

MODELING AND CONTROL TECHNIQUES FOR A CLASS OF
MOBILE-ROBOT ERROR RECOVERY PROBLEMS

Ravi Balasubramanian

CMU-RI-TR-06-39

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213.

September 2006

Copyright © 2006 by Ravi Balasubramanian. All rights reserved.

© Copyright by Ravi Balasubramanian 2006
All Rights Reserved

Abstract

A robot’s locomotion mode fails when its environmental contacts fail, a situation called a *locomotion error*. For example, a legged robot cannot move when its leg becomes trapped in a crevice, and a wheeled robot is handicapped when its wheels skid. How can a robot recover when its standard locomotion mode fails? One way is to utilize any remaining freedoms to move the robot to a situation where the robot’s standard locomotion mode is again feasible. However, planning and control for such unconventional motion is difficult, since the relationship between the robot’s controls and its motion in a locomotion error is unclear. This complexity when combined with the uncertainty in environmental interaction (perceived as an “element of luck”) in a locomotion error appears as a lack of structure, inducing operators to sometimes use random escape maneuvers. This thesis proposes finding recovery strategies by exploiting the structure inherent to the robot’s constrained mobility and environmental interaction in the locomotion error. A robot equipped with multiple locomotion modes can choose between them depending on the circumstance, ultimately contributing to robust mobility.

While robotic locomotion fails in many ways depending on the robot’s design and the environmental interaction, this thesis finds novel recovery modes involving a combination of direct actuation and dynamically coupled actuation for two specific locomotion errors: first, a high-centered legged robot, where the robot’s legs dangle in air; and second, a car trapped in a slippery pit. In the high-centered robot problem, we present a novel locomotion mode called “legless locomotion”, that allows the robot to locomote by rocking its body back and forth using leg swing without feedback about the robot’s body motions. We use experiments and computer simulation to identify legless locomotion’s properties and use simple models to derive an approximate control technique. In the car-in-ditch problem, we use computer simulation to find a control strategy involving wheel torques and an active-suspension that allows the car to roll out of the pit, while minimizing work done and perturbations to the car body. Finally, we present a classification structure for locomotion errors based on environmental influence.

Acknowledgment

I first thank my advisors Matt Mason and Al Rizzi for guiding me all these years at Carnegie Mellon. They both have helped me grasp the bigger picture of research and taught me how to chisel a way to tackle hard problems. I am lucky to work with such incredibly smart and friendly people and learn many things, not just technical ideas but also research ethics. I look up to both of you as role models.

I thank my thesis committee members Francesco Bullo and Dimi Apostolopoulos. When I started out on my thesis research, Francesco spoon-fed me one of his papers and that really helped me understand the many concepts. Dimi has always been a source of support and has given me excellent feedback whenever I have approached him.

I thank Elie Shammas and Klaus Schmidt for a year of enlightenment understanding geometric control! It was fun digressing from covariant derivatives to music, culture, and international politics! I must thank Howie Choset, who has been a pillar of support for me all these years—you are super!

I thank Brendan Meeder, who helped build The Rocking and Rolling Robot and the tracking code and was instrumental in our ICRA video.

I thank my MLAB brothers Devin Balkcom, Siddhartha Srinivasa, and Ram Ravichandran for all the great times in the lab, and Jean for being so unbelievably efficient and helpful.

I thank Dan Koditschek, Uluc Saranli, and Hal Komsuoglu for a great time at Michigan. My thesis germinated during my time in Michigan. I must have done something right there!

I thank Garth Zeglin, Amir Shapiro, Dave Conner, Sarjoun Skaff, Rob Zlot, Chris Atkeson, George Kantor, Sanjiv Kumar, Anthony Gallagher, Bambi Roberts, Jake Sprouse, Curt Bererton, Marilena Vendittelli, Murali Talupur, and Pradeep Ravikumar for your insightful comments on my work and friendship.

I thank the Robotics Institute for funding my graduate study and Suzanne Muth for all the prompt assistance she has provided. My research was supported under NSF IIS 0082339, NSF IIS 0222875, and DARPA/ONR N00014-98-1-0747 contracts.

