important definitions made in the appendix are:

C_k : one cell of our phase space decomposition, corresponding to crawl gait k, with the decomposition consisting of a total of $(N-1)!$ different cells for the various crawl gaits.

v_k : in a related graph representation of the complex, v_k is the graph vertex associated with a certain cell.

e_{ij} : if cells C_i and C_j are adjacent to one another, the graph representation contains an edge between v_i and v_j.

We consider cells to be *adjacent* whenever they share a face. These faces separate cells from one another, are a portion of the diagonal subspace of the torus, and correspond to a single pair of legs being in-phase with one another. The concept of matching faces with the various cells is described in full in Appendix B, where we also describe how to translate these cells and faces into vertices and edges of a graph.

The cellular decomposition of $\mathbb{T}^N$ is of critical importance in understanding gait regulation when using repulsion between leg pairs. As leg repulsion places repellors along the diagonal subspace of a torus, wherever two or more legs are in phase, the cellular decomposition describes how these repellors cut the torus apart and separate the cells. As edges

on a graph representation correspond to adjacency between cells, we compute distances on the graph to tell us how many repellor sets separate any two cells from one another.

Removing repellors between cells, for a gait regulation controller, allows us to "glue" together groups of cells into larger regions of phase space. The cellular decomposition, with its description of adjacency amongst cells, describes the effect of this process. This is particularly useful for gaits containing in-phase leg pairs, such as the trot or alternating tripod, as these gaits exist as cycles along the diagonal subspace of the torus, on the boundaries between cells of our complex. We first note that simply removing a repellor that separates any two cells—removing the specifications that two legs should repulse one another—results in the cells merging together at their adjacent face, with no boundary set between them anymore. In this fashion, gluing together a collection of neighboring cells allows us to design large regions of phase space that surround gaits of interest, with the gaits, normally existing on boundaries between cells, sitting in the interior of the union of some cells and their adjacent faces.

Thus, we define a gait regulation controller for gaits such as the trot or alternating tripod by removing repulsion subfunctions from an all-repulsive potential function. Rather than explicitly stating which legs should be in-phase with one another (via the attraction subfunction, $f_a(\rho_i, \rho_j)$), we simply tell the system not to keep the pair of legs out-of-phase with another (by omitting a repellor between the leg pair). For our cell complex, this operation reconnects certain adjacent cells together, resulting in regions we hope will converge to the desired gait.

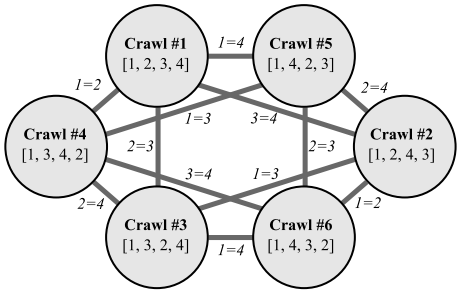
As we shall show with a simple example, these steps are not sufficient, however, to producing global trot convergence. Let us consider the cell decomposition of $\mathbb{T}^4$, a collection of 6 cells that result from the six component repulsive functions

$$r_r = \begin{bmatrix} f_r(\rho_1, \rho_2) \\ f_r(\rho_1, \rho_3) \\ f_r(\rho_1, \rho_4) \\ f_r(\rho_2, \rho_3) \\ f_r(\rho_2, \rho_4) \\ f_r(\rho_3, \rho_4) \end{bmatrix} \tag{3.13}$$

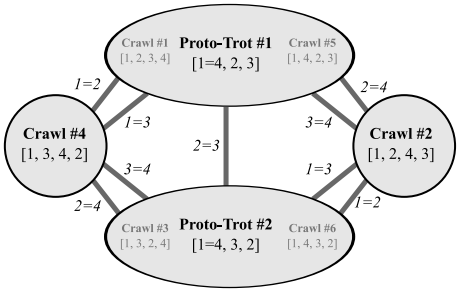
These subfunctions produce the cellular decomposition by repelling from the diagonal subspace of $\mathbb{T}^4$, preventing any two legs from being in-phase.

When we remove certain repulsive functions, $f_r(\rho_1, \rho_4)$ and $f_r(\rho_2, \rho_3)$, certain cells are no longer separated, and become glued to one another. Fig. 3.6 shows the graph representation of the cell complex as we successively remove the two repellor sets. The end result shown in the graph merging process is that two vertices remain disconnected from the trot gait region. Thus, removing just two repulsive functions is not sufficient to fully glue the cells together and produce global convergence.

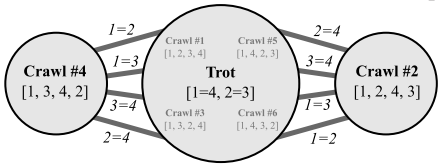
The connectedness of the vertices shown in Fig. 3.6c is very important to understand. The crawl gait cells that merge together (as represented as merged vertices) are considered



(a) Initially, all six vertices are separated by the various repellors sets. Each vertex corresponds to a distinct crawl gait leg ordering. Edges of the graph correspond to faces between cells, swaps in leg order.



(b) By removing the repellor set between legs 1 and 4, we glue the cells together along their adjacent face, thus joining vertices together via the edge between them in our graph representation.



(c) By removing an additional repellor set between legs 2 and 3, a total of four cells are glued together to build a convergent region around the trot gait.

Figure 3.6: Graph representation of the cellular decomposition for $\mathbb{T}^4$. We show the sequence of steps that occur when repellor sets are removed from a gait regulation function, how vertices merge together as we glue cells of the complex. Two vertices remain disconnected however, from the merged set, due to the remaining repulsive functions. The figure uses the tuple representation of leg orderings, as introduced in Appendix A.

to be 0-connected to the trot gait. Using the tuple representation of leg orderings (App. A), the leg ordering $(1, 2, 3, 4)$—crawl cell #1—is compatible with the trot gait $((1, 4), (2, 3))$, since no legs need to swap order for the system to converge. If our system is within crawl cells #2 and #4, however, two legs must necessarily swap order in order for the system to converge to a trot. We consider these cells to the 1-connected to the trot gait, meaning that one edge must be crossed before convergence.

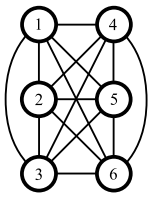
In general, we consider a cell to be k-connected if there are a minimum of k edges to be passed from its vertex to the convergent region. While the graph representation—and merging process—of something like the trot gait may seem trivial, given the 6 cells resulting from the cellular decomposition of $\mathbb{T}^4$, this process is extremely critical when we begin to consider the 120 cells of the 6-torus, $\mathbb{T}^6$.

3.3.3 Real-Time Algorithm

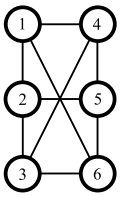
We use the connectivity amongst cells to devise a simple hybrid control approach that algorithmically reconnects cells, whenever necessary, that are disconnected from the convergent region of a desired gait. This is done by temporarily disabling repellor sets that may prevent the system from converging. As there are often many routes by which the system can choose to converge, we show how the use of distance heuristics, while searching on the discrete graph representation, introduces decision surfaces that evenly distribute the paths of convergence from distant regions of phase space. Given a simple algorithm on our possible control laws, combined with modest amounts of precomputation, this approach is readily applicable to real-time robotic systems.

The first step in preparing a gait regulation controller is defining which legs should be out-of-phase with one another in the eventual limit cycle gait. This information is sufficient to realize a variety of useful robot gaits. This information can be defined in a connection diagram, or in vector form, such as by c_r. For a trot, as discussed above, we have the initial arrangement, in which two leg repulsions are removed, allowing the system to (locally) converge to a trot gait.

$$\text{Trot: } c_r^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.14}$$



(a) The original crawl gait regulation system for a hexapedal robot. All lines of coordination refer to repulsion between legs, generating an overall potential function to produce crawl gaits.



(b) By removing six of the repulsion component functions, we allow (local) convergence to an alternating tripod gait.

Figure 3.7: Design of the basic alternating tripod gait regulation controller

For the 6-torus, we consider the alternating tripod gait, $((1,3,5),(2,4,6))$ in leg ordering tuple notation (App. A), or alternatively $\mathbf{x} = \begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}$. With many legs to be in-phase, we disable repulsions between leg pairs $(1,3)$, $(1,5)$, $(3,5)$, $(2,4)$, $(2,6)$, and $(4,6)$. Fig. 3.7 shows a graphical representation of deactivating these 6 repulsions functions. The binary vector representation of leg repulsion, as well as the connection matrix, M, are:

$$c_r = [1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1\,1\,0\,1] \tag{3.15}$$

$$M = \begin{bmatrix} 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \end{bmatrix} \tag{3.16}$$

With a definition of which repellors to initially remove, thus allowing cells in our decomposition to merge, we compute the connectivity of the graph cells. This process has already been shown for the trot gait, as in Fig. 3.6, reducing to a total of four 0-connected cells and two 1-connected. In the case of the alternating tripod, in which the decomposition contains a total of 120 cells, the connectivity result is:

36 0-connected cells

72 1-connected cells

12 2-connected cells

If we were to activate the remaining repulsion functions for the tripod gait, we expect local minima to occur, with a large number of cells that are disconnected via repellor sets. The task of defining a global gait regulation controller is to remove all local minima, guaranteeing safe convergence from anywhere on the torus.

Besides the possibility of having certain cells distant on the torus from our desired gait, we also have the question of convergence paths. A k-connected cell may have a number of different adjacency relationships leading to the 0-connected set. For this reason, we utilize simple heuristics to choose a path along the graph based upon distances between the legs that must swap order for convergence to occur. This metric adds decision surfaces at the furthermost regions of the torus, evenly partitioning the distant regions when constructing paths of convergence. In the example of the $\mathbb{T}^4$ trot, any one of four different repellors may be deactivated in order to converge, thus a decision of which is "closest" produces symmetric convergence results.

These two primary concepts, the connectedness of a cell and the determination of decision surfaces, are incorporated into an algorithmic approach for gait regulation control, shown in Procedure 1. The algorithm initially determines the cell the current gait is within, and fetches the precomputed connectivity of the cell. If within a 0-connected cell, the system will naturally converge with the nominal set of leg repulsions, thus no action is taken. If, however, within a k-connected cell, a total of k repulsion functions must be removed for us to "glue" our distant cell to the central convergent region.

Since each edge of the cell graph represents a swap between two legs, we compute the phase difference between legs as a distance heuristic to score various routes on the graph. While one heuristic guarantees that each possible path leads from a k-connected to a $(k-1)$-connected cell, this second metric heuristic compares phase differences. The routine CUTDISTANCES computes the sum of distances to a set of cuts, computed via (3.20), and is used to choose the shortest route.

Procedure 1 Gait Regulation via Determination of Cut Removal

1: **procedure** GAITREGULATION(d,$\mathbf{x}$)

Require: d is the desired gait, $\mathbf{x}$ is the set of current phase offsets

2: $k \leftarrow$ CRAWLCELLNUMBER($\mathbf{x}$)

3: $c_r \leftarrow$ INITIALIZECUTS(d)

4: $c_r \leftarrow$ CHOOSECUTS(d,k,c_r,$\mathbf{x}$)

5: **end procedure**

6: **procedure** CHOOSECUTS(d,k,c_r,$\mathbf{x}$)

7: $j \leftarrow$ CELLCONNECTEDNESS(d,k)

8: **if** $j = 0$ **then**

9: $d_{min} \leftarrow 0$

10: **return** (c_r,d_{min})

11: **end if**

12: $d_{min} \leftarrow \alpha$, where α is sufficiently large

13: **for** $k_{next} \in$ CONNECTIONS(k) **where** CELLCONNECTEDNESS(d,k_{next}) $< j$ **do**

14: (c_{new},d_{new}) $\leftarrow$ CHOOSECUTS(d,k_{next},c_r,$\mathbf{x}$)

15: $d_{new} \leftarrow d_{new} +$ CUTDISTANCE(k,k_{next},$\mathbf{x}$)

16: **if** $d_{new} < d_{min}$ **then**

17: $d_{min} \leftarrow d_{new}$

18: $c_{best} \leftarrow$ REMOVECUT(c_{new},k,k_{next})

19: **end if**

20: **end for**

21: **return** (c_{best},d_{min})

22: **end procedure**

$$d : \mathbb{T}^2 \to [-0.5, 0.5) \tag{3.17}$$

$$d(\rho_i, \rho_j) = \mathrm{mod}(\rho_i - \rho_j + 0.5, 1.0) - 0.5 \tag{3.18}$$

$$d_{abs} : \mathbb{T}^2 \to [0, 0.5] \tag{3.19}$$

$$d_{abs}(\rho_i, \rho_j) = |d(\rho_i, \rho_j)| \tag{3.20}$$

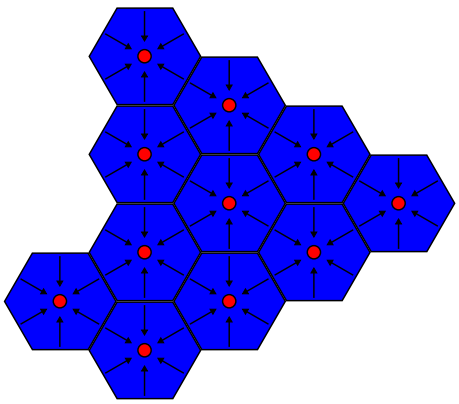
Upon choosing a route, the algorithm then deactivates the necessary repellors, gluing the disconnected cells of the complex back with the large set of cells, upon which the system converges. By continuously running this algorithm, the system adapts to passing into closer regions of convergence, and adapts to unforeseen changes in gait. Fig. 3.8 provides an abstract overview of this approach.

Returning briefly to our discussion of phase space obstacles, as well as boundary cuts on the torus, the control shown provides solutions for both. Since our gait regulation system is based solely upon repulsion between leg pairs, the $\mathcal{P}$-obstacles are represented approximately in the resulting control laws. By moving from crawl gait cell to crawl gait cell while converging from distant regions, the system attempts to keep as many legs out of phase with one another until reaching the final convergent region. In terms of placing the necessary boundary cuts to achieve convergence, our use of the cuts placed by repulsions between leg pairs, as well as decisions surfaces, introduce the necessary boundaries, thus producing convergence as required.

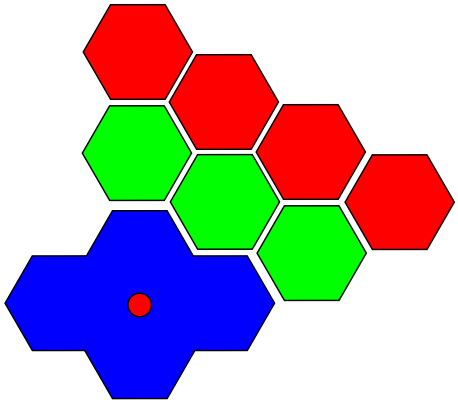
3.3.4 Management of Multiple Basins of Attraction

One unique advantage of our hybrid control algorithm—using both discrete information from a gait graph as well as metric distances on the torus—is the ability to add higher levels of hybrid switching. In this section, we discuss our ability to activate multiple gait regulation controllers concurrently, incorporating additional decision surfaces to evenly divide phase space amongst controllers.

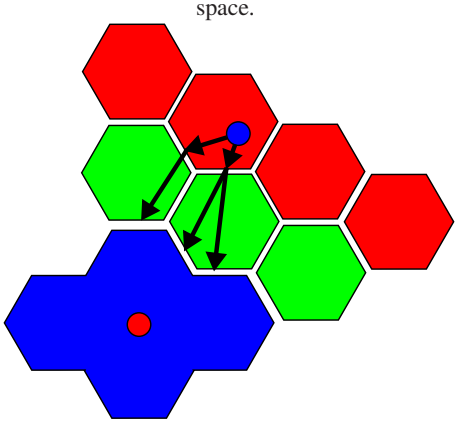
The tetrapod gaits are those that recirculate two legs at a time, grouping a hexapod's legs into three groups of two. Considering only those gaits that do not recirculate ipsilateral or contralateral leg pairs together, there exists two different arrangements for a tetrapod controller, with each controller realizing a "forward to back" gait and a "back to forward"



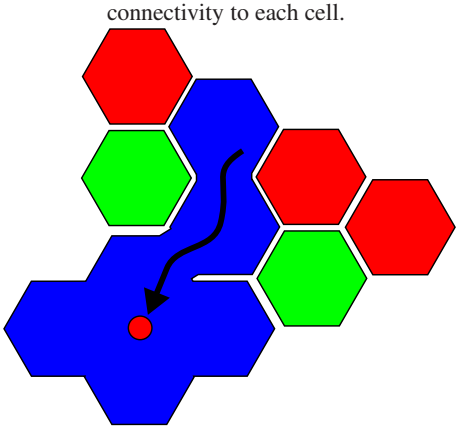
(a) We begin with a cell decomposition of phase space.



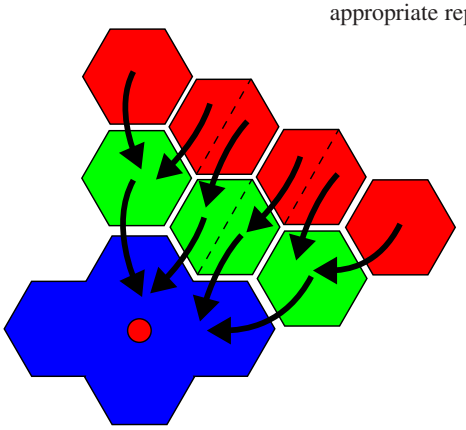
(b) By noting convergent regions, we assign connectivity to each cell.



(c) Given robot state in a distant cell, various routes to the convergent region are compared.



(d) The minimum distance route, as determined by phase differences, is followed, with appropriate repellors deactivated.



(e) The end result is a global vector control policy that achieves gait regulation for the desired gait, using only leg repulsion.

Figure 3.8: Abstract representation of hybrid gait regulation control approach

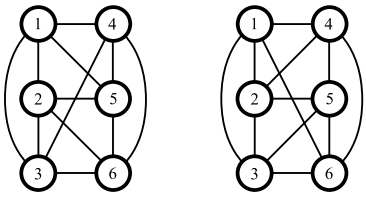
gait (for a total of 4 gaits). Fig. 3.9 shows the basic repulsion functions used to realize the two different controllers, and the resulting gait timings.

Upon computing the connectivity of the 120 cells, each controllers contains a count of $(16, 48, 48, 8)$ cells as $(0, 1, 2, 3)$-connected. When considering the union of the two controllers, and assuming that, from a given cell, you would use the nearer of the two in terms of graph distance, the connectivity is $(40, 66, 24)$. This includes 44 cells for which the two controllers have the same connectivity. If such a case occurs, we use the CUTDISTANCES metric to calculate which of the two gait arrangements is actually closer, thus introducing a metric decision surface to choose the gait.

3.4 Concluding Remarks

While methods have been proposed that coordinate the legs of a robot to converge upon preferred gait timings (§1.2.3), our approach studies the underlying topology of the space, the necessary implications of control, and devises a novel approach that uses only pairwise leg repulsions to converge to a variety of gaits.

By studying the effect that pairwise leg repulsion has upon the space of gait timings, the N-torus, we use a related cellular decomposition of the space to inform a hybrid switched control algorithm that selectively activates and deactivates certain repulsion functions in order to guarantee convergence to preferred gaits, while avoiding regions of the phase space that correspond to dangerous gait timings.



(a) Tetrapod controller basis 1

(b) Tetrapod controller basis 2

Figure 3.9: Coordination diagrams for the basis of each tetrapedal gait regulation system. By designing gait regulation around both systems, simultaneously, we realize four separate tetrapedal limit cycles.

Chapter 4

Numerical Simulations of Gait Regulation

In this chapter we introduce a large variety of control techniques, in addition to the hybrid control approach of §3.3, and analyze the ability of each to converge to desired gaits and avoid phase space obstacles, while producing control laws for a variety of gaits—crawl, trot, alternating tripod, and tetrapod gaits—with the potential application of the controllers to both quadruped and hexapod robots. In our results, we show that the switched hybrid control approach, with its basis in a cellular decomposition that approximates the obstacle sets, avoids obstacles dramatically better than prior methods.

4.1 Methodology

To study gait regulation, we utilize numerical simulations of oscillator systems, as defined by the phase offsets representing legs in a robot gait. To keep simulations aligned with the application of gait regulation to a robot, each system only applies gait regulation control during leg recirculation, assuming stance legs to be fixed on the ground for some period of the stride. By simulating different gait regulation controllers from a variety of random initial conditions, we note convergent gaits and properties for each system.

While critical point analysis may be used to determine the existence of local minima for potential functions, numerical simulations allow us to perform the same analysis, en masse, while also empirically deriving other properties of convergence, such as avoidance of $\mathcal{P}$-obstacles. Numerical simulations are further motivated by our use of *sharp* potential functions, designed to sharply repel legs from certain configurations, but with the side-effect of introducing discontinuities in derivatives of the potential function, precluding classical analysis techniques.

To simulate each system, a fixed timestep, first-order integrator runs until convergence is reached at a set of phase offsets, the *convergent gait* of the simulation. By recording the gaits produced, we compute gait convergence probabilities for each controller system.

To compare the obstacle-avoidance properties for each system, we note when the simulations, with defined regions of stance and recirculation, attempt to recirculate multiple legs together. As the precise definition of $\mathcal{P}$-obstacles is challenging (§3.1.1), we use the following metrics to define obstacle sets for quadrupeds:

3-legs For a quadrupedal robot, recirculating three or more legs together is always considered an obstacle.

2†-legs While we allow trot gaits to recirculate pairs of legs together, any other recirculation of two legs is an error, by design.

For hexapods, we define similar metrics:

4-legs Of six legs, recirculating four or more is always considered to be within the $\mathcal{P}$-obstacle set.

3†-legs As we study alternating tripod gaits, any recirculation of three legs not in the specified tripods (1,3, and 5; 2,4, and 6) is also considered an obstacle.

By recording incursions into these obstacle sets during the first five strides of a simulation, we compare the differing abilities of controllers to avoid phase space obstacles.

Quadruped simulations are performed with a duty factor of $\delta = \frac{3}{4}$, while hexapods use $\delta = \frac{5}{6}$. These duty factors are chosen to reflect the nominal and minimum duty factor possible when using a crawl gait for each set of 4 or 6 legs. Fig. 4.1 shows an example of one simulation run, a trot controller. In the example, the system begins at a random set of phase offsets, while gait regulation is performed during leg recirculation to converge to a trot gait. The diagonal shaded regions indicate the portion of phase, at each point of time, currently dedicated to stance. Legs within stance, thus, have constant-valued phase offsets.

4.2 Crawl Gaits

Simulations show that crawl gait regulation can be easily obtained using pairwise leg repulsion, achieving a variety of convergent gaits while naturally avoiding phase obstacles.

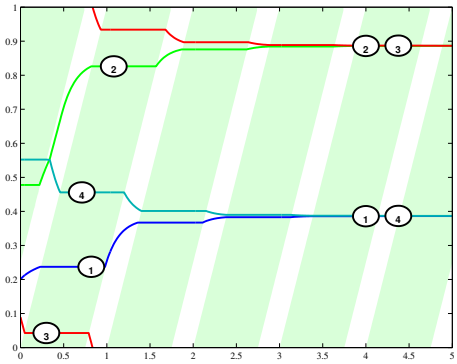
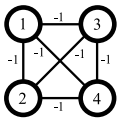


Figure 4.1: Example gait regulation simulation. Stance regions are shaded, during which no gait regulation takes place. During recirculation, legs are allowed to speed up or slow down to match the gait timing of a desired gait.

.

Crawl gaits, such as those discussed in §2.3, attempt to recirculate only a single leg at a time by keeping all legs out-of-phase with one another. As crawl gaits form the basis for our hybrid control approach to gait regulation (§3.3), we show how the crawl gait basins of attraction coexist in phase space, activated by a single policy of repulsing between all $\binom{N}{2}$ pairs of legs.

With no switching or hybrid control necessary, we produce crawl gaits by enabling repulsion between leg pairs, keeping all legs out-of-phase with one another. The gait regulation system shown in (4.3), with its associated connection diagram [28], does this with a constant control policy. As the existence of crawl gaits form the cellular decomposition used in our hybrid control approach, it is interesting to note how the crawls naturally partition phase space into multiple regions of convergence.



Crawl-4

$$c_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.1}$$

$$c_r = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.2}$$

$$\begin{aligned} \Phi_{crawl4}(\mathbf{x}) = {} & f_r(\rho_1,\rho_2) + f_r(\rho_1,\rho_3) + f_r(\rho_1,\rho_4) + \\ & f_r(\rho_2,\rho_3) + f_r(\rho_2,\rho_4) + f_r(\rho_3,\rho_4) \end{aligned} \tag{4.3}$$

The pairwise potentials included in (4.3) are designed to place repelling fields in the neighborhood of the pairwise diagonal subspaces of $\mathbb{T}^6$, leg configurations where any pair of legs have equal phase. The net effect results in equal distribution of 6 different crawl

gaits for a quadrupedal robot, shown through our simulation results (Table 4.1). Each crawl gait consists of one cyclic ordering of four legs, legs separated evenly by $\frac{1}{4}$ phase. While the repulsion functions design the "ridgelines" of the potential to be avoided, the system naturally produces convergent gaits within nearby valleys (and not necessarily coincident with valleys of the individual repulsion functions).