I thank Trevor Blackwell and Scott Wiley for hosting me at Anybots for a summer. I had a wonderful time working on your robots.

I thank my Pittsburgh music friends—Shankar and family, Lakshmi and family, Vikram, Jayashankar and family, Paddy and family, Sundararaman and family, Sujana and family, Visala and family, Ajit and family—for their love and friendship. I thank the Sri Venkateswara Temple and its community for giving me spiritual balance throughout my PhD. I thank my SPIC MACAY friends, Anand Rao, Srinivas Chellappa, Mangala Srinivas, Ranjani Rao, Deepa Ramachandran, and Raju Patil, for providing me with enough fun activities to get through my graduate study. I thank my roommate Mahesh Sabhnani and his parents for tolerating me all this time. Mahesh, your parathas are the best!

I finally thank my family—parents M. Balasubramanian and Shyamala Balasubramanian, brother Jayanth, grandmother Kamala Seetharaman, and sister-in-law Preetha—for all the unconditional love and support throughout my life. You are my inspiration and the reason for me to do anything.

Thank you everyone!

Contents

1	Introduction	1
1.1	Example Problem 1: Locomotion for High-centered Robots	4
1.2	Example Problem 2: A Car Stuck in a Ditch	10
1.3	Contributions	11
1.4	Thesis Road Map	15
1.5	Chapter Dependencies	15
1.6	Publication Note	15
2	Background	16
2.1	Locomotion Techniques	16
2.2	Mobile-Robot Error Recovery	19
2.2.1	Locomotion Error Classification	19
2.2.2	Locomotion Error Recovery	21
2.3	Dynamics Systems Modeling Techniques	21
2.4	Control for Dynamics Systems	25
2.5	Summary	30
3	Legless Locomotion: Models, Experiments, and Control	31
3.1	An Introduction to The Rocking and Rolling Robot	32
3.2	RRRobot Dynamics Model	37
3.3	Sphere-plane contact kinematics	38
3.4	Legless Locomotion Gaits	44
3.4.1	Demonstrating Legless Locomotion Using Experiments and Simulation . .	44
3.4.2	Exploring Legless Locomotion Capabilities Using Simulation	60
3.4.3	Summary of RRRobot Experiments and Simulations	65

3.5	Simplified Legless Locomotion Dynamics Models	65
3.5.1	Pivoting Dynamics Model	67
3.5.2	Single-Axis-Rotation Models	71
3.5.3	Summary	84
3.6	Toward Legless Locomotion Control	86
3.6.1	Legless locomotion control	87
3.6.2	Tracking Varying Curvature Paths	90
3.7	Summary	91
4	Dynamic Feedback Strategy for a Car in a Slippery Ditch	94
4.1	Modeling a Stuck Car in a Ditch	97
4.2	Intuition into the Normal Model's Dynamics and Controls	100
4.3	Gait Generation for the Stuck-Car Problem	102
4.3.1	A Hand-tuned Gait	102
4.3.2	Gait Search using Dynamic Programming	109
4.3.3	Comparing Algorithm 1 and the Dynamic Programming Gait	109
4.4	Summary	112
5	Conclusion	113
5.1	Contributions	113
5.2	Future Directions	114
5.2.1	Automatic Control Strategies for Robots	114
5.2.2	Robot Mobility and Design	115
5.2.3	Uncertainty in Robot-Environment Interaction	115
5.2.4	Alternate Rocking and Rolling Robot Designs	115
5.2.5	Kinematic Reductions for Mechanical Systems with Gravity	120
5.2.6	Stuck Car Problem: Experimental Validation	121
5.3	Closing Thoughts	121
	Bibliography	122

List of Tables

3.1	Rotation Time-Periods for the RRRobot-on-a-plane model and the Pivoting Dynamics model	71
3.2	Incremental motion of the yaw model.	81
4.1	Normal Car Body Model Parameters	99
4.2	Evaluation of algorithm 1 using the metric $\alpha\ \tau\ + \beta\ p\ + \gamma\ s - s_d\ $	104
4.3	Dynamic Programming Parameters	111