Fig. 4.2 shows trace plots for a handful of starting conditions for a quadruped. In each case, the system settles at the nearest crawl gait dependent upon the initial random order of the four legs. Tables 4.1-4.2 note the convergence probabilities, boundary crossings, and obstacle avoidance for the system, when studying properties en masse across all 5000 simulations.

We also produce crawl gaits for 6-legged systems, including a total of $\binom{6}{2} = 15$ pairwise repulsions in another crawl gait controller (4.6). This system converges upon one of 120 different crawl gaits that exist for a hexapod, with relatively uniform regularity. Fig. 4.3 show a sample of simulation results.

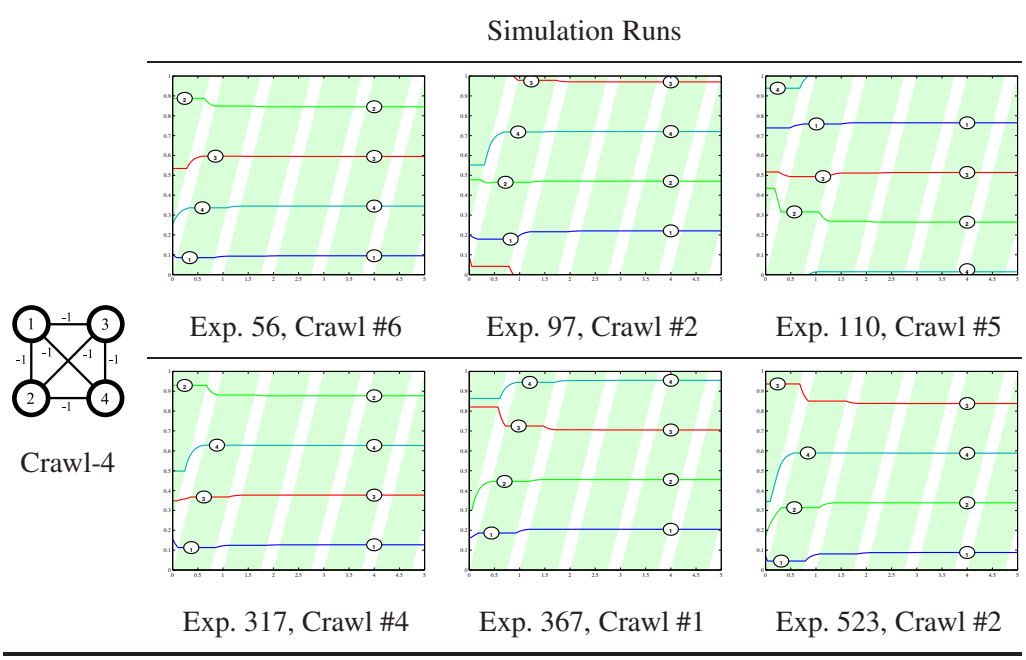


Figure 4.2: Simulation runs for Φ_{crawl}. Each simulation shows a plot of leg phase offsets over time. Modification to leg phase occurs only during leg recirculation, with legs remaining constant during stance, the highlighted regions

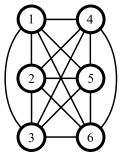
.

Table 4.1: Probabilities of convergence for crawl gaits of a quadrupedal system, collected over 5000 individual simulations. Phase offsets of the convergent gait are normalized ($\rho_1 = 0$).

Controller	**Convergent gait**	**Convergence probability**
Φ_{crawl}	Crawl #1, $\begin{bmatrix}0 & 0.25 & 0.5 & 0.75\end{bmatrix}$	16.7%
	Crawl #2, $\begin{bmatrix}0 & 0.25 & 0.75 & 0.5\end{bmatrix}$	16.9%
	Crawl #3, $\begin{bmatrix}0 & 0.5 & 0.25 & 0.75\end{bmatrix}$	16.2%
	Crawl #4, $\begin{bmatrix}0 & 0.75 & 0.25 & 0.5\end{bmatrix}$	16.9%
	Crawl #5, $\begin{bmatrix}0 & 0.5 & 0.75 & 0.25\end{bmatrix}$	16.7%
	Crawl #6, $\begin{bmatrix}0 & 0.75 & 0.5 & 0.25\end{bmatrix}$	16.5%
		Total: 100%

Table 4.2: Diagonal subspace crossings and obstacle avoidance for Φ_{crawl}. In a perfect oscillator simulation, all probabilities here would be zero. Interaction between the stance and recirculation restrictions on gait regulation, however, introduce some amount of noise, seen here as extremely low probabilities of crossing.

Controller	$\rho_1 = \rho_2$	$\rho_1 = \rho_3$	$\rho_1 = \rho_4$	$\rho_2 = \rho_3$	$\rho_2 = \rho_4$	$\rho_3 = \rho_4$	2†-leg obstacle	3-leg obstacle
Φ_{crawl}	0.04%	0.10%	0.14%	0.08%	0.04%	0.04%	2.23%	0.11%



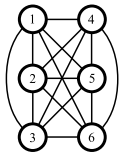
Crawl-6

$$c_a = \mathbf{0} \tag{4.4}$$

$$c_r = \underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}_{\text{15-vector}} \tag{4.5}$$

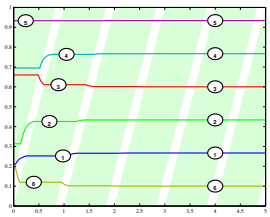
$$\Phi_{crawl6}(\mathbf{x}) = f_r(\rho_1, \rho_2) + f_r(\rho_1, \rho_3) + \ldots + f_r(\rho_5, \rho_6) \tag{4.6}$$

Upon analysis of 5000 simulation runs, the system converges to each of the 120 crawl gait limit cycles, without any other convergent gaits found. On average, each gait is convergent $1/120 \approx 0.833\%$ of the time, with relatively little variance (maximum probability of converging to a crawl gait: 1.16%, minimum: 0.56%). This small variance ($\sigma^2 = 0.0172$), relative to the sample size of 5000, indicates a probable equal likelihood for each crawl gait. Furthermore, the avoidance of obstacle sets, as shown in Table 4.2, is effective, only intersecting with the obstacle sets in very small amounts.

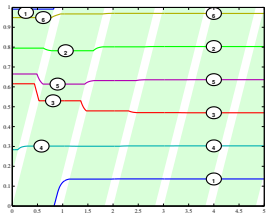


Crawl-6

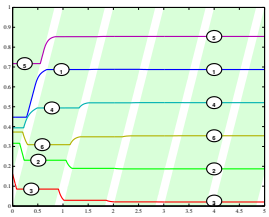
Simulation Runs



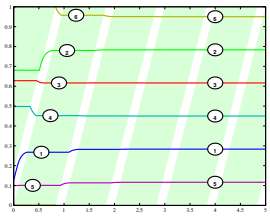
Exp. 39, Crawl #1



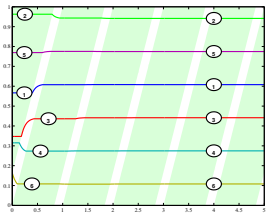
Exp. 96, Crawl #57



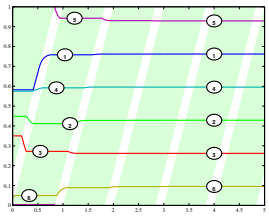
Exp. 326, Crawl #80



Exp. 554, Crawl #56



Exp. 951, Crawl #78



Exp. 1036, Crawl #93

Figure 4.3: Simulation runs for Φ_{crawl6}

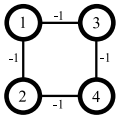
4.3 Regulating a Trot Gait

We test a variety of control methods for a trot gait, the quadrupedal gait with diagonal pairs of legs in-phase with one another. Simulation results show that our hybrid control approach is necessary to fairly partition the phase space of a trot, and compare our approach with several related trot controllers to discuss issues related to local minima, global convergence, and avoidance of phase space obstacles. We show the switched control method to have global convergence as well as excellent avoidance of phase obstacles.

4.3.1 Hybrid Switching Control

We achieve trot gaits in a similar fashion to the crawl gaits, only specifying pairwise leg repulsions to realize a trot. Using the graph reduction of the previous chapter, we show that local minima do occur for a basic trot controller, with fixed policy. By incorporating *decision surfaces* to evenly partition any occurrences of local minima, we achieve global convergence to a trot when using our hybrid control approach.

The most basic trot system, Φ_{rtrot} in (4.9), closely resembles crawl gait regulation (4.3), with the exception of disabling repulsion between leg pairs $(1, 4)$ and $(2, 3)$ as these are the in-phase leg pairs of a trot gait. Simulations for this system are shown in Fig. 4.4. While mostly converging to a trot, local minima exist where the system occasionally converges to one of the two circular crawl gaits.



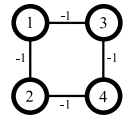
R-Trot

$$c_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

$$c_r = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.8}$$

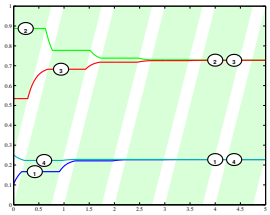
$$\Phi_r(\mathbf{x}) = f_r(\rho_1, \rho_2) + f_r(\rho_1, \rho_3) + f_r(\rho_2, \rho_4) + f_r(\rho_3, \rho_4) \tag{4.9}$$

Our design of Φ_{rtrot}, through its selection of component pairwise repulsions utilized, prevents certain leg pairs from crossing in phase space. Boundaries are introduced that seem to disconnect the N-torus, and can possibly result in local minima, as we see in simulations. In the case of a trot by repulsion, additional simple simulations can show us that by removing just one additional cut of the four in the previous example, using only three subfunctions and thus introducing one less boundary on the torus, we can achieve global

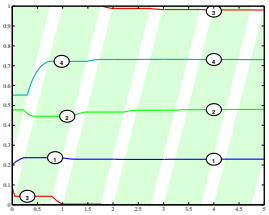


R-Trot

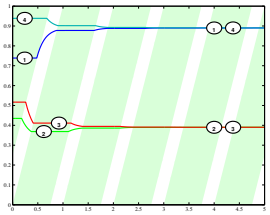
Simulation Runs



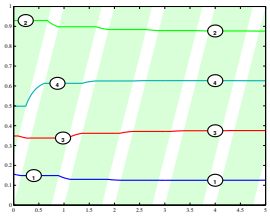
Exp. 56, Trot



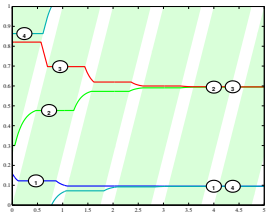
Exp. 97, Crawl #2



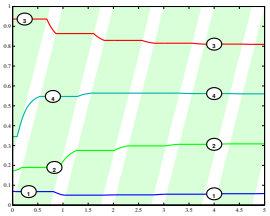
Exp. 110, Trot



Exp. 317, Crawl #4



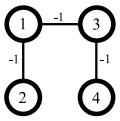
Exp. 367, Trot



Exp. 523, Crawl #2

Figure 4.4: Simulation runs for Φ_{rtrot}. This system converges most of the time to a trot gait, with notable exceptions of converging to local minima, such as in experiments 97, 317 and 523.

convergence. Φ_{grtrot} is one such function, removing one of the cuts from the previous example, with simulation results seen in Fig. 4.5.



GR-Trot

$$c_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.10)$$

$$c_r = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.11)$$

$$\Phi_{gr}(\mathbf{x}) = f_r(\rho_1, \rho_2) + f_r(\rho_1, \rho_3) + f_r(\rho_3, \rho_4) \quad (4.12)$$

Φ_{grtrot} introduces a preference to the convergence, however, as it explicitly chooses which legs should cross to produce global convergence. It can be shown that removing any other of the four repulsion functions—individually removing the cut between leg pair (1,2), (1,3), or (3,4)—also produces global convergence, with differing preferences of convergence.

Our third system for trot gait regulation uses the hybrid control approach of the previous chapter. By using decision surfaces to combat inherent preferences of choice, the system fairly chooses the nearest repulsion function to turn off. Φ_{srtrot} (4.13) is identical to Φ_{rtrot}

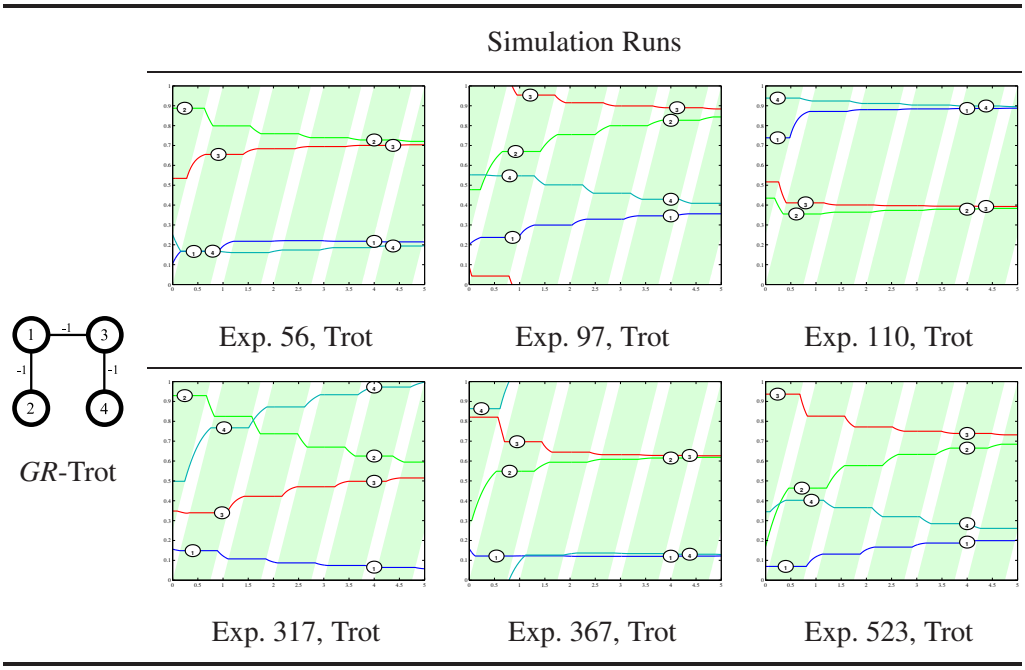
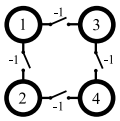


Figure 4.5: Simulation runs for Φ_{grtrot}. By including only three repulsion subfunctions, this system converges globally to a trot. In experiments 97, 317, and 523, local minima for Φ_{rtrot}, legs 2 and 4 cross, allowing the system to converge to a trot.

within its trot-convergent region, the 0-connected nodes on the related graph. When outside of this region, and within the graph nodes for Crawl #2 or #4, the system compares the distance to each artificial boundary on the potential function and selectively removes the nearest. In this way, the convergence of Φ_{srtrot} fairly partitions the local minima. Fig. 4.6 shows simulation results.



SR-Trot

$$\Phi_{srtrot}(\mathbf{x}) = c_r^T(\mathbf{x}) r_r(\mathbf{x}) \tag{4.13}$$

4.3.2 Alternative Methods

Existing methods from research literature suggest ways of stabilizing gaits such as the trot. Often, these methods specify constraints between legs using both attraction between leg pairs as well as repulsion to design gait controllers. We show that the use of attraction

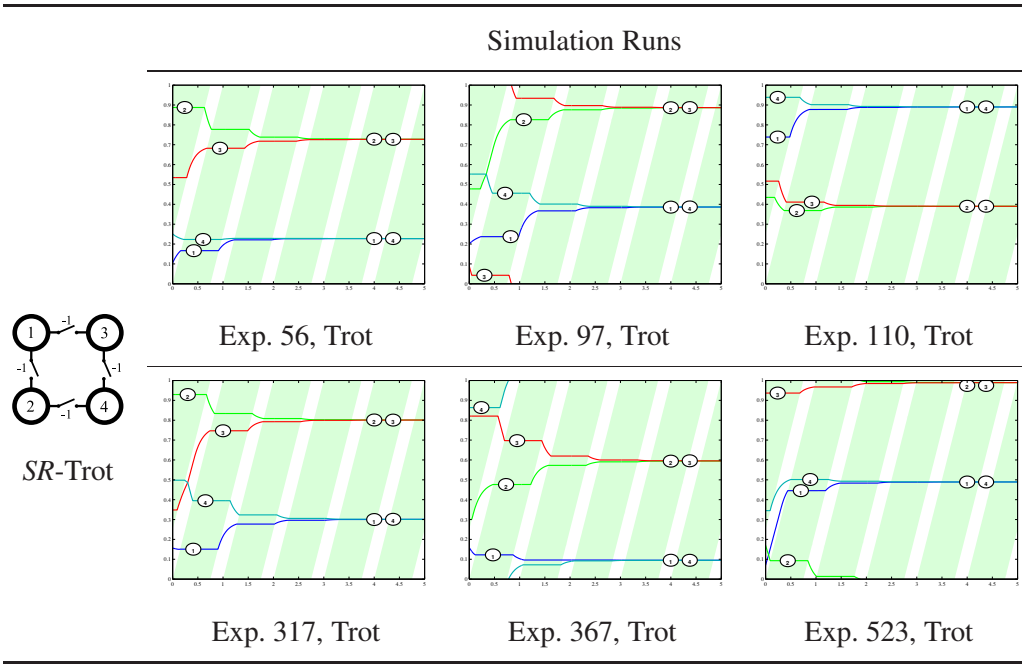
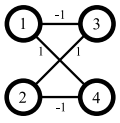


Figure 4.6: Simulation runs for Φ_{srtrot}. With the introduction of decision surfaces, this system intelligently chooses shortest convergence paths when within the regions that are local minima in Φ_{rtrot}.

between legs can be a hindrance, however, as these methods may still succumb to local minima, and, more importantly, produce odd paths of convergence that often intersect with the $\mathcal{P}$-obstacle sets, when compared to the methods of the previous section.

The first alternative method we study is Φ_{artrot} (4.16), derived from descriptions of trot leg coordination in [38]. In the description, each leg is attracted to a single other leg and repulsed from a different other leg, resulting in two sets of pairwise attraction and two sets of repulsion. To trot, leg pairs $(1,4)$ and $(2,3)$ attract one another, while leg pairs $(1,3)$ and $(2,4)$ repulse.



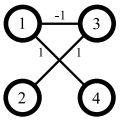
AR-Trot

$$c_a = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{4.14}$$

$$c_r = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.15}$$

$$\Phi_{artrot}(\mathbf{x}) = f_a(\rho_1,\rho_4) + f_a(\rho_2,\rho_3) + f_r(\rho_1,\rho_3) + f_r(\rho_2,\rho_4) \tag{4.16}$$

Just as Φ_{rtrot}, with its four coordination pairings between legs, produces local minima, so does Φ_{artrot}, as seen in Fig. 4.7. A slightly modified trot controller removes one of the two leg repulsions to produce global convergence, $\Phi_{gartrot}$ (4.19). This controller is derived from the description of a successful alternating tripod controller from [28].



GAR-Trot

$$c_a = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{4.17}$$

$$c_r = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.18}$$

$$\Phi_{gartrot}(\mathbf{x}) = f_a(\rho_1,\rho_4) + f_a(\rho_2,\rho_3) + f_r(\rho_1,\rho_3) \tag{4.19}$$

$$\tag{4.20}$$

Figs. 4.7-4.8 show simulation traces for both Φ_{artrot} and $\Phi_{gartrot}$. Both systems produce trot gaits, with Φ_{artrot} occasionally suffering from local minima. The fact that both systems do converge to a torus means that the boundary cuts made to the torus are sufficiently modifying the homotopy of the system (per discussion in §3.1.2). The boundary cuts made by attraction, however, place artificial ridges whenever two legs are out of phase with one another, a relatively odd constraint. For this reason, the convergence patterns seen in these systems does not look as natural as those for Φ_{srtrot}, for example.

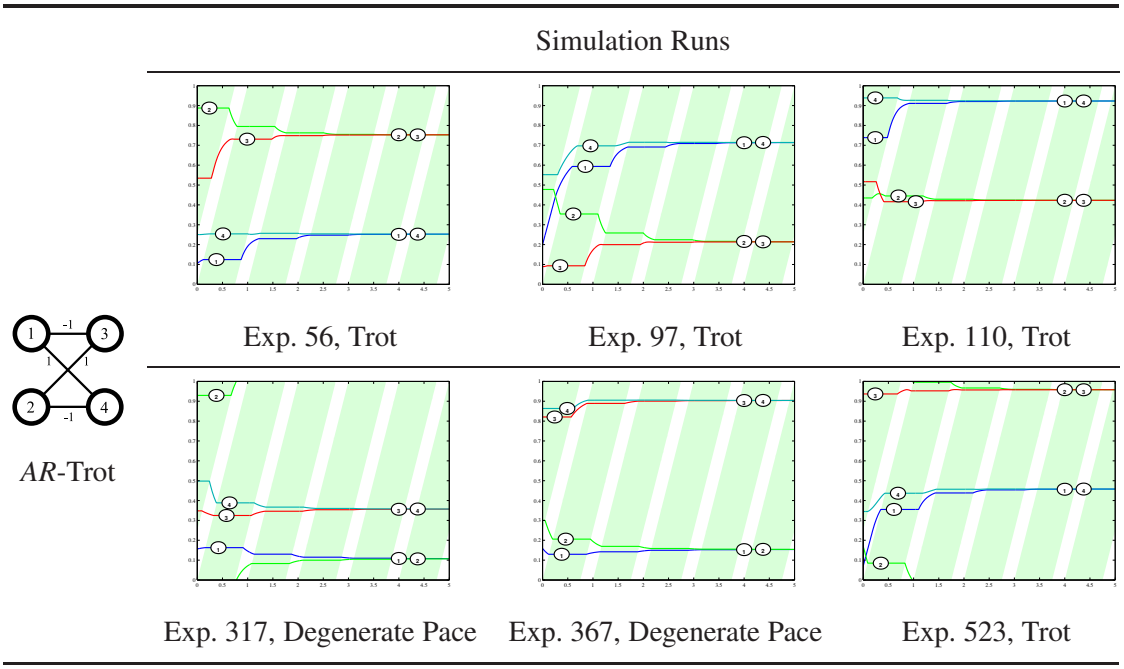


Figure 4.7: Simulation traces for Φ_{artrot}. Local minima occur in experiments 317 and 367, with the system converging to a pace rather than the desired trot.

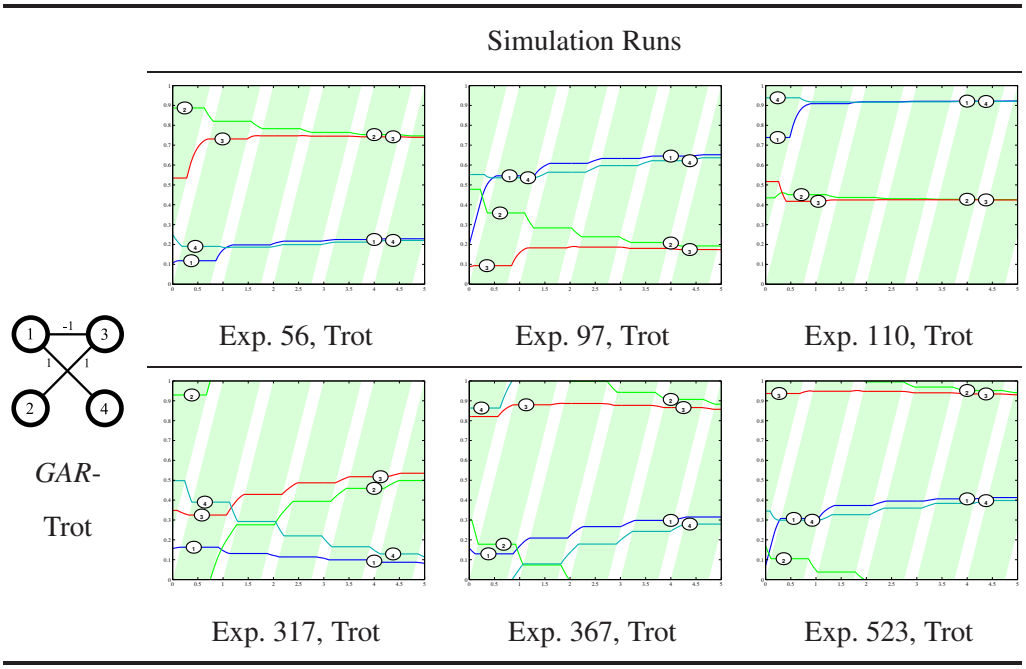


Figure 4.8: Simulation traces for $\Phi_{gartrot}$, exhibiting global convergence but allowing many legs to recirculate together, such as in 97, 317, and 367.

Another alternative method we have devised follows an opposite approach to gait regulation: rather than specify constraints to be satisfied amongst legs, the system calculates the geodesic between the current gait cycle and the desired trot gait. By following the gradient of this calculation (with exceptions made for boundary conditions, the zero measure set at which the system simply uses a convention to head one direction and not the other) the system takes the shortest-path between any two gaits. Appendix C provides background for computing geodesics between parallel cycles on the N-torus. This system, by specifying only the desired gait to achieve, does not explicitly tell us where boundaries occur on the torus, only that they occur at extremal locations, distance-wise. The performance of the geodesic method is satisfactory, and in-line with the performance of Φ_{srtrot}, Fig. 4.9 showing simulations.

4.3.3 Analysis

We know that, due to the homotopy mismatch between the torus and the cycle, each gait regulation system must necessarily introduce boundaries on the torus in order to converge. The different choices of boundary placement affects routes of convergence, and thus, the

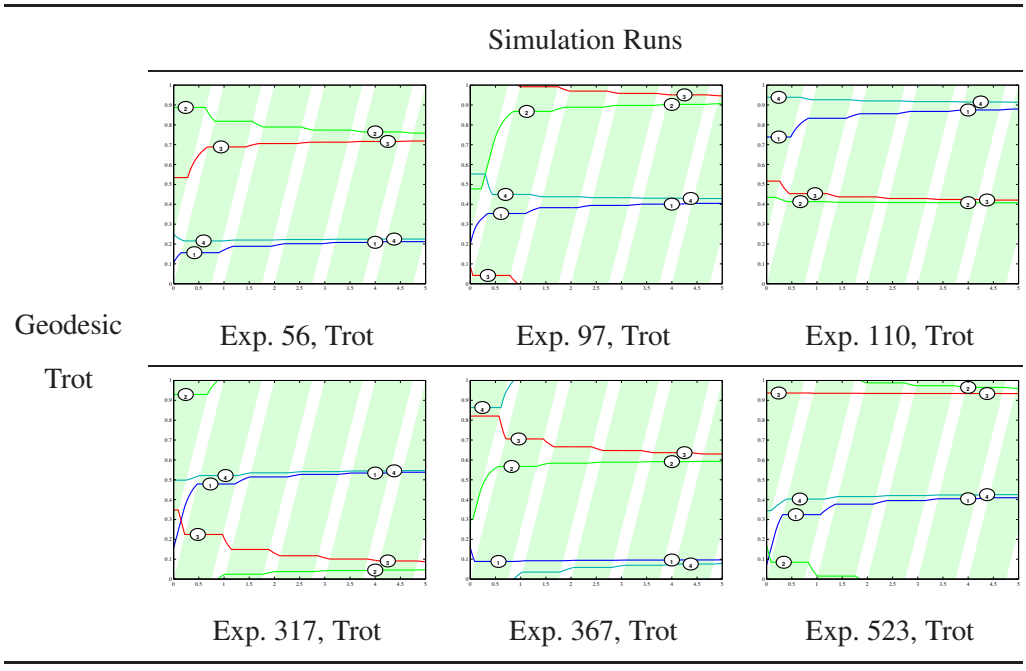


Figure 4.9: The geodesic approach to gait regulation follows the minimum-distance path between the current gait and the trot gait cycle.

ability to avoid phase space obstacles. Studying a full set of 5000 simulations for the various trot controllers, we analyze these properties.

The choice of boundary cuts, via component functions, affects the existence of limit cycle gaits. We noticed in §4.2 that a host of repulsion functions results in equal preference amongst crawl gaits. Similar conditions exist with trot gaits, as having too many cuts can often lead to local minima, as in Φ_{rtrot} and Φ_{artrot}., while slightly modified systems correctly have global convergence. Table 4.3 shows the convergence probabilities, per gait, for each trot gait regulation controller.

Systems such as Φ_{rtrot} and Φ_{artrot} converge to the trot gait roughly $\frac{2}{3}$ of the time, suffering from occasional local minima[1]. The almost identical systems to each, Φ_{grtrot} and Φ_{srtrot}, as well as $\Phi_{gartrot}$, obtain global convergence.

Looking at the convergence paths and obstacle avoidance, Table 4.4 summarizes the results for each controller's ability to avoid phase-space obstacles. As each controller makes

[1]The pace gaits produced by the AR-trot controller can be considered degenerate, since they differ from a normal pace, where $\mathbf{x} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \end{bmatrix}$.

Table 4.3: Convergence probabilities for three different trot controllers, using only repulsion between pairwise legs.

Controller	**Convergent gait**	**Convergence probability**
Φ_{rtrot}	Trot, $\begin{bmatrix} 0 & 0.5 & 0.5 & 0 \end{bmatrix}$	66.3%
	Crawl #2, $\begin{bmatrix} 0 & 0.25 & 0.75 & 0.5 \end{bmatrix}$	16.8%
	Crawl #4, $\begin{bmatrix} 0 & 0.75 & 0.25 & 0.5 \end{bmatrix}$	16.8%
Φ_{grtrot}	Trot	100%
Φ_{srtrot}	Trot	100%
Φ_{artrot}	Trot, $\begin{bmatrix} 0 & 0.5 & 0.5 & 0 \end{bmatrix}$	66.0%
	Degenerate Pace, $\begin{bmatrix} 0 & 0 & 0.25 & 0.25 \end{bmatrix}$	16.5%
	Degenerate Pace, $\begin{bmatrix} 0 & 0 & 0.75 & 0.75 \end{bmatrix}$	17.6%
$\Phi_{gartrot}$	Trot	100%
$\Phi_{geodesic}$	Trot	100%

different boundary cuts to the torus, we expect slightly different paths of convergence and thus different avoidance of obstacles as well as crossings of the diagonal subspaces of the torus (the codimension-1 where two legs are in-phase with one another, two legs crossing order). Systems that use attraction between leg pairs, Φ_{artrot} and $\Phi_{gartrot}$, perform dramatically worse at obstacle avoidance, due to the odd paths of convergence often taken. Comparing $\Phi_{gartrot}$ to $\Phi_{geodesic}$, Φ_{grtrot} or Φ_{srtrot}, all of which converge globally to a trot gait, the latter systems offer almost an order of magnitude in performance improvement.

In dealing with local minima, the use of decision surfaces of Φ_{srtrot} both avoids phase obstacles best, while also producing global convergence, due to our hybrid control approach. While Φ_{rtrot} suffers from local minima, the controller Φ_{grtrot} manages global convergence by setting a preferred convergence path. This path is not necessarily the shortest route, however, thus the system incurs a greater percentages of time spent recirculating three legs together, or the wrong set of two legs, as shown in the Table 4.4. Comparing the crossing of cuts shown in the table, we seen that the crossing of the diagonal subspaces of the torus depends directly upon the activation of repulsion between leg pairs. Φ_{srtrot}, when noticing it is within the local minima regions of Φ_{rtrot}, evenly partitions the local minima using decision surfaces, and fairly crosses any one of four diagonals, unlike Φ_{grtrot}.

The probabilities of cut crossings provide evidence for the ways that our gait regulation controllers cut the torus. Of note, whenever a repulsion function is enabled for a particular gait regulation controller, the corresponding diagonal cut is never crossed (compare c_r in (4.9) to the first row of Table 4.4.)

Comparing Φ_{srtrot} with $\Phi_{geodesic}$ is more difficult, as both have similar properties of convergence. $\Phi_{geodesic}$ seems to cross the diagonal cuts a greater percentage of time,

Table 4.4: Cut crossings and obstacle set analysis

Controller	$\rho_1 = \rho_2$	$\rho_1 = \rho_3$	$\rho_1 = \rho_4$	$\rho_2 = \rho_3$	$\rho_2 = \rho_4$	$\rho_3 = \rho_4$	2†-leg obstacle	3-leg obstacle
Φ_{rtrot}	0.0%	0.0%	1.40%	1.56%	0.0%	0.0%	2.03%	0.12%
Φ_{grtrot}	0.0%	0.0%	17.8%	18.8%	33.7%	0.0%	3.02%	0.15%
Φ_{srtrot}	8.84%	8.32%	1.52%	1.64%	8.42%	8.10%	1.19%	0.12%
Φ_{artrot}	10.9%	0.0%	16.2%	17.3%	0.0%	10.6%	15.6%	0.66%
$\Phi_{gartrot}$	24.6%	0.0%	35.3%	36.8%	34.2%	25.6%	3.87%	1.67%
$\Phi_{geodesic}$	10.8%	10.4%	1.38%	1.12%	10.0%	10.3%	0.81%	0.15%

suggesting unnecessary crossings, however the systems perform so closely we cannot distinguish well between them.

In the goal of producing a trot gait regulation controller, we argue that Φ_{srtrot} is the best available approach among the controllers presented. Its use of only repulsion between leg pairs produces convergence that best avoids obstacle regions. The inclusion of decision surfaces results in non-preferential and even convergence patterns, providing global convergence as required. To compare further between the switched hybrid approach and the geodesic approach, we move to simulations of a higher-dimensional system.

4.4 Realizing the Alternating Tripod Gait

Similar to a trot gait, we also show the application of control methods for an alternating tripod gait. Our switched control methods, now operating on a cellular decomposition including 120 distinct cells, must critically plan paths of convergence while safely avoiding obstacle regions. When compared to a handful of alternative techniques, the hybrid controller's approach best approximates the obstacle regions while also achieving global convergence.

4.4.1 Hybrid Switching Control

The hybrid controller's approach to the alternating tripod gait guarantees global convergence, while following a minimum number of leg crossings before reaching the final limit cycle. We describe the internal decision making of the hybrid control algorithm through simulation examples.

Fig. 4.10 shows examples of an alternating tripod gait converging to the gait limit cycle. At each timestep, the system determines the crawl gait cell it is within, based upon leg orderings, determining the number of leg crossings that must take place in order for convergence to occur. By comparing distances to various crossings, the simulation converges to the alternating tripod gait. Similar to the decision surfaces introduced in the previous section, the decisions here guarantee a fair convergence pattern from distant limit cycles on the torus.

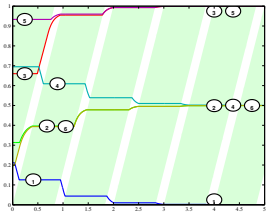
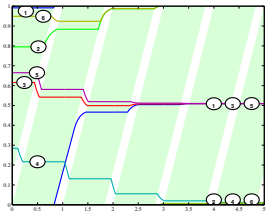
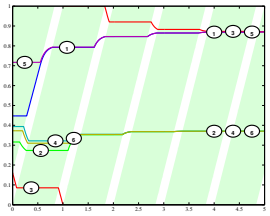
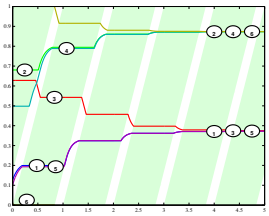
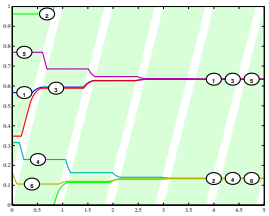
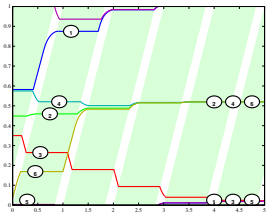
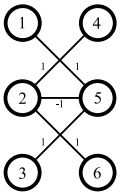
	Simulation Runs		
SR-Tripod			
	Exp. 39, Tripod	Exp. 96, Tripod	Exp. 326, Tripod
			
	Exp. 554, Tripod	Exp. 951, Tripod	Exp. 1036, Tripod

Figure 4.10: Simulation runs for $\Phi_{srtripod}$. In each simulation, the system crosses the minimal number of legs in order to converge.

4.4.2 Alternative Methods

We propose two globally convergent controllers that achieve gait regulation of the tripod gait, for the purpose of comparing with our hybrid control approach, akin to the alternative trot methods discussed previously, but also show failures of these methods to encode the same awareness of obstacle sets as $\Phi_{srtripod}$.

Similar to the trot controller, we show a gait regulation controller using both attraction and repulsion to globally converge on a tripod. This controller uses repulsion between two legs of opposite tripods, and attraction amongst legs within the tripod (4.21). Fig. 4.11 shows simulations of this system, again seeing "odd" patterns of convergence as the system takes differing routes to the convergent gait, compared to $\Phi_{srtripod}$.



GAR-

Tripod

$$\Phi_{gartripod}(\mathbf{x}) = f_r(\rho_2,\rho_5) + f_a(\rho_1,\rho_5) + f_a(\rho_3,\rho_5) + f_a(\rho_2,\rho_4) + f_a(\rho_2,\rho_6) \quad (4.21)$$

(4.22)

Like performed for the trot, we also introduce a geodesic gait regulation controller for the alternating tripod. This system follows the shortest path on the torus between the initial gait to the tripod limit cycle. Fig. 4.12 shows simulations of this system.

4.4.3 Analysis

We compare convergence properties of the three globally convergent alternating tripod gait regulation controllers. In our analysis, it is apparent that $\Phi_{srtripod}$ best captures the constraints of $\mathcal{P}$-obstacles, while always converging to the desired gait. Furthermore, by constructing the controller based upon minimizing the number of leg swaps that must occur, the system does not perform unnecessary swaps of legs.

Table 4.5 shows the global convergence properties of all three tripod gait regulation methods. Of note, we show that $\Phi_{srtripod}$, with its hybrid control algorithm, produces global convergence. While the number of component repulsion functions is combinatorially

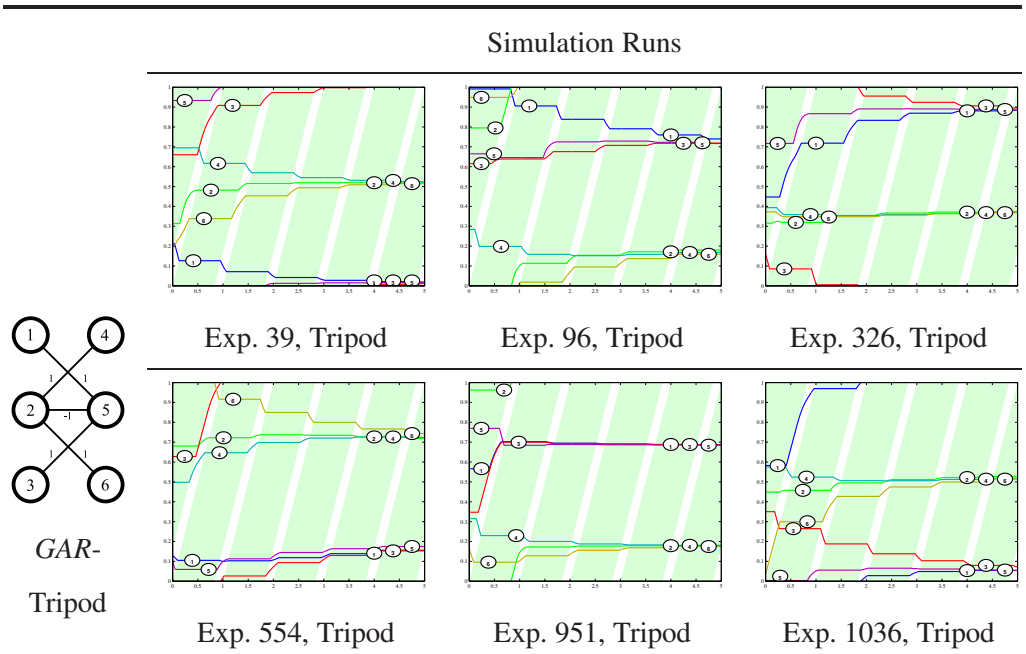


Figure 4.11: Simulation runs for $\Phi_{gartripod}$. In experiments 96, 554, and 1036, the system takes odd routes of convergence, crossing unnecessary legs, compared to $\Phi_{srtripod}$

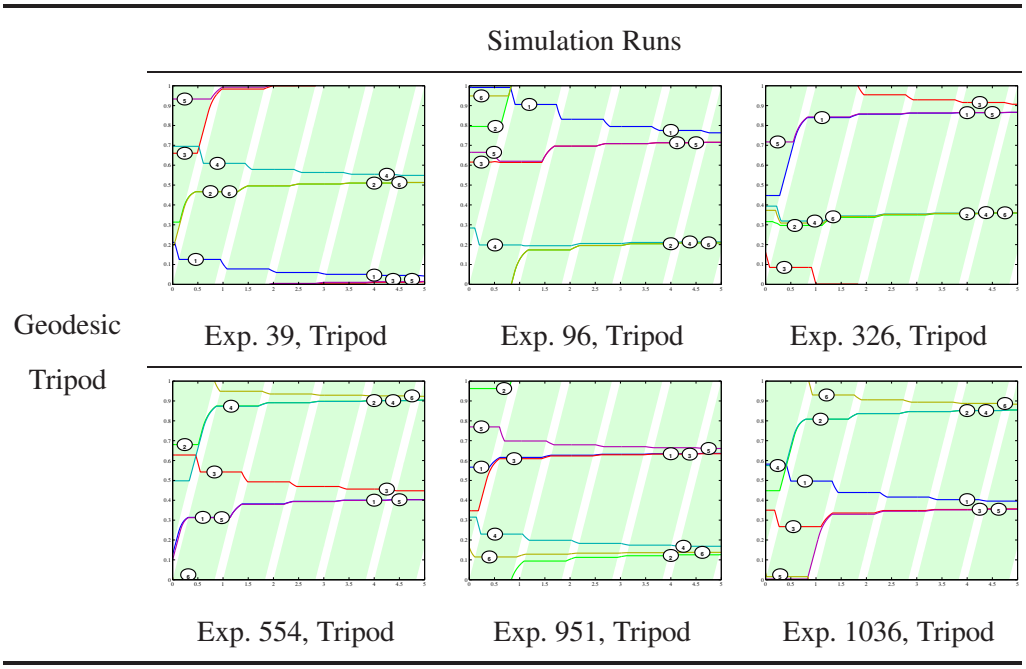


Figure 4.12: Simulation runs for $\Phi_{geodesic}$. While similar to the results for $\Phi_{srtripod}$, the system sometimes results in slightly different routes of convergence, such as in Experiments 96 and 1036.

Table 4.5: Convergence probabilities for the tripod gait regulation controllers.

Controller	**Convergent gait**	**Convergence probability**
$\Phi_{srtripod}$	Tripod, $\begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}$	100%
$\Phi_{gartripod}$	Tripod	100%
$\Phi_{geodesic}$	Tripod	100%

greater than Φ_{srtrot}, the system still intelligently produces global convergence by fairly partitioning potential local minima.

Table 4.6 shows obstacle avoidance for the three systems, and again we note an order of magnitude performance improvement when *not* using attraction between leg pairs. $\Phi_{srtripod}$ again performs marginally better than the geodesic approach at avoiding 3-legged obstacles. The marginal differences between the switched controller and the geodesic may be accounted for in simple controlling tuning, however.

We further note peculiarities in the convergence of the three systems by studying the number of times each controller allows leg pairs to cross, shown in Table 4.7. The hybrid control algorithm, by structuring its convergence to minimize the number of cut crossings, performs, on average, the minimum. Note that the crossing probabilities matches with the overall connectedness of the phase space graph for the tripod gait, where there are (36,72,12) (0,1,2)-connected cells. The other two tripod controllers perform greater numbers of crossings, suggesting greater incursion into the phase space obstacle regions.

Table 4.6: Obstacle values for tripod controllers

Controller	3†-legs	4-legs
$\Phi_{srtripod}$	0.17%	0.017%
$\Phi_{gartripod}$	1.97%	0.75%
$\Phi_{geodesic}$	0.19%	0.016%

Table 4.7: The number of diagonal crossings per controller over 5000 simulations. $\Phi_{srtripod}$, with its lower valued distribution, performs less cut crossings on average.

Controller	**Number of diagonal crossings**							**Total**
	0	1	2	3	4	5	6+	
$\Phi_{srtripod}$	29.02%	60.24%	10.72%					100%
$\Phi_{gartripod}$	21.5%	19.4%	21.6%	14.4%	9.70%	6.74%	6.60%	100%
$\Phi_{geodesic}$	28.3%	36.8%	26.4%	6.38%	1.92%	0.24%	0.02%	100%

4.5 Tetrapod Gaits for a Hexapedal Robot

The ability to design multiple basins of attraction, using our hybrid gait regulation control techniques, as is described in §3.3.4, allows us to pair multiple gait regulation controllers to form a higher-level of hybrid control, allowing the system to converge to certain gaits based upon distance comparisons between controller arrangements. We apply this technique to the tetrapod gaits, discussed previously, in order to realize an even distribution of convergence to four different tetrapod gaits, created using two different controller arrangements.

The use of our hybrid control approach is advantageous due to the readily available metrics of gait distance. By comparing both discrete distances on the crawl gait graph, as well as cut distances, the tetrapod controller chooses the "nearest" controller arrangement to activate.

The hybrid controller first chooses which controller arrangement to be used based upon graph traversal distance, with equal traversals further compared by average distance to cuts. By then activating one tetrapod gait regulation controller, and performing the graph search to deactivate the necessary cuts for global convergence, the system naturally achieves both the "forward" and "backward" versions of the tetrapod gait.

We simulate this controller over a large variety of initial random conditions, with examples in Fig. 4.13. As noted in Table 4.8, the convergence probabilities are evenly distributed over all four possible tetrapod gaits.

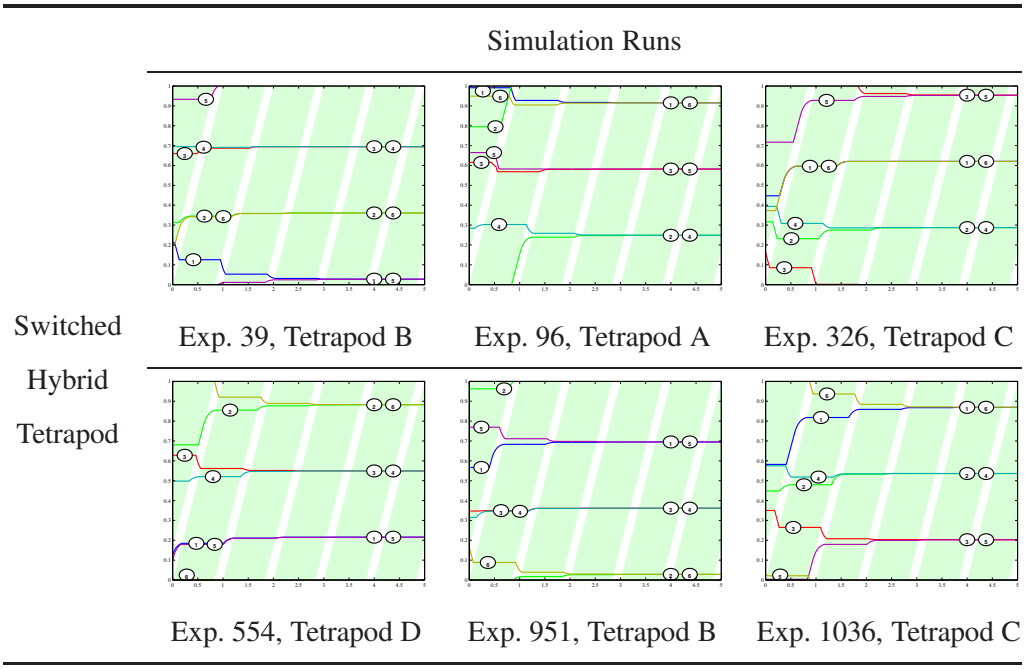


Figure 4.13: Simulation runs for $\Phi_{srtetrapod}$

Table 4.8: Convergence probabilities for the hybrid tetrapod gait regulation controller. Note the even probabilities across all four tetrapod gaits.

Controller	**Convergent gait**	**Convergence probability**
$\Phi_{srtetrapod}$	Tetrapod A, $\begin{bmatrix}0 & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & 0\end{bmatrix}$	25.6%
	Tetrapod B, $\begin{bmatrix}0 & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & 0 & \frac{1}{3}\end{bmatrix}$	24.8%
	Tetrapod C, $\begin{bmatrix}0 & \frac{2}{3} & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & 0\end{bmatrix}$	25.1%
	Tetrapod D, $\begin{bmatrix}0 & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3}\end{bmatrix}$	24.5%
	Total	100%

4.6 Concluding Remarks

By simulating a variety of gait regulation controllers, including both previous approaches for leg coordination as well as our hybrid switched control approach, we show our methods to perform dramatically better than previous approaches, offering both desired properties of global convergence as well as avoidance of dangerous gait timings.