List of Figures

1.1	NASA rover Opportunity gets stuck on Mars.	2
1.2	The RHex experimental platform high-centered on a block (http://rhex.net).	5
1.3	Three types of legless locomotion: rolling, walking, and sliding. Rolling legless locomotion (see Fig. 1.5) and walking legless locomotion (see Fig. 1.4) ensue when body rotational oscillations are produced by leg motions. Sliding legless locomotion occurs when a net frictional force is produced over the cycle of planar body motions caused by leg motions. This thesis focusses on rolling legless locomotion.	7
1.4	Body roll and yaw rotations that produce translation. Motions are represented as rotations about axes attached to the body but aligned with the world coordinate frame. There is a local roll rotation between positions (a) and (b) and positions (c) and (d); there is a local yaw rotation between positions (b) and (c) and positions (d) and (e).	8
1.5	Walking legless locomotion: body oscillations can produce locomotion for a robot with body protrusions.	9
1.6	The RRRobot experimental platform uses halteres to induce body attitude oscillations leading to body translations.	9
1.7	Sandstorm, an autonomous humvee [3], overturned during tests.	10
1.8	Schematic diagram of a wheeled robot in a large ditch. The robot mass sits on an active suspension.	11
1.9	Robots that use varied forms of underactuated locomotion. Legless locomotion occupies a new domain in the space of underactuated locomotion.	13
2.1	Statically stable locomotion classification [81].	17
2.2	Dynamically stable locomotion classification.	18
2.3	The universal planar manipulator [62].	18

2.4	Locomotion error classification	20
2.5	A single degree of freedom hopping robot that self-orientes itself before propulsion (Courtesy: Caltech/JPL).	22
2.6	The snakeboard.	23
2.7	The Spring Loaded Inverted Pendulum model is used as a template for RHex's lo- comotion.	25
2.8	Legless locomotion generic model.	26
2.9	The control problem.	26
2.10	The RRRobot control problem.	26
2.11	The planar skater.	27
2.12	The Yaw model: the body, pivoted at its body center, can freely rotate about the yaw axis, and the two legs with point masses at the distal ends are actuated.	29
3.1	The RRRobot is a hemisphere with two short actuated legs [16].	33
3.2	Body roll-yaw rotations that produce locomotion. Motions are represented as rota- tions about axes attached to the body but aligned with the world coordinate frame. There is a local roll rotation between positions 1 and 2 and positions 3 and 4; there is a local yaw rotation between positions 2 and 3 and positions 4 and 5.	34
3.3	Body pitch-yaw rotations that produce locomotion. Motions are represented as ro- tations about axes attached to the body but aligned with the world coordinate frame. There is a local pitch rotation between positions 1 and 2 and positions 3 and 4; there is a local yaw rotation between positions 2 and 3 and positions 4 and 5.	35
3.4	The path taken by a sphere rolling on a plane changes with body attitude trajectory. The figure shows how the contact point evolves over one cycle for different pitch- yaw phase relationships, given by $\theta_r = 0$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$	39
3.5	Translation magnitude as a function of body yaw phase (for a given pitch oscillation trajectory and zero roll) for a sphere (radius 0.12 m).	40
3.6	Translation direction for a sphere (radius 0.12 m) on a plane after one cycle in body- rotation space given by $\theta_r = \sin(t + \beta_r)$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$	41
3.7	Translation magnitude for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = \sin(t + \beta_r)$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$	41
3.8	Translation direction for a sphere (radius 0.12 m) on a plane after one cycle in body- rotation space given by $\theta_r = A_r \sin(t + \pi/2)$, $\theta_p = \sin(t)$, and $\theta_y = A_y \sin(t + \pi/4)$	42

3.9	Translation magnitude for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = 0$, $\theta_p = \sin(t)$, and $\theta_y = a_y \sin(t + \pi/4)$	42
3.10	Contact-point time history for a sphere (radius 0.12 m) on a plane starting from the origin for the body-rotation phase relationship given by $\theta_r = \theta_p = 0.15 \sin(8t)$, and $\theta_y = 0.1t + 0.15 \sin(8t + \pi/2)$. The arrows indicate robot yaw orientation.	43
3.11	Planar plots of contact-point time history during sideways locomotion produced by Gait 1 in RRRobot-on-a-plane simulation and RRRobot-on-a-plane experiment. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.	47
3.12	Planar plots of contact-point time history during counter-clockwise circular locomotion produced by Gait 2 in RRRobot-on-a-plane simulation and RRRobot-on-a-plane experiment. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.	48
3.13	Studying gait transitions in legless locomotion: the offset of the sinusoidal gaits change from $\pi/2$ to $5\pi/8$ to $\pi/2$ and then to $3\pi/4$	50
3.14	Untethered RRRobot experiment: planar translation.	52
3.15	RRRobot simulation: planar translation.	52
3.16	Untethered RRRobot experiment: leg 1 trajectory. The dotted lines indicate leg offset position.	53
3.17	Untethered RRRobot experiment: leg 2 trajectory. The dotted lines indicate leg offset position.	53
3.18	Untethered RRRobot experiment: phase relationship between leg motions during each period.	54
3.19	Untethered RRRobot experiment: body pitch trajectory. The dotted lines indicate body pitch offset position.	55
3.20	RRRobot simulation: body pitch trajectory. The dotted lines indicate body pitch offset position.	56
3.21	Untethered RRRobot experiment: body yaw trajectory.	56
3.22	RRRobot simulation: body yaw trajectory.	57
3.23	Untethered RRRobot experiment: phase relationship between pitch and yaw rotations during period 1 ($t \in [37.5, 41.7]$ seconds).	57
3.24	RRRobot simulation: pitch-yaw phase relationship during period 1 ($t \in [20, 30]$ seconds).	58

3.25	Untethered RRRobot experiment: phase relationship between pitch and yaw rotations during period 4 ($t \in [208, 225]$ seconds).	58
3.26	RRRobot simulation: pitch-yaw phase relationship during period 4 ($t \in [180, 210]$ seconds).	59
3.27	Untethered RRRobot experiment: body roll oscillations.	59
3.28	RRRobot simulation: time history of contact-point motion induced by out-of-phase leg motions about the vertical configuration over thirty seconds.	61
3.29	RRRobot simulation: time history of contact-point motion induced by out-of-phase leg motions offset $\pi/4$ from the vertical configuration over thirty seconds.	62
3.30	Translation curvature as a function of leg trajectory offset and phase difference: each plot shows the time history of contact-point motion over one hundred seconds. The X-axis range is $[-0.6, 0.6]$ m, and the Y-axis range is $[0, 1.1]$ m.	63
3.31	RRRobot pitch rotation amplitude as a function of leg trajectory offset and phase difference.	64
3.32	RRRobot roll rotation amplitude as a function of leg trajectory offset and phase difference.	64
3.33	RRRobot yaw rotation amplitude as a function of leg trajectory offset and phase difference.	64
3.34	RRRobot pitch-yaw phase difference as a function of leg trajectory offset and phase difference.	65
3.35	The Pivoting Dynamics model simplifies the RRRobot model (see Figure 3.1) into two parts: (a) RRRobot pivoted at its geometric center on a spherical joint and (b) a sphere on a plane.	66
3.36	A planar eccentric-mass wheel performs harmonic oscillations for small amplitude.	70
3.37	The simple pendulum performs harmonic oscillations for small amplitude.	70
3.38	Planar plots of contact-point time history during sideways locomotion produced by Gait 1 in RRRobot-on-a-plane simulation, RRRobot-on-a-plane experiment, and Pivoting Dynamics simulation. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.	72
3.39	Planar plots of contact-point time history during counter-clockwise circular locomotion produced by Gait 2 in RRRobot-on-a-plane simulation, RRRobot-on-a-plane experiment, and Pivoting Dynamics simulation. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.	73