The use of only pairwise leg repulsion, compared to the use of leg attraction by prior methods, results in an order of magnitude improvement in the avoidance of dangerous gait timings. Furthermore, our hybrid switched algorithm guarantees that a system allows only the minimal number of legs to swap order during convergence to a gait. Lastly, the ability to design the deployment of convergent basins allows us to apply gait regulation for situations in which multiple preferred gaits exist.

Chapter 5

Gait-Based Design of a Climbing Robot Behavior

Following our approach for legged locomotion control, utilizing gaits to deploy behaviors as discussed in Chapter 2, we design a complete feedback behavior for a hexapedal climbing robot, the RiSE robotic platform, and discuss the application of our gait regulation techniques to the climbing task. As we introduce gait-based control laws that disturb an initial gait's timing properties, gait regulation is a critical component to climbing behaviors, thus keeping the robot in proximity to preferred gaits.

5.1 The RiSE Robotic Platform

This section describes the experimental platform used to evaluate methods of gait-based control and gait regulation. While the author is principally responsible for the behaviors and gait control software used with the current robot, as well as maintenance of other portions of the software infrastructure, this section familiarizes the reader with overall efforts of the RiSE team to create a general purpose climbing machine.

The RiSE robotic platform (Fig. 5.1) is a biologically inspired hexapedal robot designed for vertical climbing as well as horizontal mobility [4, 42]. Unique constraints of this robot make it perfectly suited to the gait-based control approach for designing feedback behaviors.

The robot contains a total of twelve Maxon RE16 4.5 W motors[1], actuating two degrees of freedom per leg. The total weight of the robot is approximately 3.8 kg, with a length of 68 cm, including a 28 cm rigid tail appendage in addition to the six legs. Each pair of motors uses a belt-drive system and differential mechanism to actuate two joints: the *crank*

[1] Maxon Motor AG, Sachsein (CH), http://www.maxonmotor.com

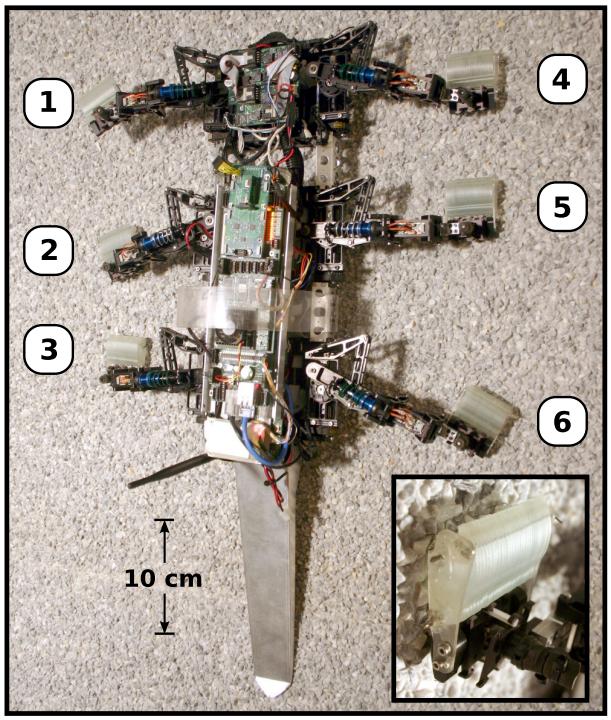


Figure 5.1: RiSE is shown climbing a crushed stone wall using its compliant feet with embedded microspine structures. Leg numbering conventions are noted. *Inset*: the upper left foot (#1) attached to the surface of the wall.

joint moves the foot along a planar path using a four-bar mechanism, while the *wing* joint allows adductor/abductor motions, moving a foot closer to or further from the climbing surface. By centralizing the motors within the body of the robot, leg mass is minimized.

With only twelve active degrees of freedom for a six-legged robot, RiSE is severely underactuated and must rely upon tuned passive mechanics to locomote effectively [4, 40, 42], based upon the biological concept of preflexes [30]. The stucco climbing behavior that we describe here makes use of engineered robot feet, manufactured using Shape Deposition Manufacturing [33, 48], containing dozens of individual microspines per foot. Each microspine is embedded into a multi-material structure with tuned compliances [3]. The following section describes how RiSE employs open-loop gaits to locomote using these specific feet, utilizing specifically designed motor trajectories tuned to engage microspines on surfaces asperities to gain traction for climbing.

With on-board power, electronics, and control software, the robot is both computationally and power autonomous, capable of extended runtimes in excess of 45 minutes, while using a total of up to three 90 mAH lithium-polymer battery packs. Motor controller boards, sensor nodes, and the main CPU board, housing a Geode 266 MHz processor, communicate over a modified I2C bus [4]. RHexLib control software—originally designed for the RHex hexapedal robot [41] and adapted for use with RiSE—contains a static scheduler that allows the robot to query sensor measurements and issue control commands at a fixed rate of 300 Hz, and is run under the QNX real-time operating system[2]. A human operator controls the robot via a joystick connected to a laptop computer. While control software runs in real-time on the robot's CPU, the operator may issue occasional commands to affect control parameters. The laptop communicates with the robot over a standard 802.11b wireless link using commercial off-the-shelf (COTS) components. This allows for 2-way data communication, issuing the robot control commands while also relaying robot health data back to the operator, and allowing for logging of real-time robot data for analysis.

In addition to the use of optical encoders on each of RiSE's motors as well as an on-board inertial measurement unit, the robot's legs contain elements for sensing forces, measuring 3 perpendicular axes of force per leg. Strain gauge load cells built into distal leg components achieve force sensing for the fore-aft and adduction/abduction directions, measuring forces normal and tangential to the feet. The third axis of force sensing is provided

[2]QNX real-time operating system, http://www.qnx.com

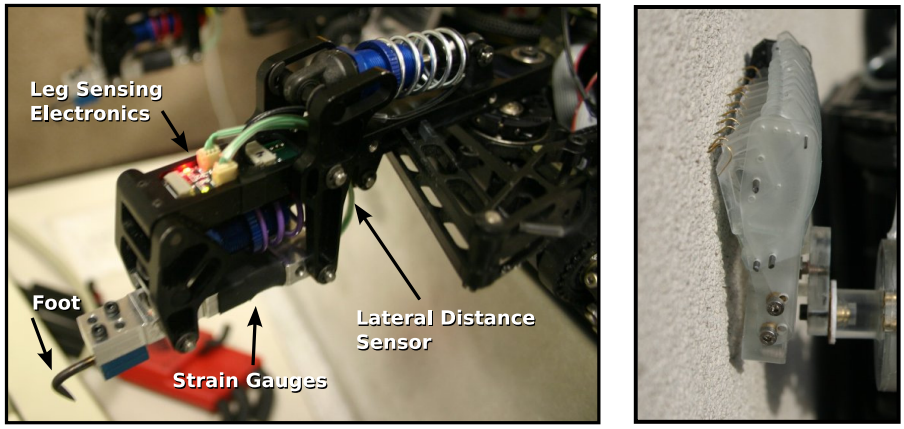


Figure 5.2: *Left:* Annotated picture of a RiSE lower leg. Various types of feet can be installed at the distal end of the leg. Two strain gauges measure the normal and fore-aft forces on a foot. With a compliant passive degree-of-freedom in the lateral direction, a distance sensor, paired with a spring constant, gives us lateral force. *Right:* A RiSE foot with individually-compliant microspines on stucco.

by measuring the lateral extension of a passive leg joint using a Hall effect sensor [42]. The strain sensors are calibrated with known masses, and exhibit a linear relationship between raw measurements and force, in Newtons. The signal of the Hall effect sensor, with a magnet moving relative to the sensor when the lateral leg extension changes, follows a sine wave, thus we fit an arcsine curve to the raw sensor values to determine a calibrated distance measurement of leg extension. That distance is converted to force using the spring constants associated with the miniature spring-damper units installed on RiSE's legs [42, 45].

5.2 Design of an Open-Loop Climbing Gait

A climbing robot faces unique constraints, as the power density requirements to design and build a fully actuated climbing robot are believed to be beyond limitations currently placed by commercial motor technologies [27]. By using a robot that has few actuated links and a large set of compliant elements, as RiSE is designed, we achieve climbing behaviors for various surfaces. This section describes the design of an open-loop gait used for one such behavior, climbing a vertical stucco surface.

Following the decomposition of open-loop gaits into both temporal and spatial components (§2.2), we design and tune gait parameters in two separate ways, following the same spatio-temporal split. Tuning the geometry of a foot's motion to the specific surface in question, in our case a stucco surface, allows us to pair the leg motion with differing gait timings, as we do in our experimental results.

We first consider the motion a microspine foot (Fig. 5.2) must take to engage with a stucco surface. Fig. 5.3 shows the workspace resulting from one of RiSE's legs, considering the possible motions of the crank and wing joints. Any leg motion, in joint coordinates, will create a curve on this manifold (plus additional tolerance due to compliance). Design of a gait relies upon designing the geometry of the robot's leg joint trajectory to engage compliances while moving on this manifold.

The gait is designed to recruit the crank joint to engage microspines and produce traction forces while climbing. Traction force is aligned in the fore-aft direction and is along the path a foot takes when the four-bar mechanism is actuated via the crank degree of freedom. After making contact with the surface, the crank joint turns faster to engage the spines, then slower during stance to propel the robot up the wall. At the end of the stroke, the crank

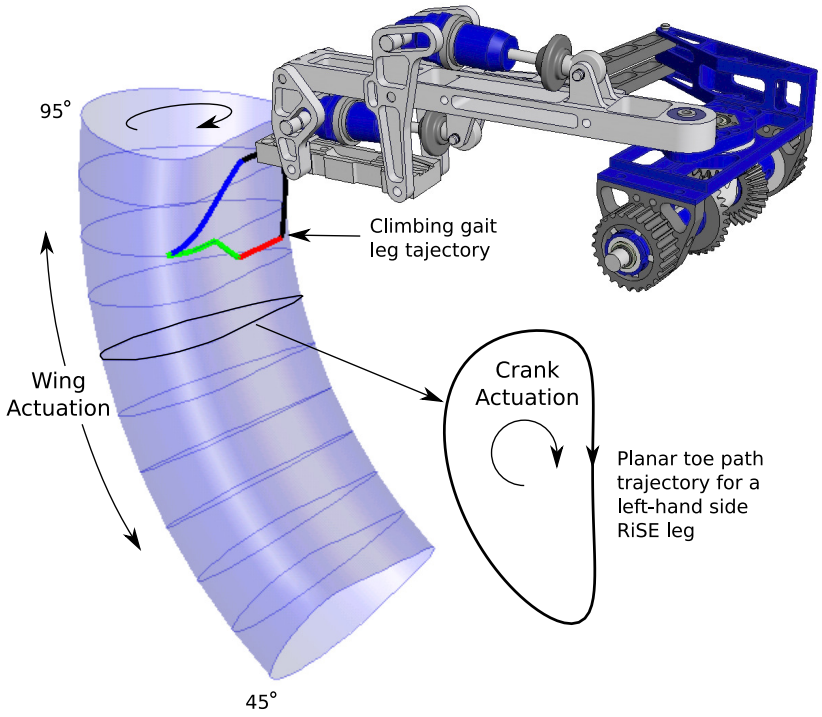
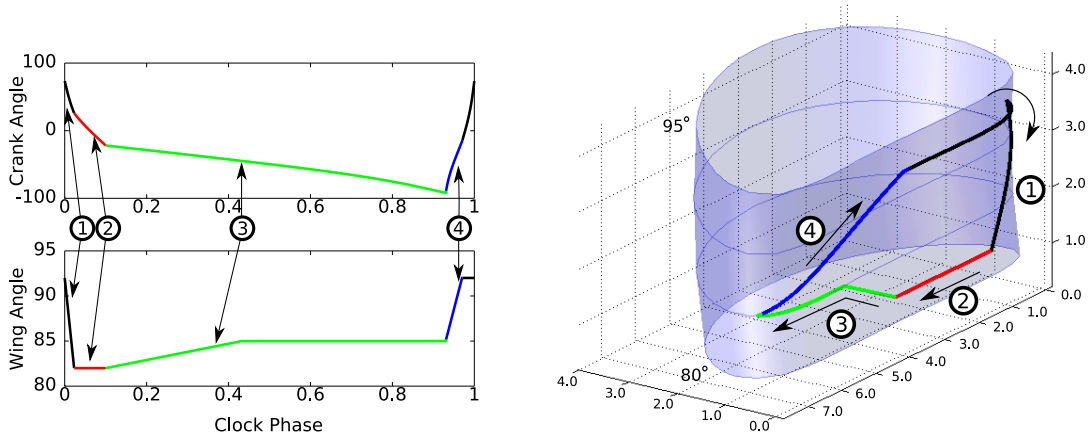


Figure 5.3: The two leg joints combine to parameterize a 2-dimensional manifold of allowable foot positions. The leg joint trajectory corresponds to a loop on this manifold. A physical robot leg is shown for comparison.

reverses to unload the leg compliance and disengage the spines before the leg recirculates. This sequence of steps is seen in the plots of Fig. 5.4.

The normal force of a foot plays a crucial role in climbing and is largely tuned by adjusting the gait parameters determining the wing joint trajectory. The foot first produces a positive force when it strikes the surface. Pull-in force (negative normal force) is necessary to keep the robot's body close to the wall, and the wing joint is used to perform this task, increasing after the feet are attached.



Joint coordinates

Workspace coordinates (units in cm)

Figure 5.4: A single foot's trajectory, shown in joint and body frame coordinates. In body frame coordinates, a bounding box of the trajectory is shown, with units in centimeters. The sequence of events are as follows: **(1)** the robot lowers the wing joint to present a foot to the surface; **(2)** the attachment stroke drags a foot along the surface to engage it; **(3)** the foot enters stance and generates pull-in and fore-aft force; and **(4)**, the crank direction reverses while the foot is lifted away from the surface, both unloading compliance and beginning recirculation.

RiSE often makes use of the lateral compliance found in its legs in other scansorial behaviors. The climbing behavior described here, however, uses only the normal and fore-aft compliances found in the microspine toes. Additional lateral compliance would add more robustness to foot attachment, but is currently limited by the relative strength of the toes and by the yawing rotation required by a foot during the stance phase, due to the robot's underactuation.

For temporal gait tuning, the robot's overall large mass and the limited strength of the microspines currently limits it to slow stable gaits when climbing vertical surfaces. For this reason, the open-loop gait was initially tuned for use with a pentapedal crawl gait, phase offsets separated by $\frac{1}{6}$ and duty factor of approximately $\frac{5}{6}$. This gait, as is discussed in §2.3, attempts to maintain surface contact with at least five legs at all times.

The integration of these two components—the temporal portion, which dictates when each foot begins stance and later recirculates, and the foot path trajectories encoded within the spatial portion—results in a whole body motion that can be used to climb surfaces without sensor feedback. A proportion-derivative (PD) control loop, shown in Fig. 2.1, is used to follow desired motions, and gait parameters are tuned manually to achieve effective climbing.

5.3 Layered Approach for Feedback Control

Following our techniques for applying feedback control to open-loop gaits (§2.4), we devise a complete feedback behavior that layers various controllers atop the gait described above, in order to add robustness for climbing.

The various components of the climbing behavior are shown in Fig. 5.5. The most basic layer of behavioral feedback comes from reactive force controllers, adjusting the gait based upon ground reaction forces during locomotion. Second, an algebraic relationship controls robot speed and the duty factor of the behavior, based upon basic gait timing. Finally, the highest level of the behavior, gait regulation, supplies feedback to keep the robot using certain gait timings during locomotion.

5.3.1 Reactive Task-Level Controllers

Our approach for force control is to design a behavior using an open-loop gait, taking advantage of the built-in leg and foot compliances, and adjust the gait based upon errors in expected ground reactions. With a reactive model for control, these force controllers operate at the lowest-level of our feedback behavior.

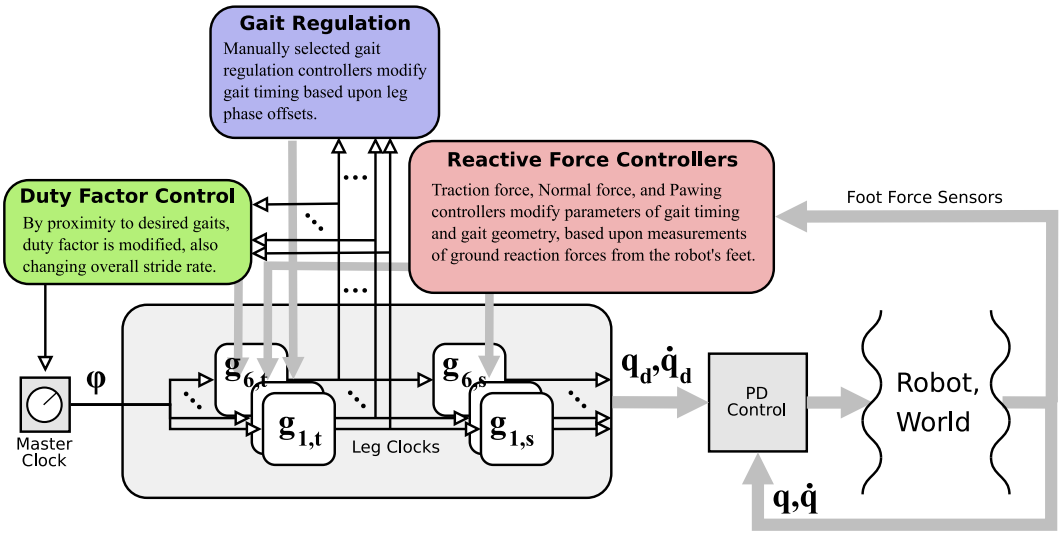


Figure 5.5: The layering of feedback control atop the open-loop gait (from Fig. 2.1). Force control techniques, in the form of traction force, normal force, and leg pawing control, adjust gait to perform task. Duty factor control adjusts overall stability and robot speed, while gait regulation corrects for errors in desired gait timing.

Traction Force Control

As the open-loop gait, described above, is designed to engage the microspine toes built into RiSE's feet, as well as to load traction force (force in the fore-aft direction) throughout the toe compliances and across multiple feet of the robot, we discuss control laws that attempt to guarantee this occurrence.

As traction force carries the weight of a robot while climbing vertically, we apply control to regulate the force experienced per foot. Traction of large magnitude may cause individual microspines to disengage, over-extend, or be damaged permanently. Conversely, a foot carrying too little force may not be well attached to the wall and contributes little to the distribution of force throughout the body [23, 45].

In the ideal case, traction force should be balanced from side to side of the body, resulting in zero net torque. On each side, given identical foot structures attached to each foot, the legs should experience similar loads. If $f_{t,i}$ is the traction force for leg i, we compute the desired values for legs on the left and right sides of the robot as:

$$f_l^*(t) = \frac{\sum_{i=1}^{6} f_{t,i}(t)}{2\sum_{i=1}^{3} s_i(t)} \tag{5.1}$$

$$f_r^*(t) = \frac{\sum_{i=1}^{6} f_{t,i}(t)}{2\sum_{i=4}^{6} s_i(t)} \tag{5.2}$$

The motion of the climbing gait's stance stroke is aligned with traction force, thus we apply control by differentially adjusting the phase of an individual leg's stance, modifying the timing of the gait using a simple gait-based proportional controller that compares desired force to measured traction force, shown in (5.4). This controller, with incorporation of a simple error deadband, is applied directly to phase offsets for legs in stance, with an increase in phase offset decreasing an individual foot's traction force. Fig. 5.6 shows an example of using a discrete version of this controller to regulate the traction force of a single leg.

$$f_{f,i}^* = \begin{cases} f_l^* & \text{if } i \in 1,2,3 \\ f_r^* & \text{if } i \in 4,5,6 \end{cases} \tag{5.3}$$

$$u_i = k_p(f_{t,i} - f_f^*) \tag{5.4}$$

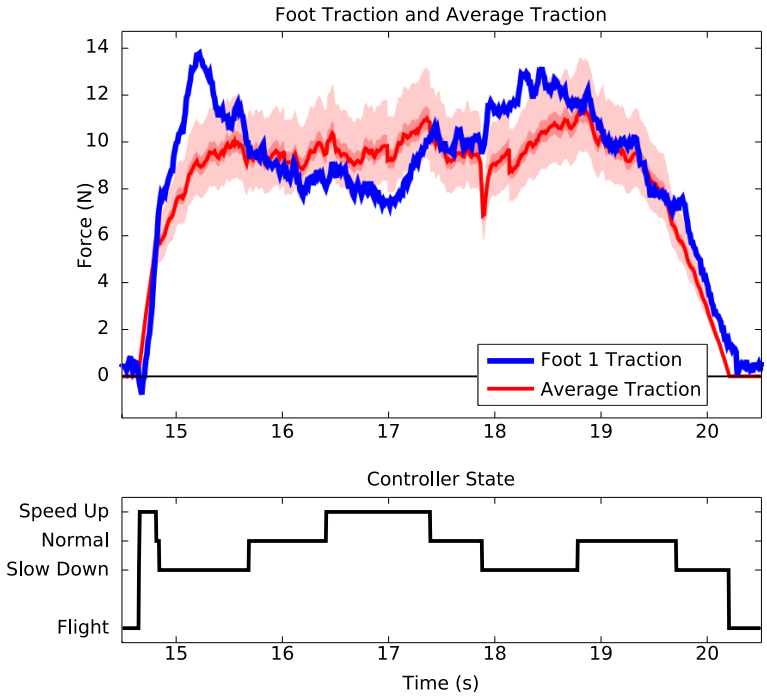


Figure 5.6: Result of applying traction force controller to a single leg. The top plot shows the actual force measured at a foot, as well as the desired force over the stride. The shaded region notes the deadband region. The lower plot shows the controller action.

RiSE occasionally yaws to one side due to small slips per side during climbing. If uncorrected, the robot will continue to turn toward that side. To alleviate this problem, as well as have higher level control of the climbing, a simple strategy for turning was implemented.

A previous method for turning is discussed in [24], where the robot alternated between its normal gait and specific "turning" gaits. This approach was difficult to tailor to the adaptive gait systems described here because it assumed that the robot was using fixed gaits. A better strategy is to make use of the traction force controller. The traction force controller equalizes the foot force by comparing the individual foot forces to the average foot force. By modifying that average value for either the set of right or left legs in proportion to the heading error, the robot will generate higher forces on one side of the robot compared to the other side. This naturally turns the robot since the legs on one side of the robot are moving faster than the other side and the imbalance of forces introduces a small torque.

Normal Force Control

In contrast to the control of traction force via gait timing, a normal force controller that adjusts the wing limits of the geometric component of the gait function, $g_{i,s}$, is used to guarantee that feet make contact with the wall before attempting to load the traction force.

Sometimes problems can occur when a foot is unexpectedly far from the surface, often due to the robot pitching back or due to surface irregularities. If the foot fails to make contact, the critical chain of steps that are designed into the open-loop gait—generate normal force, load traction force, and generate adhesion—is broken. To address this challenge, the wing angle is lowered until the foot "feels" the surface (1 N of force is measured in the normal direction). This is done by adding an extra *wing angle offset* to the position commanded by the open-loop gait. After 1 N of normal force is registered, the leg returns back to the nominal wing angle causing the robot to be pulled into the surface. This step helps to correct for pitch errors and generate adhesion force. Fig. 5.7 shows the allowable range of wing angle modification, overlaid onto a section of the wing angle plot from Fig. 5.4.

Pawing

Empirically, we have determined one additional control strategy to be critical for our climbing behavior, a "pawing" reflex [18], by which a leg reattempts attachment after detecting detachment of a stance foot. Unlike the force balancing controller described above, which

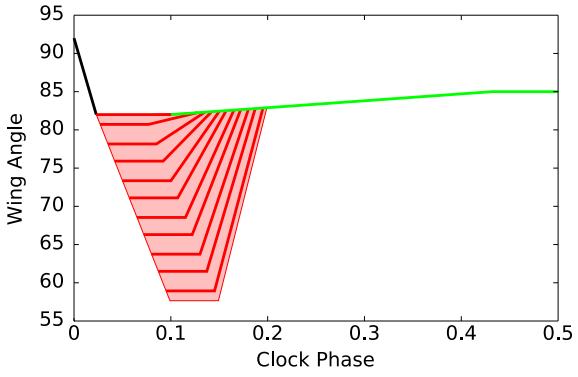


Figure 5.7: Allowable modification to wing angle during normal force control. Leg is lowered past nominal gait trajectory until surface is detected, at which time traction force is loaded, then wing angle reversed to generate adhesion force.

applies differential adjustments to the robot's gait timing, pawing adds an additional challenge because it applies discrete changes, suddenly dictating that a foot should begin recirculation rather than continue through stance.

Attachment failures often occur when a foot slips while trying to attach and can be associated with bare spots where attachment is difficult. Pawing attempts to reattach the foot both quickly and in a slightly different location on the climbing surface by quickly recirculating the foot. Unlike the two strategies above that make differential adjustments to the gait, pawing discretely changes gait parameters.

Pawing can occur throughout the stance section of the gait. Thus, when the leg is recirculating, the leg clock needs to be reset. This new offset to the leg clock is calculated by comparing the current phase to the phase at which detachment occurs. Position offsets are added to joint angles to maintain continuity of commanded positions and are computed as follows: If $g_i(\phi_d)$ are the normal joint positions of a leg at the beginning of detachment, and $g_i(\phi_p)$ are the joint positions at the beginning of a pawing behavior, then $g_i(\phi_p) - g_i(\phi_d)$ are position offsets that allow the robot to execute the detachment stroke starting at the pawing position. When detachment occurs, the position offset is reduced to zero while the leg recirculates. When the leg attaches, the leg returns to the nominal trajectory. An example of a pawing motion is shown in Fig. 5.8.

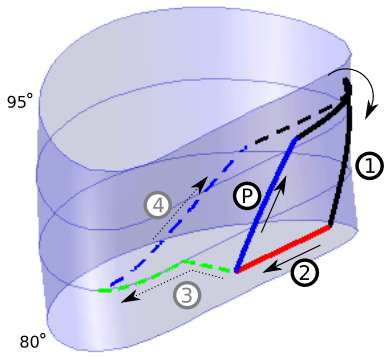


Figure 5.8: A leg may exhibit a pawing behavior after recirculation and attempted attachment, steps **(1)** and **(2)**. Unlike the trajectory shown in Fig. 5.4, a pawing leg will, upon sensing failed attachment, skip steps **(3)** and **(4)** by lifting up earlier, **(P)**, in order to recirculate and retry attachment.

5.3.2 Active Duty Factor and Speed Control

The second layer of our feedback behavior pertains to active control of the current gait's duty factor, the percent of each stride dedicated to stance. Certain gaits allow a wider range of duty factors than others, as, for instance, the trot and tripod gaits can use duty factors as low as $\delta = 0.5$ to locomote faster while remaining statically stable. Thus far, we have discussed the implications of gait regulation control upon changing leg phase offsets, $\mathbf{x}$, in order to avoid dangerous regions of the control space, $\mathcal{P}$-obstacles. Modification of duty factor increases or decreases the geometric size of this obstacle set, easily visualized as the shaded regions of Fig. 3.1 changing size, thus limiting the valid gaits that perfectly avoid the obstacle. Lower duty factor corresponds to faster locomotion, thus it is of great interest to develop methods that modify duty factor when locally close to such gaits. In this section, we develop a simple control law that uses an algebraic relationship with phase offsets to implement duty factor control.

A trade-off between speed and stability exists for legged gaits. The gaits, such as the crawl, that keep a maximum number of legs in stance—points of contact with the ground—have stability but must locomote slowly, as the robot's stride rate is limited by the motor speeds at which individual legs may be recirculated. A gait with a lower duty factor, less stable with fewer legs in contact with the ground, covers the same stance stroke in less time, while giving a greater percentage of the stride to recirculation.

To take advantage of this trade-off, we command lower duty factors when the robot nears certain desired gaits. Using a default duty factor of $\delta = 0.83$, we allow the system to reduce to $\delta_g = 0.75$ when near tetrapod gaits, and $\delta_g = 0.63$ for a tripod gait, programmed using a simple linear funnel based upon distance from a desired gait limit cycle (Fig. 5.9), allowing 11% and 32% speed increase for the two gait types, respectively, over a pentapedal crawl.

Control of duty factor necessitates modification of a gait's phase offsets, however, with an algebraic relationship relating the two (5.5). As duty factor is the percentage of clock phase dedicated to stance, a leg 50% through stance for one duty factor has a different phase angle than for a different duty factor. Depending upon whether a leg is in stance or recirculating, we make a change to the leg's phase offset to guarantee continuity of the leg's motion:

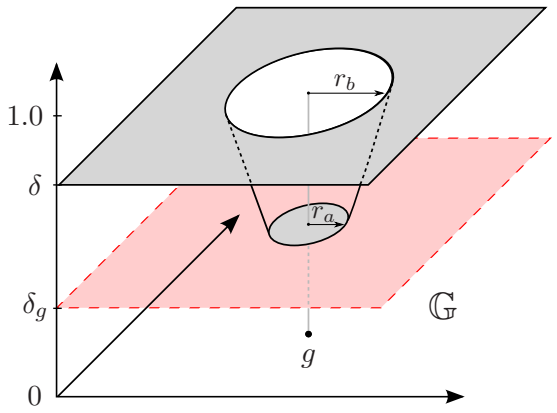


Figure 5.9: To control duty factor of a behavior, we place a linear funnel within the phase space, $\mathbb{T}^N$. A robot is allowed to use lower duty factors, $\delta_g < \delta$, only when close to a desired gait, g. r_a and r_b are gait distance radii used to design the funnel.

$$\Delta\rho_i = \begin{cases} -\frac{\psi_i}{\delta_i}\Delta\delta_i & \text{if } \psi_i \in [0, \delta_i), \text{ stance} \\ \frac{\psi_i}{\delta_i}\Delta\delta_i & \text{if } \psi_i \in [\delta_i, 1), \text{ recirculation} \end{cases} \tag{5.5}$$

While this algebraic relationship does modify the phase offsets of a gait, it does not change the ordering of leg timings, and performs control only near to desired gaits of convergence. Empirically, this system does not affect the stability properties of our gait regulation controllers.

Reducing a gait's duty factor increases the length of the recirculation stroke, thus a second speed improvement can be realized by increasing stride frequency while keeping recirculation time constant. For a tripod gait, the changing of duty factor from $\delta = 0.83$ to 0.63 allows the gait's stride rate to increase by a factor of 2.2, realizing an even greater speed increase (for a total of 187% increase in robot speed over a crawl gait).

5.3.3 Gait Regulation and Basin Management

Our final layer of control to complete our feedback behavior is gait regulation. We maintain a collection of different gait regulation controllers, each designed to realize certain gaits, and switch between controllers when necessary. Each of these controllers has a domain of the full 6-torus, thus it is of critical importance that they avoid the $\mathcal{P}$-obstacles we discuss

in §3.1.1. As the gait regulation controllers we have developed operate on a cellular decomposition of the torus, naturally avoiding the obstacle sets, they are designed to converge to preferred gaits while avoiding dangerous recirculations of the wrong sets of multiple legs.

The first controller arrangement realizes the pentapedal crawl gaits, Φ_{crawl}, activating all 15 repulsion component functions to partition phase space into a collection of 120 different basins of attraction. With a high number of basins distributed throughout phase space, as well as using the stable crawl gaits, this gait regulation system is useful when requiring slow stable locomotion with an automated defense against perturbations while climbing. Pentapedal crawl gaits operate with a duty factor of $\delta = 0.83$.

Tetrapod gaits are achieved through a second gait regulation system containing four basins of attraction. The allowable speed is greater than the crawl system, while still offering a selection of possible gaits to react to sudden perturbations. We only use the "diagonal" tetrapod gaits, those that do not recirculate contralateral nor ipsilateral legs together. Of the four possible gaits, each has an equally sized basin of attraction, as empirically shown in §4.5. The tetrapod gait regulation controller uses a duty factor of $\delta = 0.75$ when close to the preferred gaits, and $\delta = 0.83$ everywhere else.

A final controller converges globally onto the limit cycle of the alternating tripod gait, our fastest, but least stable, locomotive gait. By constructing a global limit cycle and navigating phase space to avoid obstacle sets, this system is guaranteed to converge to a single limit cycle, from even distant starting gaits. When nearby the tripod gait, the duty factor reduces from the default value (0.83) to $\delta = 0.63$, to achieve faster locomotion.

Controller selection is activated by a human operator, manually selecting the controller arrangement to use.

5.4 Concluding Remarks

With specific application to the RiSE robot, a general purpose climbing robot, we have introduced a complete climbing behavior designed for climbs of vertical exterior building surfaces, such as stucco and brick. Our approach to behavior design begins with an open-loop gait, tuned for the task of climbing these specific building surfaces, that is paired with reactive feedback controllers to attempt to guarantee some climbing task, while duty factor

and gait regulation controllers modify properties of gait timing and attempt to keep the robot safe while continuing to climb.

Avoiding the loss of static stability, during a climb, is of critical importance when applying feedback. The feedback controllers we design operate on gait parameters and must necessarily modify an open-loop gait's timing as they make corrections for errors in ground reaction forces and other measures of the climbing task. While the robot deviates from preferred gait timings, duty factor modification attempts to control the size of the dangerous gait timing set, $\mathcal{P}$-obstacles, while our gait regulation control methods avoid the obstacles while converging back to known suitable gaits.

Portions of the behavior are fully autonomous from the robot's perspective, with control performed for leg motions, error recovery and gait regulation. Currently, a human operator drives the robot, dictating overall stride rate (climbing speed), steering commands, and selection of a gait regulation controller depending upon desired gait set. Thus, the human controls the high-level selection of behavior type, while the robot manages limb-level control of motion to achieve the selected behavior and gait.

Chapter 6

Experimental Results of Robotic Climbing

We demonstrate our approach to gait regulation as a critical component of a robust climbing behavior for a robotic hexapod. When paired with reactive force controllers that necessarily deviate from a desired gait, our gait regulation methods safely return the robot to specific gaits, as well as allow the robot to switch between gaits while locomoting.

6.1 Introduction to Data Sets

A total of four sessions of experiments are included in the analysis of climbing behaviors. Each experiment session tested the robot with slightly different controller arrangements, as well as of different test capabilities.

The first dataset comes from a series of climbing demos and tests conducted at the Southwest Research Institute[1] (SwRI). As part of these tests, the robot performed multiple climbs of a multistory building on-site at SwRI.

SWRI_2006_04 : two successive climbs of a stucco building, using an early version of the behavior described in §5, with pentapedal gait regulation. Approximately 32 minutes of robot data from total climbing distance of over 24 meters.

Successive experiments were conducted in a lab setup. Using a commercially available wall similar to the SwRI surface used in SWRI_2006_04, consisting of crushed quartz gravel embedded in a resin backing[2], climbing experiments were performed on-site at

[1] Southwest Research Institute, San Antonio, TX, http://www.swri.org

[2] Stoneflex Stone Aggregate Panels—CEP Panels, Inc., Naperville, IL

Boston Dynamics, Inc.[3]. Three separate sessions of robot climbing were performed, consisting of:

BDI_2007_06 : To elucidate the effect of each component of our climbing behavior, various arrangements of the behavior controller were tested. This included pentapedal climbing on a vertical test wall. Total climbing of 19 meters, approximately, 37 minutes total.

BDI_2007_11 : To test the robot's ability to transition between various gait regulation controllers, tests were conducted on the climbing wall, angled to 70 degrees. Total climbing of approximately 6 meters, 9 minutes of data.

BDI_2008_02 : To accurately collect statistics related to ground reaction forces when comparing open-loop gaits to the full feedback climbing behavior, tests were conducted on a 65 degree wall for each of the pentapod, tetrapod, and tripod gait regulation controllers. Total climbing of 20 meters, approximately 20 minutes of robot climbing data.

6.2 Effect of Behavioral Feedback Control

We first study the overall effect of our feedback climbing behaviors. The complete behavior, with its various components of feedback control, provides greater robustness and climbing utility over an open-loop gait, resulting in extended climbing run-times, reduced occurrences of dramatic slip events, and more regular ground reaction forces when climbing.

6.2.1 Behavioral Effect of Controller Components

Using dataset BDI_2007_06, we compare the performance of various arrangements of the components of the climbing behavior. Individually testing the effect of each controller component allows us to note how each adds to the overall robustness of the climbing behavior.

Noting the related sections of §5, the controller arrangements tested were:

Open-Loop Gait (*OL*) : the robot moves its legs using cyclic feedforward motions, with no task-level feedback. §5.2

[3]Boston Dynamics, Inc., Waltham, MA

Traction Force (*T*) : the open-loop gait is augmented with traction force control which causes the feet to vary speed while in stance. §5.3.1

Normal Force (*N*) : the open-loop gait is additionally searching for the climbing surface, measured via normal force. §5.3.1

Pawing (*P*) : the open-loop gait is run; however, if a foot fails to contact the surface, pawing causes it to retry. §5.3.1

Traction plus Gait Regulation (*T+GR*) : similar to *T*, but with the addition of pentapedal gait regulation to keep the legs out of phase. §5.3.1 and 5.3.3

Complete Feedback Behavior (*FB*) : the robot executes all of the above controllers together simultaneously to create the full climbing behavior.

Three successive 1 meter climbs were performed for each controller arrangement. Controllers containing traction force control had the ability to execute turning commands to keep the robot climbing straight (as described in §5.3.1). In all other tests the robot was allowed to veer slightly until the 1 m mark was reached. If the robot slipped before reaching 1 m, the accumulated climbing distance was recorded and averaged for statistical purposes.

Joint angles, motor currents, gait parameters, and forces (measured via the 3-axis force sensors on each foot) were logged for each climbing run. Data were grouped together by controller type, and numerical analysis was performed. Computed values are:

Stance Count (*SC*) : the desired number of legs in stance was averaged (with an associated variance). Higher values indicate a more stable gait.

Load Count (*LC*) : the number of legs carrying load (defined as an individual leg carrying traction force of at least 2 N and adhesion force greater than zero). Higher values indicate more stable distribution of force, and are correlated with duty factor.

Stance Force (*SF*) : the traction force in Newtons, measured during stance and averaged over time for all legs. Smaller variance is desired. The expected value is close to the total effect of gravity on the body, but deviates somewhat between behaviors.

Velocity (*V*) : an average ground speed, calculated by distance climbed over time, in cm/s.

Distance to Failure (*DF*) : the total distance in cm climbed by the robot until a failure occurred (defined as the robot falling off the wall). For climbs of less than 1 meter, multiple climbs were averaged together. Multiple successful climbs of 1 meter accumulate distance, until a failure occurs.

Results of analysis using these values are shown in Table 6.1. Three runs were performed for all of the controllers except the Complete Feedback Behavior, FB. For each of these controllers, the robot failed at least once. Ten runs were done for the Complete Feedback Behavior, in which the robot was run until failure, after a total of 9.6 m of climbing.

Table 6.1 is useful to determine the effect of each controller on climbing performance. For example, the wide variance of traction force and large discrepancy between stance count and load count indicate that *OL* is not very successful at climbing. *T* minimizes the variance of force during stance; however, it does so by modifying gait timing, and thus robot does not climb very far before slipping (note the wide variance of stance count). Not surprisingly, *P* achieves a better load count than *O* or *N*, but quickly slips due to bad gait timing. Note that *T+GR* excels in most of the calculated numerical values, particularly high stance count and load count. The complete controller, *FB*, may not score as well as *T+GR*, but the added robustness from incorporating all control strategies together results in a behavior that climbs nearly three times as far as *T+GR* and an order of magnitude farther than all other control approaches. Furthermore, while all behaviors were commanded with identical stride rates, the overall locomotion of *FB* has the highest locomotion speed, indicating less overall slipping.

6.2.2 Control Results for Various Gait Types

By studying the salient statistics of the robot's grasp and force patterns of locomotion, we show how feedback behaviors increase the reliability of locomotion produced across a variety of gait regulation controllers. The analysis we now discuss is similar to the previous section, but is tested across the various gait regulation controllers: the alternating tripod and tetrapod gait regulation controllers (§5.3.3) are tested, in addition to the pentapedal crawl system.

For each gait type, we study both open-loop versions of the climbing behavior, as well as feedback behaviors utilizing gait regulation. While RiSE is capable of climbing a vertical

Table 6.1: Analysis of behavioral controllers for climbing. Bold values indicate the best-performance controller arrangements for each measure.

Controller	SC $(n \leq 6)$	σ^2 (n^2)	LC $(n \leq 6)$	σ^2 (n^2)	SF (N)	σ^2 (N^2)	V (cm/s)	DF (cm)
OL	5.00	***0.00***	4.36	*0.57*	8.20	*26.00*	0.804	78.7
T	4.92	*0.60*	**5.07**	*0.56*	8.30	***10.52***	0.889	48.7
N	5.00	***0.00***	4.36	*0.45*	8.50	*25.56*	0.707	81.1
P	4.84	*0.28*	4.55	*0.51*	8.74	*23.51*	0.763	35.1
T+GR	**5.07**	*0.13*	5.03	***0.38***	8.20	*12.10*	0.871	293
FB	4.98	*0.20*	4.67	*0.41*	8.61	*13.16*	**0.895**	**960**

wall using the climbing behaviors described here, its current capabilities are limited to doing so with only the pentapedal gait regulation controller (as discussed in the previous section). The tests studied here, data set `BDI_2008_02`, are performed at 65 degrees, allowing fair comparison between open-loop and closed-loop behaviors, for all three gait types, without incurring dramatic failures during climbing. The lower climbing angle also allows the robot to accurately measure foot traction forces, as some of the foot sensors encounter saturation limits when utilizing the alternating tripod gait on higher angle surfaces.

A total of 18 meters of climbing were performed on a 65 degree wall, three meters for each of the six controller arrangements.

To compare overall stability of the robot's grasp to a climbing wall, we compute similar statistics as the previous section, including load counts (*LC*), stance forces (*SF*), and robot velocity, as before. Table 6.2 shows the overall results. In addition to the previous statistics, we also show:

Body Force (*BF*) : the sum of traction force over all legs of the robot. This value give an overall idea of regularity of force acting upon the body during locomotion. Higher variances indicate greater irregularities in overall traction force.

Body Torque (*BT*) : an estimated torque value, computed by comparing the traction forces for individual legs between the left and right sides of the body. Non-zero values

indicate lack of robot turning, while lower variances correspond to more stable grasp over time.

The lower variances associated with feedback behaviors, compared to their respective open-loop controllers, particularly in total body force, is due to more regular locomotion, overall. The body is less likely to be sharply torqued in either direction, and the overall force the body is subjected to is more uniform. The overall torques do signify a turning bias to the feedback behaviors, however this is most likely associated with the robot pawing legs on one side occasionally to regrasp the surface, coupled with occasionally saturated force sensors. The overall variance is reduced, however, suggesting regularity in the torque value.

6.2.3 Occurrence of Climbing Slip Events

To study the effect of more stable locomotion, we compare the occurrence of dramatic slip events, the marked drop in ground reaction forces by individual feet or the whole body, during climbing. The feedback behaviors greatly reduce the number of slips that occur during climbing.

Studying the various gait controllers from data set `BDI_2008_02`, we note slip events as follows:

Foot slip : During the middle of stance, if a foot dramatically loses traction force of 6 N or greater in less than 0.1 seconds, we consider this to be a slip. 6 N was chosen as it is a majority of the traction carried by an individual foot during stance.

Body slip : If the total traction force carried by all six legs drops by roughly 20 N in a similar amount of time (0.1 s), we consider this to be a body slip. 20 N is roughly half of the force normally carried by the robot.

As can be seen in Table 6.3, the feedback behaviors reduce slip events for both individual legs as well as the whole body. Note, however, that the tripod gaits still incur a very large number of slips, particularly for middle legs (legs 2 and 5), suggesting the climbing surface is indeed a challenge for this gait type.

Table 6.2: Force Analysis of Regulation Controllers

	LC	σ^2	SF	σ^2	BF	σ^2	BT	σ^2	V
Units	$n \leq 6$	n^2	N	N^2	N	N^2	Nm	$(Nm)^2$	*cm/s*
Pentapod FF	4.12	*0.69*	8.79	*29.00*	46.32	*28.12*	-0.02	*0.84*	1.245
Pentapod FB	4.38	*0.61*	8.99	*13.47*	49.43	*21.94*	0.04	*0.44*	1.216
Tetrapod FF	4.15	*0.53*	9.66	*26.06*	46.76	*32.97*	-0.01	*0.44*	1.894
Tetrapod FB	4.33	*0.84*	9.55	*15.10*	49.05	*24.78*	0.11	*0.24*	1.801
Tripod FF	3.79	*1.09*	10.42	*22.91*	44.02	*55.94*	0.00	*1.35*	2.384
Tripod FB	3.62	*1.54*	9.54	*16.24*	44.94	*44.26*	0.02	*0.57*	2.319

Table 6.3: Occurrence of slipping for various gait types

Controller	Individual Leg Slips								Body Slips	
	1	2	3	4	5	6	Total	Per Stride	Total	Per Stride
Pentapod FF	5	9	5	3	7	3	32	0.67	7	0.17
Pentapod FB	7	5	2	1	2	1	18	0.36	6	0.12
Tetrapod FF	17	15	5	7	15	3	62	1.29	15	0.31
Tetrapod FB	7	6	3	1	5	5	27	0.55	10	0.21
Tripod FF	10	34	5	14	19	4	86	1.54	29	0.52
Tripod FB	6	30	2	5	14	1	58	1.18	10	0.20

6.2.4 Graphical Force Data Analysis

The overall stability of locomotion, for the climbing runs in `BDI_2008_02`, may also be studied via statistical patterns, per stride, of traction force for each behavior.

Figs. 6.1-6.3 show force profile analysis of the six different controller arrangements, in which aggregate measurements of force data are plotted based upon the stride for each leg. Average values, quartile distances, and upper and lower bounds are indicated for the locomotion.

The feedback behaviors presented in these figures show an improvement in force patterns, seen as tighter variances around a nominal average. Legs are less likely to lose grasp

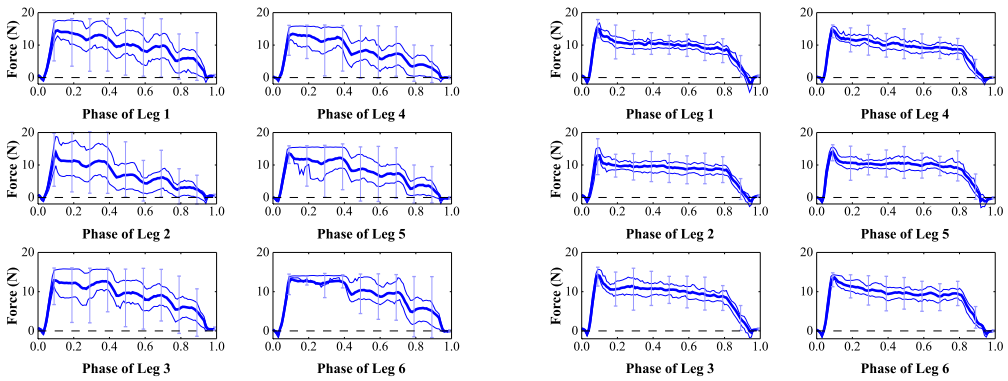


Figure 6.1: Traction force profiles for open-loop pentapod climbing (*left*) and closed-loop climbing using pentapedal gait regulation (*right*), for the six feet of the robot. Each force profile begins and ends with recirculation, thus the non-zero forces in between corresponds to stance. Thick center lines are average traction forces, while thinner surrounding lines indicate quartile values. Upper and lower bounds, excluding statistical outliers, are marked every 0.1 phase. Each set of force profiles is computed from a total of 5 meters of climbing. Note the smoother and tighter appearance of the force profiles for the feedback behavior.

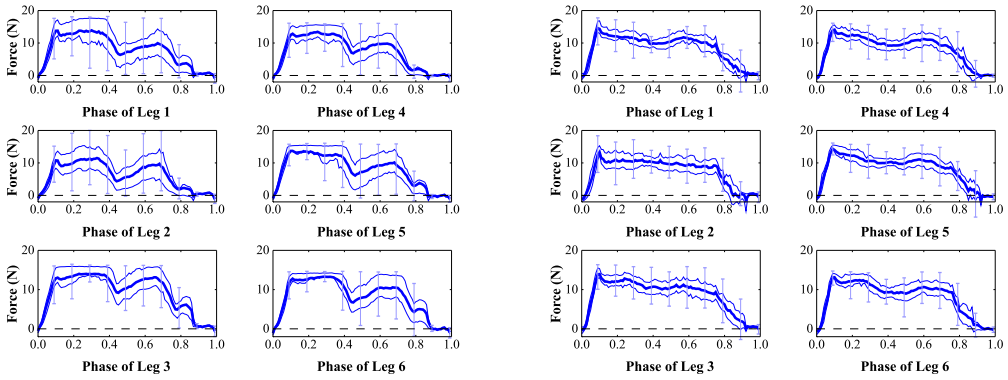


Figure 6.2: Traction force profiles for open-loop tetrapod climbing (*left*) and closed-loop climbing using tetrapod gait regulation (*right*).

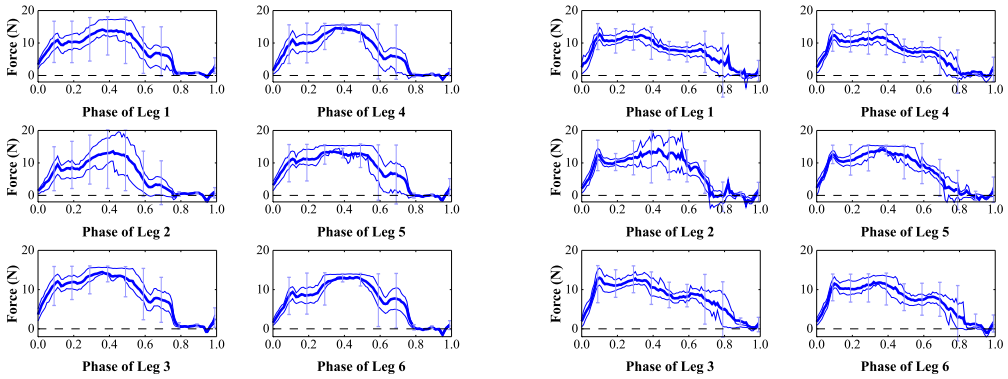


Figure 6.3: Traction force profiles for open-loop tripod climbing (*left*) and closed-loop climbing using tripod gait regulation (*right*).

over a stride (note the difference for pentapedal crawl behaviors), maintaining stance forces over a stride and reducing sharp discontinuities.

6.3 Regulating Robot Speed via Duty Factor Control

The incorporation of duty factor control, as described in §5.3.2 allows RiSE to adapt its stance-recirculation timing based upon proximity to a desired gait. This adaptation allows the robot to produce higher speeds when at lower duty factors, while conversely reducing overall stability of locomotion.

We first study the speed improvement from activating duty factor control for a tripod gait. In all of our experiments, stride rates are normalized such that leg recirculation time, the limiting factor in robot climbing speed, is constant. Thus, gaits with lower duty factors (greater percentage of stride dedicated to recirculation) have faster stride frequencies. With an extra 2 meters of climbing (conducted as part of dataset `BDI_2008_02`, but separate from the other 18 meters of climbing), we compare an alternating tripod gait regulation system with and without duty factor adjustment. In both climbing runs, flight recirculation time is kept constant, thus determining robot speed via commanded duty factor. As shown in Table 6.4, allowing the robot to use lower duty factors results in a dramatic increase in climbing speed.

We have already shown basic evidence of the speed-stability trade-off that exists for gaits of varying duty factors. This is best shown in the climbing speed differences of Table 6.2. It is also true that the current robot can only climb vertical surfaces when keeping

Table 6.4: Tripod Gait Regulation, with and without duty factor adjustment.

Duty Factor Adjustment	Duty Factor Value	Climbing Speed (*cm/s*)
Disabled	Fixed at 0.83	1.05
Enabled	Ranging between 0.63 and 0.83	2.35

a total of five feet in contact with the ground, thus offering greater stability with a slower, pentapedal gait.

Using the 18 meters of comparative climbing data from `BDI_2008_02`, we further compare how duty factor has an effect upon speed and stability. Fig. 6.4 plots the various average duty factors, for each of the six climbing controllers tested in §6.2.2. As we can see, as the duty factor reduces, robot climbing speed increases, while keeping recirculation time constant. Feedback behaviors, with the various modifications being made to the gait during locomotion, particularly with duty factor and speed control performed for the tripod and tetrapod, climb at slightly slower rates than their respective open-loop gaits, but do so with greater stability due to feedback.

This stability, per behavior, is broadly noted in Fig. 6.5. Average duty factors are plotted with statistics on the total body traction force (sum of traction force for all legs), for the six different behaviors. While the pentapedal and tetrapod climbing gaits have reduced variance in total traction force, the tripod gait—for both open-loop and closed-loop behaviors—has

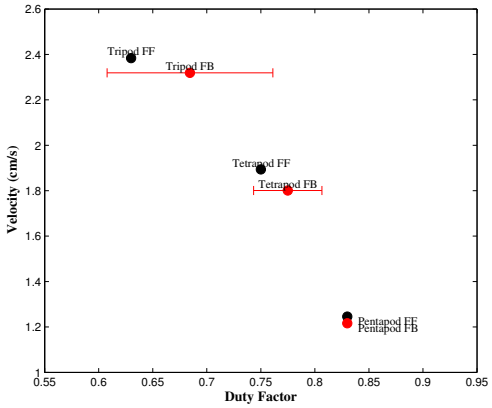


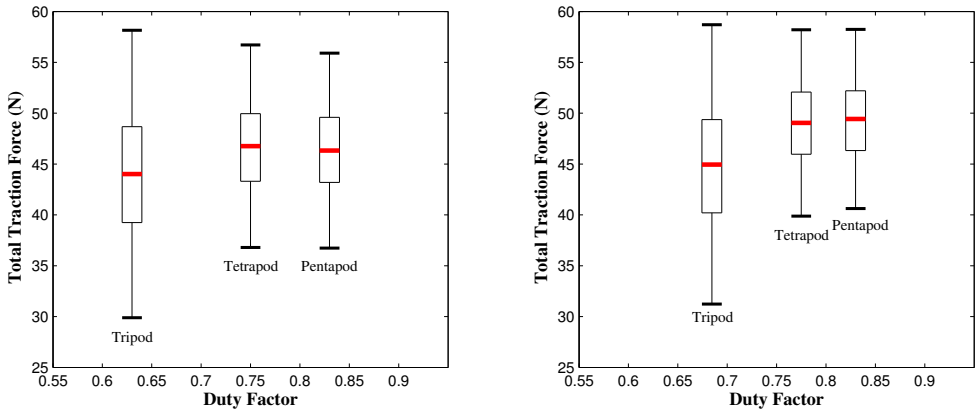
Figure 6.4: Velocities of various behaviors (Table 6.2) plotted for the average duty factors of each controller. For feedback versions of the tripod and the tetrapod, where duty factor control is active, one standard deviation spread is noted.

relatively high variance. The tripod gaits also measure less total force, but this is presumably due to sensor saturation of the robot's middle legs.

The alternating tripod gait, with its dramatic increase in speed, is very useful from a behavioral point of view, but can only be used with surfaces that can support the gait's low stability. Tetrapedal and pentapedal gaits offer greater levels of stability, but locomote at slower speed.

6.4 Design, Selection, and Execution of Gait Regulation Basins

Given the various useful gaits for locomotion tasks, we now study how our various gait types, with the explicit design of their associated gait regulation controllers, contribute to overall locomotion. Two aspects of the management of gait regulation controllers are useful to study from our experimental data. Foremost, our use of multiple basins of attraction allows a robot to naturally converge upon a nearby gait when locomotive perturbations introduce gait discontinuities. Secondly, the ability to switch between gait regulation controllers during locomotion allows a robot to take advantage of the trade-offs between fast vs. stable gaits, all while actively locomoting up a surface.



(a) Feedforward Gaits

(b) Feedback Behaviors

Figure 6.5: Comparisons of duty factors, averaged for each behavior, compared to total traction force, Body Force (BF). Force is shown for each behavior using box-and-whisker plots

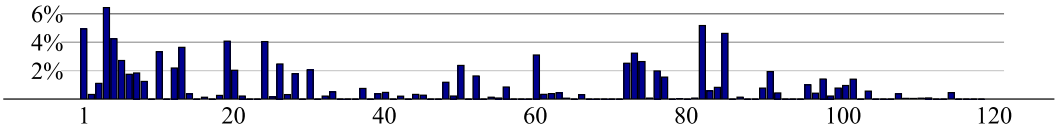
6.4.1 Climbing with Multiple Basins of Attraction

Our gait regulation controllers often contain multiple basins of attraction, each associated with a stable limit cycle, such as the crawl gait systems. These basins serve as an automated defense against perturbations during locomotion, allowing the robot to transparently switch between gaits while climbing. In this section, we show how, when presented with climbing runs of varying difficulty, the robot takes greater advantage of this transparent switching as the climbing becomes more difficult, the multiple basins thus helping the robot on challenging surfaces.

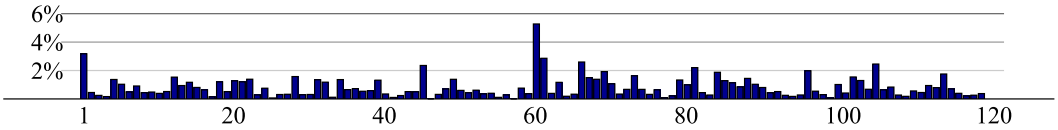
Using the crawl gait system, with its 120 different basins of attraction, we analyze data from demonstrations of the RiSE robot climbing a multistory building at the SwRI test facility. In addition to a public demonstration of the robot performing an untethered 10.9 meter climb of a multistory building, several additional data collection runs were performed during exhaustive testing of the RiSE platform. Two of those runs, comprising dataset `SWRI_2006_04`, are studied in this section. An *endurance run* required the robot to climb as far as it could on a single battery pack before ultimately running out of power (9.35 meters in 17:22, 0.90 $\frac{cm}{s}$). Another climb, the *speed run*, commanded the robot to recirculate its legs at a faster rate, resulting in an overall faster climb of the entire building (11.05 meters in 15:01, 1.23 $\frac{cm}{s}$). The speed run was commanded to climb 50% faster, but in actuality only climbed 37% percent faster than the endurance run. This disparity is an indication that the robot suffered from more frequent and more significant slippage while climbing at higher speeds.

For each climb we analyze the times at which legs swap order in a gait, as the robot switches from crawl gait basin to basin. By collecting information about these basin switches over each entire run, we derive statistics regarding basin usage and switching. Fig. 6.6 shows the distribution of the robot's usage of the 120 different basins, in terms of percentage of climbing time, for the two different experiments.

While each histogram appears largely as noise, there is less variance for the speed run, indicating that the robot switched between and utilized the various gaits more evenly. Studying the average number of strides spent in each basin, the speed run, while recirculating legs 50% faster (but only 37% increase in actual climbing speed), switches between basins over



(a) Histogram of gait usage, endurance run, $\sigma^2 = 1.77$



(b) Speed run, $\sigma^2 = 0.568$

Figure 6.6: Distribution of the robot's usage of different gait basins throughout sustained climbing runs. As the robot locomotes faster and incurs greater slippage, as in the speed run, the robot switches basins at a greater rate, with a more even distribution of gait usage.

twice as often as the endurance run. The speed run switches basins every 0.47 strides, compared to 0.99 strides for the endurance run. The maximum time spent in a single basin is 4.77 strides (20.66 sec) for the speed run and 9.83 strides (64.7 sec) for endurance.

The use of a multitude of basins of attraction, distributed over the phase space of gaits, allows the robot to switch between preferred limit cycles transparently, utilized frequently during challenging climbing.

6.4.2 Purposeful Switching between Gait Regulation Controllers

Gait switching is also possible through purposeful changes to the gait regulation controllers, switching between the controller strategies presented in §5.3.3. By switching between pentapedal, tetrapedal, and tripod gait regulation strategies, we allow the robot to go from slow yet stable gaits to faster gaits when desired during climbing. With our careful specification of properties of convergence, our gait regulation controllers are well-suited to the task of gait switching.

While gait switching is possible through the use of precomputed transition paths through phase space, such as in [24], this approach contains a combinatorial problem in the total number of gaits considered, and precludes the use of task feedback control laws while transitioning. Instead of using fixed gait transitions, our method for switching between gaits is to simply swap between gait regulation controllers and use the natural dynamics of the

gait regulation system to find a path between gaits. Given potentially distant starting conditions, far from a desired gait's limit cycle, the convergence properties we have designed into our gait regulation controllers become extremely useful. Foremost, the fact that phase space obstacles are encoded within the controllers allows the robot to follow natural paths avoiding them when possible. Secondly, the ability to design single global attractors, without local minima, or alternatively to include multiple basins that provide full coverage of phase space, guarantees that the system will always converge toward a desired gait upon switching.

To test our approach for gait switching, we command the robot to switch back and forth between a pair of gaits five times, a total of 10 gait switches per pair. With three different pairs of gaits (pentapod and tetrapod; pentapod and tripod; tetrapod and tripod), this constituted 30 individual gait transitions.

In experiments, the robot never failed to find a path between gaits, successfully transitioning at all times, while managing to continuously climb and balance foot forces. Examples of gait switching are shown in Figs. 6.7-6.10.

Since the task feedback controller introduces gait disturbances while actively balancing foot forces, it may not be possible for our system to exactly reach a desired gait while climbing. Thus, we measure the success of each gait switch by the robot reaching a certain distance from the desired gait, the average gait distance that each controller keeps the robot during climbing. The average amount of time and strides necessary for each transition is shown in Tables 6.5 and 6.6.

The system takes the longest amount of time to transition from pentapod to either tetrapod or tripod gaits, due to the potentially distant starting distance from a desired limit cycle. When switching to a pentapod gait, however, with its 120 different basins of attraction, the system takes very little time to settle in at a nearby gait.

With the inclusion of operator-commanded gait switching, the robot has the ability to switch between controllers at any given time. Suggested applications include, for instance, going from a highly angled wall, on which a faster tripod gait is successful, to a 90 degree wall, on which, with our existing robot, only pentapod gaits climb. The ability to include task feedback, such as force control, with the dynamical system for gait regulation, allows the robot to switch gaits while actively controlling its task as well.

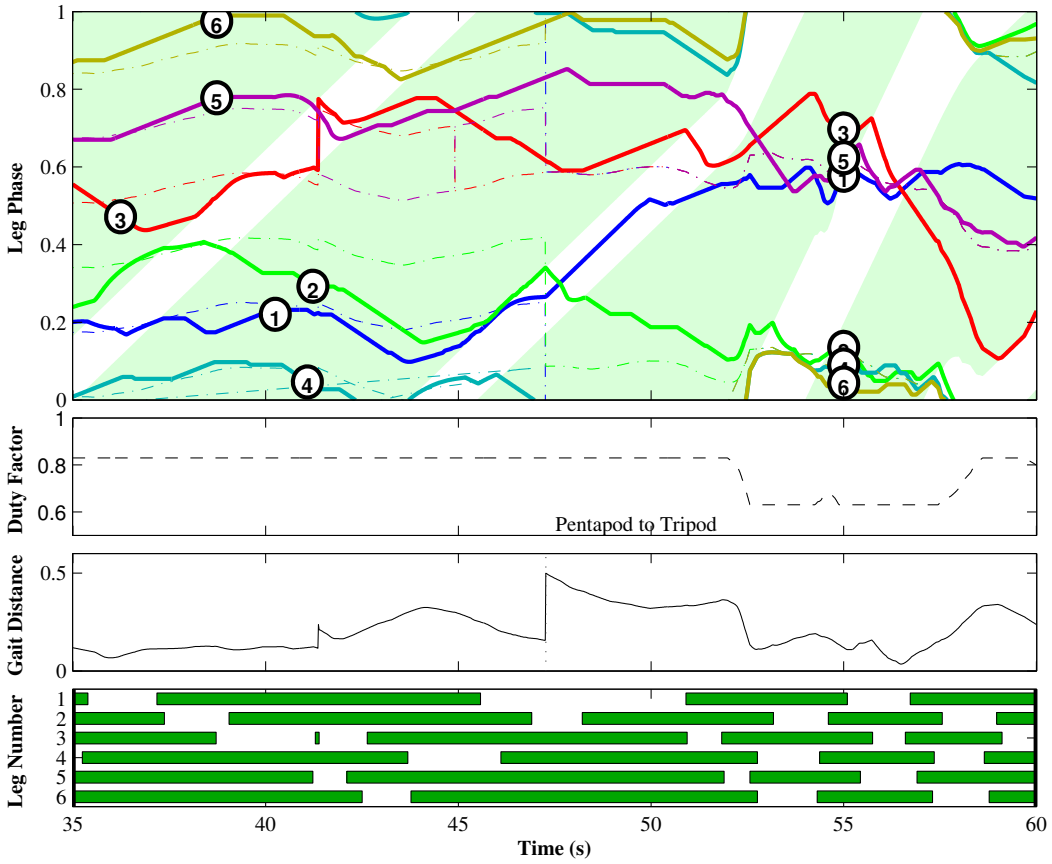


Figure 6.7: Example of RiSE switching gait regulation controllers while climbing. At $t = 47.5$ sec, the robot switches from pentapedal gait regulation to tripod, taking approximately 5 sec to exhibit tripod-like locomotion. The top plot shows individual leg phase offsets, solid lines, with current desired phase offsets (dashed lines). When the lines are within the lightly shaded green regions, task controllers modify the leg phases to perform force control (with an example of pawing occurring around $t = 42.5$ sec). In unshaded regions, legs recirculate and undergo gait regulation. The 2nd and 3rd plots show commanded duty factor and geodesic distance to the desired gait's phase offsets, while the bottom plot shows the patterns of stance for all six legs. As gait distance decreases, the robot lowers its duty factor, thus affecting the shape and length of stance regions.

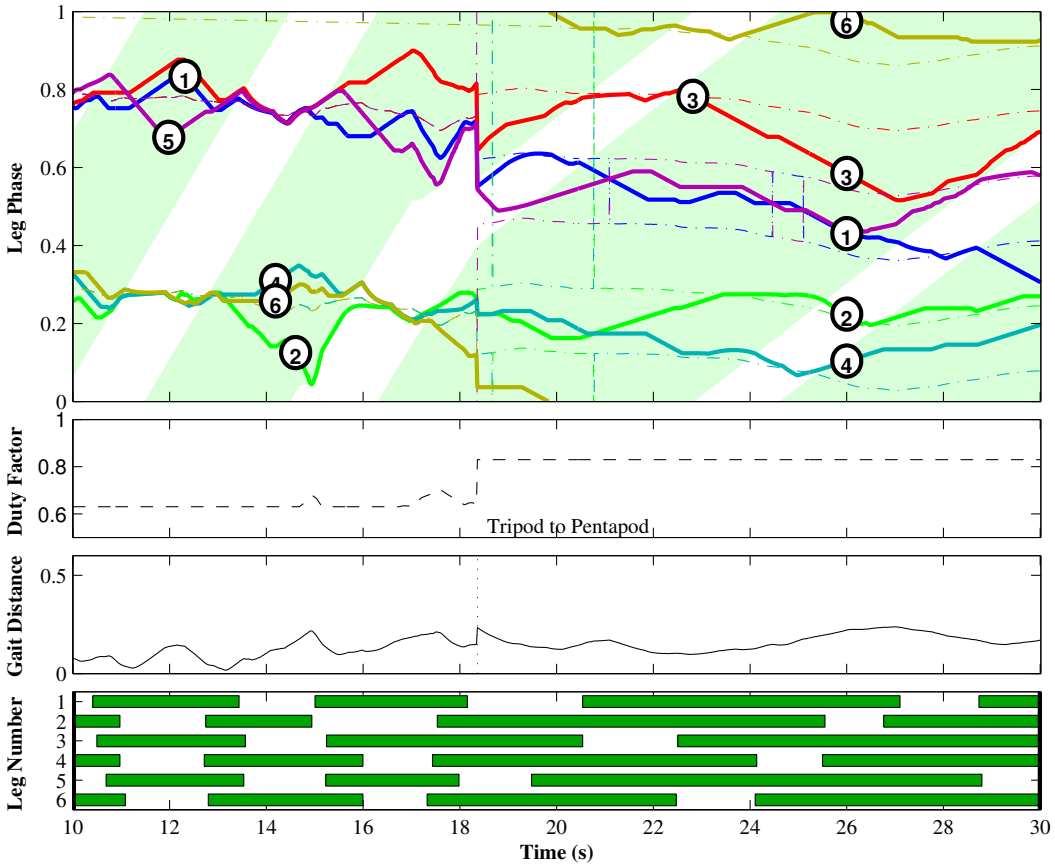


Figure 6.8: Example of RiSE switching gait regulation controllers. The robot, while climbing, executes a switch of gait regulation strategy from a tripod gait back to the pentapedal crawl gait. The system quickly settles to the nearest crawl gait.

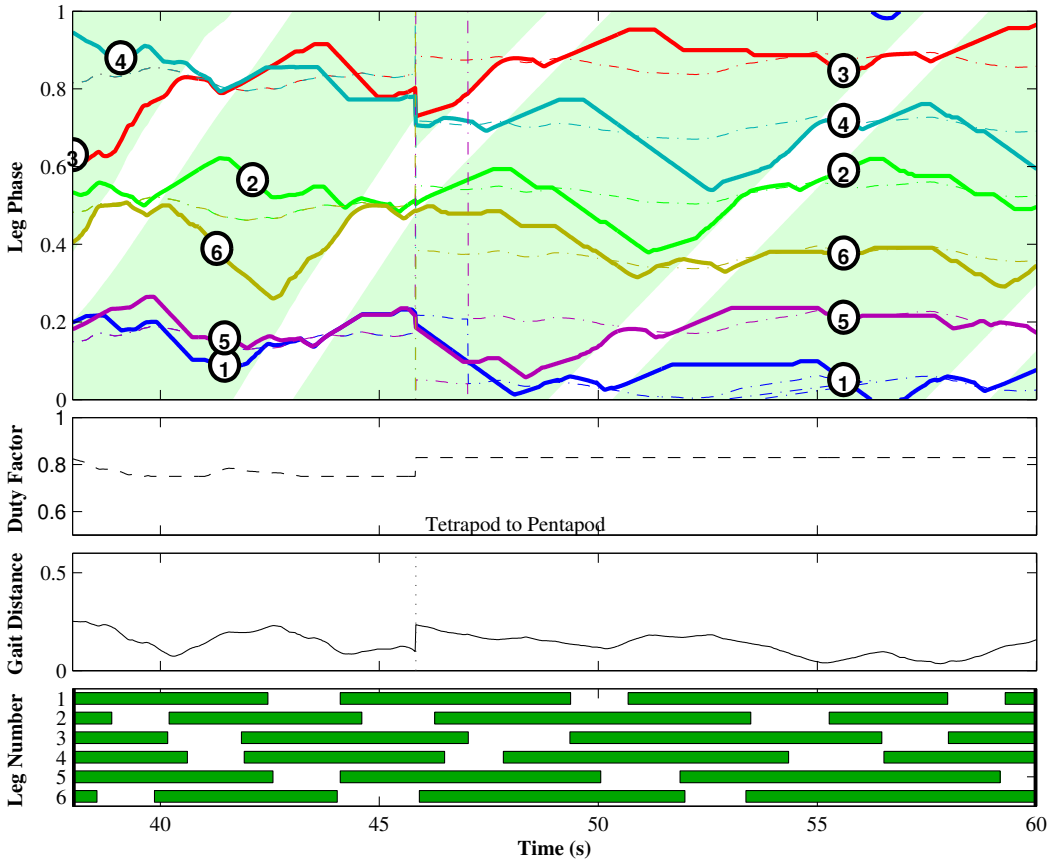


Figure 6.9: Example of RiSE switching gait regulation controllers. Switching from a tetrapod gait to a pentapod gait, the system also easily transitions.

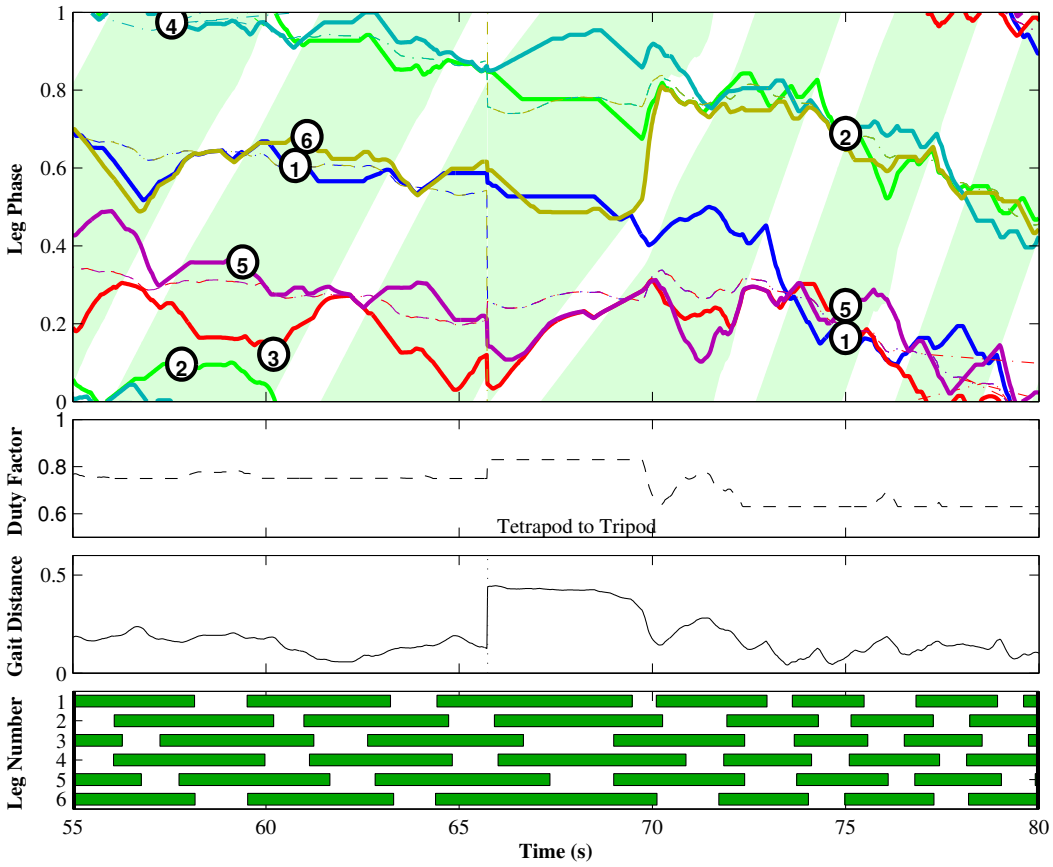


Figure 6.10: Example of RiSE switching gait regulation controllers. Going from a tetrapod to a tripod gait, the system must careful manage leg ordering to complete the tripod.

Table 6.5: Average switching times, in seconds, between various gait types

		Desired Gait		
		Pentapod	Tetrapod	Tripod
Previous Gait	Pentapod		9.15	9.41
	Tetrapod	3.11		3.90
	Tripod	2.66	3.59	

Table 6.6: Average switching times, in terms of gait strides, per gait type

		Desired Gait		
		Pentapod	Tetrapod	Tripod
Previous Gait	Pentapod		1.13	0.95
	Tetrapod	0.37		0.53
	Tripod	0.26	0.50	

6.5 Concluding Remarks

This chapter studies the performance results of our approach to behavior control for the task of climbing a flat exterior building surface. We study a multitude of climbing runs to analyze the empirical effects of our controller components, note the differences amongst various types of preferred gaits, and show the application of gait switching during climbing.

The various controller components, including our reactive feedback control laws on gait parameters in addition to gait regulation, each offer benefits to the overall climbing behavior, with the application of all components xtogether resulting in stable climbing and extended run-times. The use of gait regulation controllers for a variety of gait timings allows us to expose a trade-off between the stability and speed for different gaits. Finally, the ability to actively switch between preferred gaits while climbing, utilizing the global properties of convergence for our gait regulation controllers, allow the robot to safely transition from gait to gait during climbs.

Chapter 7

Conclusions and Future Work

This thesis shows methods of gait regulation that are, when paired with other gait-based controllers, critical to the success of an underactuated climbing robot. While behavioral feedback controllers try to maintain the task of climbing a challenging surface, gait regulation control avoids dangerous leg timings while converging upon desired gaits of choice.

The contributions of this thesis cover a wide range, from control methods for legged robots, to discussions of high-dimensional toroidal spaces, onto realizing behaviors and concrete results with a physical robot platform.

Gait regulation is discussed in depth, the use of a control system to force a robot to converge upon desired gaits while avoiding potentially dangerous gait timings. We discuss the convergence properties that are critical to these control systems—the necessity of boundary cuts on a high-dimensional torus—and introduce a cellular decomposition of the torus that provides a basis for designing convergent control laws. After discussing the existence of $\mathcal{P}$-obstacles, and their relation to dangerous recirculations of multiple legs, we show how the cellular decomposition also approximates avoidance of these obstacle sets. Most importantly, a hybrid control algorithm is used to navigate the cellular decomposition in order to realize controllers that converge to a variety of gait timings. Simulation results show that this approach improves performance over prior methods.

With the design of a complete feedback behavior for a climbing robot, incorporating force controllers that modify a gait based upon ground reaction forces, we show gait regulation to be critical in the robustness of the climbing task. By keeping the robot close to stable gaits during locomotion, run-time performance is dramatically increased compared to

open-loop gait control. Furthermore, the ability to transition amongst gait types, by switching gait regulation controllers, allows the robot to move between slow but stable gaits and fast but less stable gaits, a trade-off that adds significant functionality to the climbing task.

In summary, the core contributions may be listed as:

1. An algorithmic approach for gait regulation that avoids dangerous gaits by planning paths of convergence over a cellular decomposition of a high-dimensional torus, offering affordance in selection of different gait types.

2. A study into the control effects of gait regulation, noting necessary implications of control when placing attractors and repellors on the torus.

3. The design of a feedback climbing behavior that builds upon a successful open-loop gait by adding control laws to both the geometry and timing of the gait, using force-based reactive control laws as well as gait regulation.

4. Experimental results with a climbing robot, analyzing over 50 meters of climbing data, to show both climbing robustness as well as the utility of gait regulation control to allow transitioning between different gait types upon command.

7.1 Future Work

Many additional studies follow from the results of this thesis, ranging from further experimental work with a robot to studies into the topological spaces that result from constraints we have defined on the torus.

Using the capabilities of the existing RiSE platform, simple modifications could add robot autonomy by utilizing current control approaches. Currently, a human operator initiates transitions between gaits by manually swapping gait regulation controllers. By studying the ways in which the robot could assess its own climbing performance, it would be of use to program the ability to autonomously decide the times at which it is useful to change gait types. When climbing becomes too difficult, the robot should switch to slow, stable gaits. When climbing is relatively easy, the robot should switch to the less stable yet faster gaits to ascend as fast as it can. This ability would allow the robot to react to sudden changes in climbing performance. In addition to this new form of autonomy, it would

be further beneficial to replace the human operator's steering command with the use of an inertial measurement unit to direct the robot straight upward while climbing

A second direction for future research lies in the idea of *preference* amongst the various gaits and leg pairs of a robot. Preference is already hinted at when designing gait regulation controllers, as we only consider certain tetrapod and tripod gaits, based upon which leg pairs we prefer to be in-phase in these gaits. This preference amongst leg pairs should be further incorporated into our algorithm for gait regulation. When navigating from distant regions of phase space, the hybrid algorithm requires pairs of legs to occasionally recirculate together to swap order, with no preference upon specific leg pairs. It would be worse, however, for a climbing robot to recirculate both of its front legs, compared to certain other pairs of legs, for instance. Preferences such as this could be included as a heuristic on the graph representation of our cellular decomposition when planning routes of convergence. A route of convergence that must travel further, in terms of gait distance on the torus, should be preferred if it follows a path that does not recirculate undesirable pairs of legs together.

A second way in which preference could be added to our current gait regulation controllers is to the leg orderings of crawl gaits. Currently, 120 different pentapedal crawl gaits each have equal likelihood of convergence. The wave gait, however, is greatly preferred by biological systems, thus it would be of interest to expand the region of convergence around such a preferred gait to increase its likelihood as desired, while decreasing the likelihood of other crawl leg orderings. This is beyond the capabilities of our proposed approaches for gait regulation control, as the use of only leg repulsion for control does not allow the preference of certain leg orderings over others, suggesting that new methods must be introduced.

While our proposed approach for gait regulation control tends to avoid dangerous gait timings better than prior methods, a full study into the nature of the $\mathcal{P}$-obstacle set is a third direction for future research. Foremost, a precise understanding of the geometry of this obstacle set should be focused upon, understanding exactly which legs of a robot, if recirculated together, would result in the robot slipping or falling. With this definition, we imagine designing control laws upon the complement space of the obstacle, the *safe* gait timings, such that our control laws could guarantee paths of convergence that do not intersect with the obstacle region. The currently proposed methods, while approximating the obstacle set, cannot make such a guarantee. Furthermore, when we apply duty factor

control as we do to vary speed and stability of the robot, the obstacle geometry changes. As such, exact avoidance of a dynamically changing obstacle set will be beneficial.

A slightly related direction for future study is in gaits that, by design, recirculate some legs more than once per stride, yet do not intersect with the $\mathcal{P}$-obstacle set. Our current work is only capable of considering gaits in which each leg recirculates exactly once. In terms of a gait cycle on the torus, the current type of gait cycles wind around the torus exactly once for each axis. These other gait types would wind multiple times for certain axes. These gaits, while different from the gaits we have currently considered, can also produce stable locomotion and their inclusion would add generality to our control methods.

At a broad level, the paradigm followed in this thesis, discussing control by identifying underlying topological spaces while noting the necessary constraints that exist upon those spaces, will potentially prove useful for other studies. In this work, we discuss gait regulation control by noting how boundaries must be introduced on a torus and identify phase space obstacles as regions to avoid. Future work will further consider constraints upon the tangent space of the torus—relationships amongst leg phase velocities in a legged system—when designing controllers for dynamic gaits. We expect this to be a rewarding study as we move to dynamic legged climbing robots.

Appendices

A Combinatorics of Gaits

Our use of stance phase offsets (§2.2) allows us to describe robot gaits as a continuum, upon which we perform continuous control to modify the gait a robot uses during locomotion. By instead discussing discrete leg orderings, and describing gaits using an alternative notation, we derive various combinatorics related to the possible number of certain gait types.

We define a gait ordering as a cyclic ordered tuple, defining the order in which the N legs of a robot recirculate. For examples, $(1, 2, 3)$ is a definition stating that, for a certain gait, leg 1 recirculates first, followed by leg 2, and finally leg 3, after which the cycle repeats.

Foremost, we note that, with a repetitive task, all cyclic permutations of a gait ordering—permutations that only "rotate" the tuple—are identified. Thus $(1, 2, 3)$ refers to the same cyclic ordering as $(2, 3, 1)$ and $(3, 1, 2)$. (For brevity, gait orderings are generally referred to by the specific ordering beginning with leg 1).

We consider two types of gait orderings: those that recirculate legs individually and those that recirculate multiple legs together. We shall use the notation $\mathsf{G}[N, k]$ to refer to the set of possible leg orderings of N legs that recirculate in groups of k.

$\mathsf{G}[N, 1]$ is the set of gait orderings that recirculate legs individually, commonly referred to as crawl gaits. When considering only those gaits that recirculate each leg once per stride, one can easily note the cardinality of the set, by simply counting the number of ways one can order N legs. To deal with the cyclic identification of tuples, the total is divided by the number of elements, N, so that identified gaits are counted only once.

$$
\begin{aligned}
|\mathsf{G}[N, 1]| &= \frac{\binom{N}{1}\binom{N-1}{1}\binom{N-2}{1}\cdots\binom{1}{1}}{N} && (7.1)\\
&= \frac{N(N-1)(N-2)\ldots 1}{N} && (7.2)\\
&= (N-1)! && (7.3)
\end{aligned}
$$

By this notation, there are $(4-1)! = 6$ crawl gaits for a quadruped. For a hexapod, there exist $(6-1)! = 120$ crawl gaits.

We now consider cases where $k > 1$, and discuss the cardinality of $\mathsf{G}[N,k]$. In gaits such as these, legs recirculate together, and we describe the gait ordering as an ordering of individual tuples. For instance, $((1,4),(2,3)) \in \mathsf{G}[4,2]$ is a trot gait, where legs 1 and 4 recirculate together as a pair, followed by legs 2 and 3. By considering the number of ways by which we can group N legs into groups of k (where N is evenly divisible by k), one can derive a similar rule for the total number of such gaits:

$$
\begin{aligned}
|\mathsf{G}[N,k]| &= \frac{\binom{N}{k}\binom{N-k}{k}\binom{N-2k}{k}\cdots\binom{k}{k}}{N/k} && (7.4)\\
&= \frac{N!}{k!(N-k)!}\frac{(N-k)!}{k!(N-2k)!}\frac{(N-2k)!}{k!(N-3k)!}\cdots\frac{k!}{k!(0!)} && (7.5)\\
&= \frac{N!}{(k!)^{N/k}N/k} && (7.6)\\
&= \frac{k(N-1)!}{(k!)^{N/k}} && (7.7)
\end{aligned}
$$

Indeed, (7.3) is the same as (7.7) where $k = 1$. Considering gaits for a quadruped in which legs are paired, there exist $|\mathsf{G}[4,2]| = 3$ such gaits (the trot, pace, and bound gaits commonly noted), and a single gait in which all legs recirculate together, the pronk: $|\mathsf{G}[4,4]| = 1$.

Similarly, for a hexapedal robot, there exist $|\mathsf{G}[6,3]| = 10$ possible gaits that recirculate legs in two groups of three (commonly referred to as tripod gaits), and $|\mathsf{G}[6,2]| = 30$ gaits where legs recirculate as three groups of two. Not all gaits are considered equal, however, and the body of this thesis focuses on developing convergent control laws to realize only a handful of these hexapedal gaits. Of the tripod gaits, only the alternating tripod, $((1,3,5),(2,4,6))$, recirculates neither ipsilateral nor contralateral legs together. Similarly, of the gaits paired into groups of two, only four such tetrapod gaits satisfy these same requirements.

To efficiently count, order, and identify the possible crawl gaits, we make use of Lehmer codes, an instance of using a factorial-based number system [1]. Lehmer codes are unique values assigned to each possible permutation of a set of N elements. As our cyclic crawl gaits have cardinality of permutation of $N-1$ elements, there exists an easy mapping from gait orderings to Lehmer codes and vice-versa. Appendix C provides example functions for

computing Lehmer codes. As our work has only focused on specific cases of grouped gaits, we have not developed a means of counting and identifying such gaits as of yet.

The reader will note that this appendix fails to discuss two other possible types of gait orderings. Foremost are crawl-like gaits that recirculate some legs more than once per stride. While these gaits are perfectly valid and may produce locomotion, they are beyond the scope of this thesis. Furthermore, gaits in which some legs recirculate together, but the number of legs recirculating varies, are also beyond our current studies.

B Cellular Decomposition of Phase Space

The hybrid controller presented in the body of this thesis performs both metric and discrete distance calculations based upon a cellular decomposition of the N-torus, the gait coordination space used in this thesis. This appendix describes informally the properties of our chosen decomposition, connecting the cellular structure to the set of robot crawl gaits. In brief, we describe the cellular decomposition using a graph representation in which vertices represent cells and edges represent adjacency relationship between cells, allowing us to discretely plan paths within our cellular decomposition.

B.1 Crawl Gait Cells

We first define the cells that make up our decomposition, based upon the crawl gaits. A crawl gait is defined as a gait that keeps all N legs of a robot out-of-phase with one another. Included in this set, with appropriate phase offset values and duty factors, are the gaits that maintain a maximun number of legs in contact with the ground during locomotion, by only recirculating legs individually. We begin with a set of N phase offsets:

$$\rho_i \in S^1 \tag{7.8}$$

$$\mathbf{x} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_N \end{bmatrix} \tag{7.9}$$

We define a crawl gait "cell" as the region of phase space in which legs take a particular cyclic ordering, for instance the ordering $(1, i, \ldots, m) \in \mathsf{Perm}[N]$, a tuple that is a permutation of the set $\{1, 2, \ldots, N\}$. As Appendix A already discusses our notation of these cyclic orderings, as well as how many possible orderings exist, we proceed to describe a gait cell, C_k, corresponding to a given leg ordering as:

$$C_k = \{(\rho_1, \ldots, \rho_N) \in \mathbb{T}^N | \rho_1 < \rho_i < \ldots < \rho_m < \rho_1\} \tag{7.10}$$

By construction, each cell is an open set of N-dimensions. In the above equation k is the cell number, identified with one of the permutations. There exist a total of $(N-1)!$ permutations of the set $\{1, 2, \ldots, N\}$ (with all cyclic permutations identified, App. A), thus there are $(N-1)!$ disjoint crawl gait cells, each a distinct cyclic ordering of N legs. The union of cells is a subset of $\mathbb{T}^N$:

$$\mathcal{C}_N \subset \mathbb{T}^N \tag{7.11}$$

$$\mathcal{C}_N := \bigcup_{k=1}^{(N-1)!} C_k \tag{7.12}$$

Identification with the Topological Configuration Space

The *configuration space* of a topological space $\mathcal{M}$

$$\mathsf{Conf}_n[\mathcal{M}] := \{(m_1, \ldots, m_n) \in \mathcal{M}^n : m_i \neq m_j \forall i \neq j\} \tag{7.13}$$

is a well studied object in topology [21].

It is readily noted that any member of $\mathsf{Conf}_N[S^1]$ must also be a member of some cell of $\mathcal{C}_N$, and that every cell must be a proper subset of $\mathsf{Conf}_N[S^1]$. Thus, the two constructs are equivalent.

A forthcoming publication will indeed show that the cardinality of the number of disjoint sets included within $\mathsf{Conf}_N[S^1]$ is $(N-1)!$, matching our cell definition.

B.2 Graph Network over Cell Complex

We use the cellular decomposition, and the equivalent group of permutations, to define a graph structure describing the connectedness of crawl gait cells, in which each node represents a cell, and each edge an $(N-1)$-dimensional "face" between adjacent cells. This graph is critical to the hybrid algorithm presented within the body of the thesis.

Let $A \in \mathsf{Perm}[N], B \in \mathsf{Perm}[N]$ be two different permutations, representing cells C_A and C_B. We say that A and B are singly connected if and only if they differ in only a single swap of leg order. Consider the following two permutations as an example:

$$A = (1,2,3,4) \in \mathsf{Perm}[4] \tag{7.14}$$

$$B = (1,3,2,4) \in \mathsf{Perm}[4] \tag{7.15}$$

These tuples are singly connected, as a swap of the pair $(2,3)$ transforms one string into the other. In essence, the gait orderings must pass through an "intermediate" gait ordering where legs 2 and 3 have equal phase, $(1,(2,3),4)$.

In terms of gait cells, a "letter swap" corresponds to the $(N-1)$-dimensional face, F_{AB}, between adjacent cells, C_A and C_B. This face is defined by exactly two legs having equal phase:

$$C_A = \{(\rho_1,\ldots,\rho_N) \in \mathbb{T}^N | \rho_k < \ldots < \rho_i < \rho_j < \ldots < \rho_k\} \tag{7.16}$$

$$C_B = \{(\rho_1,\ldots,\rho_N) \in \mathbb{T}^N | \rho_k < \ldots < \rho_j < \rho_i < \ldots < \rho_k\} \tag{7.17}$$

$$F_{AB} = \{(\rho_1,\ldots,\rho_N) \in \mathbb{T}^N | \rho_k < \ldots < (\rho_i = \rho_j) < \ldots < \rho_k\} \tag{7.18}$$

One can see that there exist a total of N such faces for each cell, each face corresponding to a different swap of consecutive legs in a crawl gait ordering. We use this property to define our graph structure, a 1 skeleton of the cells and faces.

$$v_A := \text{Vertex representing cell } C_A \tag{7.19}$$

$$v_B := \text{Vertex representing cell } C_B \tag{7.20}$$

$$e_{A,B} := \text{Graph edge between } g_A \text{ and } g_B, \text{ representing face } F_{AB}. \tag{7.21}$$

Studying the basic combinatorics of this graph definition, one can easily note that, while there are $(N-1)!$ vertices, there exist $\binom{N}{2}$ possible leg swap pairs—*types* of edges between vertices. When considering the set of all graph vertices connected by a particular leg swap, one can further see that this defines all pairs of permutations where the leg swap pair are consecutively swapped, and all other legs fill the remaining ordering of the gait. Therefore,

we have found that there exist $(N-2)!$ pairs of permutations that are connected via a certain edge type, giving the full graph a total of $\binom{N}{2}(N-2)! = \frac{N!}{2}$ distinct edges.

With this graph in hand, we can easily calculate discrete metrics on this graph by considering the application of Dijkstra's Algorithm [17] for an undirected, unit edge cost graph, which we will notate as CONNECTEDNESS$[v_i, v_j]$, the minimal number of edges between vertices v_i and v_j.

The body of this thesis considers in depth the use of crawl gait cells to describe convergent regions of gait regulation controllers. By understanding relationships between neighboring vertices on the graph, we describe locally convergent regions for various control strategies.

C Relevant Code Examples

Provided in this appendix are sample implementations of some of the code used to generate gait regulation controllers, as well as to analyze the results.

Program 1 GaitGeodesic.m: computation of geodesic between parallel gait cycles.

```
function [gaitdist,gaitvector] = GaitGeodesic(gait1,gait2)
% GAITGEODESIC computes geodesic distance between gait cycles
%
%   GAITGEODESIC(GAIT1,GAIT2) calculates the minimum distance between
%   the two gaits with phase offsets listed in GAIT1 and GAIT2.  As
%   there exist 2^N different straight-line paths between two points
%   on an N-torus, this routine must compare each.  For each path, the
%   straight-line route is projected onto the [1 1 ... 1] vector in
%   order to compute distance perpendicular to that vector.  This
%   is because gaits are defined as cycles parallel to that
%   vector.  By computing the perpendicular distance, this routine
%   efficiently calculates the minimum geodesic between any two
%   points on the two cycles.
%
%   [GAITDIST,GAITVECTOR] is returned, the geodesic distance and the
%   vector along which the two cycles have minimum distance.  While
%   GAITDIST is deterministic, GAITVECTOR is not.
%
%   Author: Clark Haynes <gch@cs.cmu.edu>, 2007

% count number of legs
N = length(gait1);

% Initialize gaitdist with high value
gaitdist = 1.0;

% Compute a Nx2 vector containing the "left" and "right" distances for
% each axis, for the N points in the gait.
diffgait = [mod(gait1-gait2,1.0) mod(gait2-gait1,1.0)];

% loop over all 2^N direct paths between the two points, our phase offsets
for i=1:(2^N)

  % each value of 'i' corresponds to a specific set of "left"
  % vs. "right" choices.  Use binary representation to do this.
  binaryStr = dec2bin(i,N);

  % build path between points by selecting distance direction for
  % each axis
  vector = zeros(N,1);
  for j=1:N
    vector(j) = diffgait(j,str2num(binaryStr(j))+1);
  end

  % project vector onto [1 1 ... 1] vector, then subtract off of
  % overall distance to compute perpendicular distance
  distvector = vector - dot(vector,ones(N,1))*ones(N,1) / N;

  % distance value is simple norm of the vector
  dist = norm(distvector);

  % the vector with minimum distance
  if dist < gaitdist
    gaitdist = dist;
    gaitvector = distvector;
  end
end
```


Program 2 LegPhaseToLegOrder.m: conversion of phase offsets into leg ordering.

```
function legorder = LegPhaseToLegOrder(phaseoffsets)
% LEGPHASETOLEGORDER converts phase offsets to leg ordering
%
%   LEGPHASETOLEGORDER takes a set of phase offsets, and computes the
%   ordering of legs based upon those phase offsets.  Assuming all
%   offsets are uniquely valued, the ordering is a deterministic
%   value.
%
%   LEGORDER is returned, a cyclic permutation of [1 2 3 4 5 6]
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2007

% get length and "condition" the values
N = length(phaseoffsets);
phaseoffsets = reshape(phaseoffsets,[N 1]);
phaseoffsets = mod(phaseoffsets,1.0);

% initialize the order to default
legorder = [1:N]';

% sort the phase offsets, simultaneously reordering leg orders
sorted = sortrows([phaseoffsets,legorder]);

% copy out leg orderings
legorder = sorted(:,2);

% make certain leg ordering begins with 1
if legorder(1) ~= 1
  index = find(legorder == 1);
  legorder = legorder([index:end 1:index-1]);
end
```

Program 3 LegOrderToGaitNumber.m: computation of Lehmer code from leg ordering.

```
function gaitnumber = LegOrderToGaitNumber(legorder)
% LEGORDERTOGAITNUMBER computes a gait number for a given leg
% ordering
%
%   LEGORDERTOGAITNUMBER(LEGORDER) takes the given leg ordering,
%   such as [1 2 3 4 5 6], and converts it to a gait number, the
%   lehmer number associated with a related permutation.  Lehmer
%   codes are utilized in this calculation.
%
%   GAITNUMBER is returned, the unique number associated with the
%   supplied ordering of N legs.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2007

% first make certain the '1' is first
legorder = circshift(legorder,length(legorder)-find(legorder==min(legorder))+1);

% the following code computes a lehmer code based upon a
% permutation, considering everything except for the first element,
% 1.  The code takes each element of the permutation and orders it
% within a second vector.  The insertion ordering gives us the
% factoradic notation.
lehmer = [0];
xset = [];
for i=length(legorder):-1:2
  for j=1:length(xset)
    if xset(j) > legorder(i)
      j = j-1;
      break;
    end
  end
  xset = [xset(1:j) legorder(i) xset(j+1:end)];
  lehmer = [j lehmer];
end

% compute lehmer code from factorial
gaitnumber = 1;
for i=1:length(lehmer)
  gaitnumber = gaitnumber + factorial(length(lehmer)-i)*lehmer(i);
end
```

Program 4 GaitNumberToLegOrder.m: reverse operation of previous program.

```
function legorder = GaitNumberToLegOrder(numlegs,gaitnum)
% GAITNUMBERTOLEGORDER calculates a leg ordering from gait number
%
%   GAITNUMBERTOLEGORDER(NUMLEGS,GAITNUM) takes a given gait number
%   and converts it into an ordering of NUMLEGS legs.  For example,
%   the 6-legged gait where gaitnum=1 is [1 2 3 4 5 6].  Because
%   there is a cycle in the orderings, this function returns
%   orderings that always begin with "1".
%
%   LEGORDER is returned, a leg ordering of N elements.
%
%   Author: Clark Haynes <gch@cs.cmu.edu>, 2007

% the set of orderings on N legs is equivalent to the set of
% permutations for N-1 legs.
legorder = Permutation(numlegs-1,gaitnum)';

% prefix the permutation with "1", adding 1.0 to all others
legorder = [1; legorder + 1];
```


Program 5 LegPairToCutNumber.m: numbering system for counting pairs of legs.

```
function cutnumber = LegPairToCutNumber(numlegs,legpair)
% LEGPAIRTOCUTNUMBER converts a pair of legs into an id number
%
%   LEGPAIRTOCUTNUMBER(NUMLEGS,LEGPAIR): for a given number of legs
%   (NUMLEGS) and a specific pair of legs (of the N choose 2 pairs of
%   legs), convert the pair into an id number.  This allows us to
%   efficiently number the pairs of a gait.  Pairs are ordered
%   numerically. [1 2] is #1, [1 3] is #2, and so on.
%
%   CUTNUMBER is returned, the id of the cut.
%
%   Author: Clark Haynes <gch@cs.cmu.edu>, 2006

% put the pair into ascending order
legpair = sort(legpair);

% terse representation of input values
i = legpair(1);
j = legpair(2);
N = numlegs;

% a simple formula exists to calculate cut number,
%   cutnumber = N*(i-1) - sum(1:i-1) + j - 1;
% when refactored, it becomes the following
cutnumber = (N-1/2)*i - i^2/2 + j - N;
```

Program 6 CutNumberToLegPair.m: reverse operation of previous program.

```
function legpair = CutNumberToLegPair(numlegs,cutnumber)
% CUTNUMBERTOLEGPAIR converts an id number into leg pair
%
%   CUTNUMBERTOLEGPAIR(NUMLEGS,CUTNUMBER): for a given number of
%   legs (NUMLEGS) and an id number for a pair of legs, this
%   function returns the leg pair.  This allows efficient numbering
%   of leg pairs.  Pairs are ordered numerically.
%
%   LEGPAIR is returned, a listing of two legs.
%
%   Author: Clark Haynes <gch@cs.cmu.edu>, 2006


i = 1;
while cutnumber > (numlegs-i)
  cutnumber = cutnumber - (numlegs-i);
  i = i+1;
end
j = cutnumber + i;

legpair = [i j];
```

Program 7 Permutation.m: efficient counting of permutations using Lehmer codes.

```
function perm = Permutation(N,lehmernumber)
% PERMUTATION efficiently computes a permutation of N elements
%
%   PERMUTATION(N,LEHMERNUMBER) uses a factoradic representation to
%   efficiently compute a permutation of N elements from its lehmer
%   number.  This code is derived from the description of the
%   factoradic numbering system on Wikipedia:
%     http://en.wikipedia.org/wiki/Factoradic
%
%   PERM is returned, a kth permutation of N elements, where k is
%   the lehmer number.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2007

% compute the factoradic, a unique factorial-base representation
factoradic = [];
lehmer_rem = lehmernumber - 1;
for j=[N:-1:1] - 1
  f = factorial(j);
  val = floor(lehmer_rem / f);
  lehmer_rem = lehmer_rem - val * f;
  factoradic = [factoradic val];
end

% the factoradic tells us where to "insert" each of our N elements
% into the permutation.
l = [1:N];
perm = [];
for f=factoradic
  perm = [perm l(f+1)];
  l = [l(1:f) l(f+2:end)];
end
```


Program 8 PairwiseLegRepulsion.m: activation of repulsion functions between leg pairs.

```
function xdot = PairwiseLegRepulsion(x,binarylist)
% PAIRWISELEGREPULSION activates repulsion between desired pairs
%
%   PAIRWISELEGREPULSION(X,BINARYLIST) uses a 15-vector of 0 and 1
%   values to selectively enable repulsion between leg pairs.  The
%   15-vector corresponds to the N choose 2 pairs for 6 legs.  X is
%   the set of phase offsets.
%
%   XDOT is returned, a negative gradient of the potential function
%   created with pairwise leg repulsion
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2007

N = length(x);
xdot = zeros(N,1);

% loop over all N choose 2 possible repulsions
for k=1:length(binarylist)
  if binarylist(k)
    pair = CutNumberToLegPair(N,k);
    i = pair(1);
    j = pair(2);

    % calculate force from repulsion function
    attr = cos(pi* (mod(x(i) - x(j), 1.0)));

    % apply it positive to first leg of pair
    xdot(i) = xdot(i) + attr;

    % apply it negative to second leg of pair
    xdot(j) = xdot(j) - attr;
  end
end
```

Program 9 RunSimulations.m: example run of gait regulation simulations.

```
function RunSimulations
% RUNSIMULATIONS simulates various gait regulation controllers
%
%   RUNSIMULATIONS is called without any arguments, initializing a
%   random value for the set of phase offsets of a six-legged system,
%   then calling gait regulation controllers for pentapod, tetrapod,
%   and tripod controllers.  Simulations are filtered based upon a
%   nominal duty factor, and plots are generated.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2008

% range of time for simulations
timestep = 0.05;
timerange = [0:timestep:4];

% initialize a random set of phase offsets
phaseoffsets = rand(6,1);

disp('Final phase offsets (normalized):')

xdata = runsimfunc(@PentapodCoord,phaseoffsets,timerange);
figure(1)
title('Pentapod Coordination')
plot(timerange,xdata)
xlabel('Time (s)')
ylabel('Phase')

disp('Pentapod')
mod(xdata(:,end)-xdata(1,end),1.0)'

xdata = runsimfunc(@TetrapodCoord,phaseoffsets,timerange);
figure(2)
title('Tetrapod Coordination')
plot(timerange,xdata)
xlabel('Time (s)')
ylabel('Phase')

disp('Tetrapod')
mod(xdata(:,end)-xdata(1,end),1.0)'

xdata = runsimfunc(@TripodCoord,phaseoffsets,timerange);
figure(3)
title('Tripod Coordination')
plot(timerange,xdata)
xlabel('Time (s)')
ylabel('Phase')

disp('Tripod')
mod(xdata(:,end)-xdata(1,end),1.0)'

function xdata = runsimfunc(funcname,initialx,timerange)
% RUNSIMFUNC runs the simulation with a given function

% initialize data, random initial phase offsets
xdata = zeros(6,length(timerange));
xdata(:,1) = initialx;

% for loop to run simulation
for i=2:length(timerange)
  % simple first order integrator
  funcgradient = funcname(xdata(:,i-1));
  funcgradient = checkphase(timerange(i),xdata(:,i-1),funcgradient);
  xdata(:,i) = xdata(:,i-1) + ...
      (timerange(i)-timerange(i-1)) * funcgradient;
end
xdata = mod(xdata,1.0);

function newxdot =  checkphase(phi,x,xdot)
% CHECKVELOCITY guarantees certain conditions on our phase velocity

% simulations run with a fixed duty factor of 5/6
dutyfactor = 5/6;

% find any velocities during leg stance, and set to 0
xbar = mod(phi - x,1.0);
xdot(find(xbar < dutyfactor)) = 0;

% clip the velocity so that legs don't go in reverse, or go more
% than double speed (to reflect actuator limitations).
xdot = min(xdot,1.0-1e-6);
xdot = max(xdot,-1.0+1e-6);

newxdot = xdot;
```


Program 10 RecursiveCellSearch.m: the switched hybrid gait regulation control algorithm.

```
function [cutsetcp,mincost] = RecursiveCellSearch(gaitnumber,cutset,cutdist,connectedness)
% RECURSIVECELLSEARCH performs a search over gait cells
%
%   RECURSIVECELLSEARCH(GAITNUMBER,CUTSET,CUTDIST,CONNECTEDNESS)
%   performs a recursive search through the graph of gait cells in
%   order to find a "minimum" distance route to a convergent region,
%   based upon the topology of the graph, the connectedness of the
%   cells, and the metric distances on the torus between leg pairs.
%
%   [CUTSET,MINCOST] are returned, the set of repulsion functions to
%   activate, as well as the minimum distance metric.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2008

% make a copy of the list of pairwise cuts
cutsetcp = cutset;

% foremost, if the current gait cell is 0-connected, just return.
% We don't have to modify anything to achieve convergence
if connectedness(gaitnumber) == 0
  mincost = 0.0;
  return
end

% calculate the neighboring gait cells, and which cuts separate the
% current cell from those.  This is a local computation of the gait
% cell graph representation.
neighborcells = zeros(1,6);
connections = zeros(1,6);
for i=1:6
  legorder = GaitNumberToLegOrder(6,gaitnumber);
  legpair = [legorder(i),legorder(mod(i,6)+1)];
  if i==6
    neworder = [legorder(6); legorder(2:5); legorder(1)];
  else
    neworder = [legorder(1:i-1); legorder(i+1); ...
                legorder(i); legorder(i+2:6)];
  end
  neighborcells(i) = LegOrderToGaitNumber(neworder);
  connections(i) = LegPairToCutNumber(6,legpair);
end

% initialize distance cost to high value
mincost = 100;

% look at graph to find cells with connectedness less than current
% cell.  compare distance to cell with overall minimum distance
for i=1:6
  g = neighborcells(i);
  c = connections(i);
  if connectedness(g) < connectedness(gaitnumber);
    % if this cell has lower connectedness, perform recursive step
    [newcutset,testcost] = ...
        RecursiveCellSearch(g,cutset,cutdist,connectedness);

    % compute distance to cell
    newcost = testcost + cutdist(c);

    % compare overall distances
    if newcost < mincost
      mincost = newcost;
      cutsetcp = newcutset;

      % if we follow this route, deactivate the boundary between cells
      cutsetcp(c) = 0;
    end
  end
end
```

Program 11 PentapodCoord.m: pentapedal gait regulation controller.

```
function xdot = PentapodCoord(x)
% PENTAPODCOORD coordinates all legs to repulse
%
%   PENTAPODCOORD(X) simply applies all pairwise leg repulsions, in order
%   to produce convergence at crawl gaits.  The code below is written
%   in generally for any number of legs, for which the pentapod case
%   is just one example.
%
%   XDOT is returned, a gradient vectory policy directing the
%   system to the nearest pentapod gait.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2006

% N is the number of legs
N = length(x);

% turn on all N(N-1)/2 pairwise repulsions, 15 different repulsions
% for a six legged system
xdot = PairwiseLegRepulsion(x,ones(N*(N-1)/2,1));
```

Program 12 TetrapodCoord.m: tetrapedal gait regulation controller.

```
function xdot = TetrapodCoord(x)
% TETRAPODCOORD performs switched repulsion control for tripod
%
%   TETRAPODCOORD(X) takes a set of phase offsets, and returns a
%   negative gradient of a potential function that will make the
%   system converge to one of four chosen tetrapod gaits.  This
%   function uses a hybrid switched approach that guarantees
%   convergence and avoids local minima, while only using leg
%   repulsion to avoid multiple legs recirculating together.
%
%   XDOT is returned, a gradient vector policy.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2008

% connectedness of tetrapod policy one, with the following grouping of
% legs: ((1,5),(2,6),(3,4))
connectedness1 = [1 2 2 3 1 2 1 2 2 3 1 2 1 2 1 2 1 1 0 1 0 1 2 2 2 ...
                  1 3 2 2 1 1 0 2 2 0 1 2 1 1 1 1 2 2 1 1 2 2 3 2 1 ...
                  3 2 2 1 1 0 2 2 0 1 2 1 1 1 1 2 2 1 1 2 2 3 2 1 2 ...
                  1 0 0 1 2 0 0 2 1 1 2 0 0 2 1 0 0 1 2 1 2 0 1 0 1 ...
                  2 2 1 2 2 1 3 2 1 2 2 1 3 2 1 1 2 1 2 1];

% related binary definition of repulsions to always activate
cutset1 = [1 1 1 0 1 1 1 1 0 0 1 1 1 1 1];

% connectedness of tetrapod policy two, with the following grouping of
% legs: ((1,6),(2,4),(3,5))
connectedness2 = [1 2 2 1 3 2 0 1 0 1 2 2 2 1 1 2 2 3 2 1 1 1 1 2 2 ...
                  1 1 2 2 3 2 1 1 2 2 3 0 1 0 1 2 2 1 2 1 2 1 1 0 1 ...
                  0 1 2 2 1 2 2 1 3 2 1 2 2 1 3 2 1 1 2 1 2 1 1 2 2 ...
                  1 3 2 0 1 0 1 2 2 2 1 1 2 2 3 2 1 1 1 1 2 1 2 0 0 ...
                  2 1 2 1 2 1 0 0 0 0 1 2 1 2 1 2 0 0 2 1];

% related binary definition of repulsions to always activate
cutset2 = [1 1 1 1 0 1 0 1 1 1 0 1 1 1 1];

% compute the phase differences for each pair of legs.  these are
% used as a metric to compare routes of convergence.
cutdist = zeros(1,15);
for i=1:15
  pair = CutNumberToLegPair(6,i);
  leg1 = pair(1); leg2 = pair(2);
  cutdist(i) = min(mod(x(leg1) - x(leg2), 1.0), ...
                   mod(x(leg2) - x(leg1), 1.0));
end

% convert phase into leg order and gait cell
legorder = LegPhaseToLegOrder(x);
gaitnumber = LegOrderToGaitNumber(legorder');

% if connectedness of policy 1 is closer, use it
if connectedness1(gaitnumber) < connectedness2(gaitnumber)
  [cutset,mincost] = RecursiveCellSearch(gaitnumber,cutset1, ...
                                         cutdist,connectedness1);
end

% likewise, if connectedness of policy 2 is closer, use it
if connectedness2(gaitnumber) < connectedness1(gaitnumber)
  [cutset,mincost] = RecursiveCellSearch(gaitnumber,cutset2, ...
                                         cutdist,connectedness2);
end


% if their connectednesses are equal, use mincost to determine closest
if connectedness1(gaitnumber) == connectedness2(gaitnumber)

  [cutset1out,mincost1] = RecursiveCellSearch(gaitnumber,cutset1, ...
                                              cutdist,connectedness1);
  [cutset2out,mincost2] = RecursiveCellSearch(gaitnumber,cutset2,...
                                              cutdist,connectedness2);

  % add in original cut distances of the tetrapod.  this is critical
  % to differentiate between when both strategies are 0-connected.
  mincost1 = mincost1 + sum(cutdist([4,9,10]));
  mincost2 = mincost2 + sum(cutdist([5,7,11]));

  % choose one with minimum distance (with preference on boundary
  % for policy #1)
  if mincost1 < mincost2
    cutset = cutset1out;
  else
    cutset = cutset2out;
  end
end

% use the binary cutset to activate the required repulsions
xdot = PairwiseLegRepulsion(x, cutset);
```

Program 13 TripodCoord.m: alternating tripod gait regulation controller.

```
function xdot = TripodCoord(x)
% TRIPODCOORD performs switched repulsion control for tripod
%
%   TRIPODCOORD(X) takes a set of phase offsets, and returns a
%   negative gradient of a potential function that will make the
%   system converge to the tripod gait.  This function uses a hybrid
%   switched approach that guarantees convergence and avoids local
%   minima, while only using leg repulsion to avoid multiple legs
%   recirculating together.
%
%   XDOT is returned, a gradient vector policy.
%
%   Author: Clark Haynes <gch@cs.cmu.edu> 2008

% this is the connectedness of the 120 cells to the alternating tripod
% gait's convergent region.  2-connectedness are gait cells that
% must pass through 2 cuts to converge.
connectedness = ...
    [2 1 1 1 1 2 1 1 1 1 0 0 1 1 2 1 2 1 1 1 0 0 1 1 1 0 1 1 ...
     0 1 1 0 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 2 1 ...
     1 1 1 2 2 1 1 1 1 2 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 ...
     1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 1 2 1 2 1 1 0 0 1 1 ...
     1 1 2 1 1 1 1 2];

% cutset is the default set of 15 binary repulsions that would
% normally (but only locally) converge to the alternating tripod gait
cutset = [1 0 1 0 1 ...
          1 0 1 0 ...
          1 0 1 ...
          1 0 ...
          1];

% compute the phase differences for each pair of legs.  these are
% used as a metric to compare routes of convergence.
cutdist = zeros(1,15);
for i=1:15
  pair = CutNumberToLegPair(6,i);
  leg1 = pair(1); leg2 = pair(2);
  cutdist(i) = min(mod(x(leg1) - x(leg2), 1.0), ...
                   mod(x(leg2) - x(leg1), 1.0));
end

% convert phase into leg order and gait cell
legorder = LegPhaseToLegOrder(x);
gaitnumber = LegOrderToGaitNumber(legorder);

% make a call to the recursive algorithm
cutset = RecursiveCellSearch(gaitnumber,cutset,cutdist,connectedness);

% use the binary cutset to activate the required repulsions
xdot = PairwiseLegRepulsion(x, cutset);
```

Bibliography

[1] Factoradic - wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/factoradic.

[2] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. B. Brown, D. McMordie, U. Saranli, R.J. Full, and D.E. Koditschek. Rhex: A biologically inspired hexapod runner. *Autonomous Robots*, 11:207, 2001.

[3] Alan T. Asbeck, Sangbae Kim, M. R. Cutkosky, William R. Provancher, and Michele Lanzetta. Scaling Hard Vertical Surfaces with Compliant Microspine Arrays. *International Journal of Robotics Research*, 25(12):1165–1179, 2006.

[4] Kellar Autumn, Martin Buehler, Mark Cutkosky, Ronald Fearing, Robert J. Full, Daniel Goldman, Richard Groff, William Provancher, Alfred A. Rizzi, Uluc Saranli, Aaron Saunders, and Daniel E. Koditschek. Robotics in scansorial environments. *Unmanned Ground Vehicle Technology VII*, 5804(1):291–302, 2005.

[5] G. Brambilla, J. Buchli, and A.J. Ijspeert. Adaptive four legged locomotion control based on nonlinear dynamical systems. In *From Animals to Animats 9. Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB'06)*, volume 4095 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.

[6] Timothy Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *International Journal of Robotics Research*, 25(4):317–342, 2006.

[7] R. A. Brooks. A Robot That Walks; Emergent Behaviors from a Carefully Evolved Network. Memo 1091, MIT AI Lab, February 1989.

[8] Rodney A. Brooks. From earwigs to humans. *Robotics and Autonomous Systems*, 20(2-4):291–304, 1997.

[9] J. Buchli, F. Iida, and A.J. Ijspeert. Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3903–3909. IEEE, 2006.

[10] J. G. Cham, S. A. Bailey, J. E. Clark, R. J. Full, and M. R. Cutkosky. Fast and robust: Hexapedal robots via shape deposition manufacturing. *International Journal of Robotics Research*, 21(10), 2002.

[11] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *In Proceedings of IROS'04*, September 2004.

[12] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica K Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, April 2005.

[13] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. The MIT Press, 2005.

[14] A. H. Cohen, S. Rossignol, and S. Grillner (eds.). Wiley Inter-Science, NY, 1988.

[15] Holk Cruse. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, 13:15 – 21, 1990.

[16] F. Delcomyn. Neural basis for rhythmic behaviour in animals. *Science*, 210:492–498, 1980.

[17] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[18] V. Durr. Stereotypical searching movements in the stick insect: kinematic analysis, behavioural context and simulation. *Journal of Experimental Biology*, 204:1589–1604, 2001.

[19] Volker Dürr, André Krause, Josef Schmitz, and Holk Cruse. Neuroethological concepts and their transfer to walking machines. *International Journal of Robotics Research*, 22(3-4):151–168, 2003.

[20] K. S. Espenschied, R. D. Quinn, H. J. Chiel, and R. D. Beer. Leg coordination mechanisms in stick insect applied to hexapod robot locomotion. *Adaptive Behavior*, 1(4):455 – 468, 1993.

[21] Edward Fadell. *Geometry and Topology of Configuration Spaces*. Springer, Berlin, 2001.

[22] Mattia Frasca, Paolo Arena, and Luigi Fortuna. *Bio-Inspired Emergent Control of Locomotion Systems*. World Scientific Publishing Company, 2004.

[23] G. Clark Haynes and Alfred A. Rizzi. Gait regulation and feedback on a robotic climbing hexapod. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[24] G. Clark Haynes and Alfred A. Rizzi. Gaits and gait transitions for legged robots. In *Proceedings of the IEEE International Conference On Robotics and Automation*, pages 1117–22, Orlando, FL, USA, May 2006.

[25] M. Hildebrand. Symmetrical gaits of horses. *Science*, 150:701–708, 1965.

[26] R. Hodoshima, T. Doi, Y. Fukuda, S. Hirose, T. Okamoto, and J. Mori. Development of titan xi: a quadruped walking robot to work on slopes. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[27] John M. Hollerbach, Ian W. Hunter, and John Ballantyne. A comparative analysis of actuator technologies for robotics. pages 299–342, 1992.

[28] E. Klavins and D.E. Koditschek. Phase regulation of decentralized cyclic robotic systems. *International Journal of Robotics Research*, 21(3), 2002.

[29] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *In Proceedings of the National Conference on Artificial Intelligence*, July 2004.

[30] T.M. Kubow and R.J. Full. The role of the mechanical system in control: A hypothesis of self-stabilization in hexapedal runners. *Philosophical Transactions: Biological Sciences*, 354(1385):849–861, 1999.

[31] V. R. Kumar and K. J. Waldron. Adaptive gait control for a walking robot, 1989.

[32] M. Lewis, A. Fagg, and G. Bekey. Genetic algorithms for gait synthesis in a hexapod robot, 1993.

[33] R. Merx, F. Prinz, K. Ramaswami, M. Terl, and L. Weiss. Shape deposition manufacturing. In *Solid Freeform Fabrication Symposium*, 1994.

[34] Eadweard Muybridge. *Animals in Motion*. Dover Publications, New York, 1899.

[35] K. Pearson. The control of walking. *Scientific American*, 235:72–86, 1973.

[36] Ioannis Poulakakis, James Andrew Smith, and Martin Buehler. Modeling and experiments of untethered quadrupedal running with a bounding gait: The scout ii robot. *International Journal of Robotics Research*, 24(4):239–256, 2005.

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

[38] Marc H. Raibert. *Legged Robots That Balance*. The MIT Press, 1986.

[39] Marc H. Raibert. Trotting, pacing, and bounding by a quadruped robot. *Journal of Biomechanics*, 1991.

[40] A. Rizzi, G. Clark Haynes, R. Full, and D. Koditschek. Gait generation and control in a climbing hexapod robot. In *SPIE Unmanned Systems Technology VII*, volume 6230, Orlando, FL, 2006.

[41] Uluc Saranli, Martin Buehler, and Daniel E. Koditschek. RHex: A Simple and Highly Mobile Hexapod Robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

[42] A. Saunders, D. Goldman, R. Full, and M. Buehler. The rise climbing robot: body and leg design. In *SPIE Unmanned Systems Technology VII*, volume 6230, Orlando, FL, 2006.

[43] Amir Shapiro, Elon Rimon, and Shraga Shoval. A foothold selection algorithm for spider robot locomotion in planar tunnel environments. *International Journal of Robotics Research*, 24(10):823–844, 2005.

[44] Shin-Min Song and Kenneth J. Waldron. *Machines That Walk: The Adaptive Suspension Vehicle*. The MIT Press, 1988.

[45] M. A. Spenko, G. C. Haynes, A. Saunders, A. A. Rizzi, M. Cutkosky, R. J. Full, and D. E. Koditschek. Biologically inspired climbing with a hexapedal robot, 2008. In press.

[46] Joel D. Weingarten, Richard E. Groff, and Daniel E. Koditschek. A framework for the coordination of legged robot gaits. In *IEEE Int. Conf. on Robotics, Automation and Mechatronics (RAM)*, volume 2, pages 679 – 686, 2004.

[47] Joel D. Weingarten, Gabriel A. D. Lopes, Martin Buehler, Richard E. Groff, and Daniel E. Koditschek. Automated gait adaptation for legged robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 3, pages 2153 – 2158, 2004.

[48] L. E. Weiss, R. Merz, F. Prinz, G. Neplotnik, P. Padmanabhan, L. Schultz, and K. Ramaswami. Shape deposition manufacturing of heterogenous structures. *Journal of Manufacturing Systems*, 16(4):239–248, 1997.

[49] David Wettergreen, H. Thomas, and Chuck Thorpe. Planning strategies for the ambler walking robot. In *IEEE Int. Conf. on Systems Engineering*, pages 198 – 203, August 1990.

[50] D. M. Wilson. Insect walking. *Annual Review of Entomology*, 11:103 – 122, 1966.

[51] Viktor Zykov, Josh Bongard, and Hod Lipson. Evolving dynamic gaits on a physical robot. In Maarten Keijzer, editor, *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, Seattle, Washington, USA, 26 July 2004.