3.40	RRRobot's roll freedom (side view).	74
3.41	RRRobot's pitch freedom (side view).	74
3.42	RRRobot's yaw freedom (top view).	75
3.43	Decoupled RRRobot dynamics.	75
3.44	The lateral translation gait: comparison of RRRobot's motion with motion predicted by the single-axis models over thirty seconds. Leg 1 trajectory: $\pi/2 + 0.3 \sin(8t)$, and leg 2 trajectory: $\pi/2 + 0.3 \cos(8t)$	77
3.45	The circular translation gait: comparison of RRRobot's motion with motion predicted by the single-axis models over thirty seconds. Leg 1 trajectory: $\pi/4 + 0.3 \sin(8t)$, and leg 2 trajectory: $\pi/4 + 0.3 \cos(8t)$	77
3.46	The Pitch model: Variation in body pitch oscillation amplitude as a function of leg offset and leg phase difference.	79
3.47	The Yaw model: the body, pivoted at its body center, can freely rotate about the yaw axis, and the two legs with point masses at the distal ends are singly actuated. . . .	79
3.48	Lie bracket-inspired leg motions to change body yaw.	82
3.49	The Yaw Model: Variation in body yaw oscillation amplitude as a function of leg offset and leg phase difference.	84
3.50	Yaw model height function. Trajectory A (leg 1: $5\pi/8 + 0.15 + 0.3 \sin(8t)$ and leg 2: $5\pi/8 + 0.15 + 0.3 \cos(8t)$) produces net body yaw, while trajectory B (leg 1: $\pi/2 + 0.3 \sin(8t)$ and leg 2: $\pi/2 + 0.3 \cos(8t)$) does not produce net yaw.	85
3.51	RRRobot yaw-pitch phase difference as a function of leg trajectory offset and phase difference, as predicted by the single-axis models.	85
3.52	Similarity in planar translation between a vertical unicycle and RRRobot (top view).	87
3.53	RRRobot translation as a function of offset and leg phase difference.	88
3.54	RRRobot translation as a function of offset and leg phase difference as predicted using the decoupled models (compare with Fig. 3.53)	88
3.55	Amplitude modulation used in the decoupled Yaw model.	89
3.56	Mapping between RRRobot linear velocity and yaw velocity and leg offset and phase difference.	89
3.57	Mapping between RRRobot linear velocity and yaw velocity and leg offset and phase difference as predicted by the decoupled models.	90
3.58	RRRobot planar translation as a function of leg phase-difference β shown in Fig. 3.59. The leg trajectories take the form $0.3 \sin(8t) + \pi/4$ and $0.3 \sin(8t + \beta) + \pi/4$	92

3.59	Variable RRRobot leg-trajectory phase differences.	92
4.1	Schematic diagram of a wheeled robot in a large ditch. The robot mass sits on an active suspension.	95
4.2	Schematic diagrams of the Normal Car Body model. The wheel, propelled by wheel torques, rolls in the ditch, and the car body is supported by an active suspension. . .	95
4.3	Schematic diagram of a wheeled robot in a small ditch. The robot mass sits on an active suspension.	96
4.4	The vertical car body model.	96
4.5	Schematic diagram of the interaction of wheel motion and car body motion. The white-headed arrows indicate the direction of car body motion, and the black-headed arrows the traction force.	100
4.6	Algorithm 1 applied to the Normal Car Body model (ditch radius 6 m and maximum wheel torque magnitude 300 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.	105
4.7	Algorithm 1 applied to the Normal Car Body model (ditch radius 6 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.	106
4.8	Algorithm 1 applied to the Normal Car Body model (ditch radius 4 m and maximum wheel torque magnitude 300 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.	107
4.9	Algorithm 1 applied to the Normal Car Body model (ditch radius 4 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.	108

4.10	A comparison of a solution based on algorithm 1 and a solution derived from dynamic programming applied to the Normal Sprung Mass model (ditch radius 6 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and sprung mass position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.	110
5.1	Schematic diagram of an RRRobot with orthogonal leg rotational axes.	116
5.2	An eight-legged RRRobot	117
5.3	Legless locomotion generic model.	119

Chapter 1

Introduction

Every mobile robot has a standard locomotion mode, depending on the robot’s design and its interaction with the environment. But a robot’s locomotion fails when the expected robot-environment interaction fails. These situations, called *locomotion errors*, significantly impact robotic mobility, even if they are isolated instances of failure. For example, a wheeled robot locomotes by pushing off the ground using wheel torques, but slips when ground traction is poor. The key motivation for our work is: *How can we handle locomotion errors to improve the robustness of robotic mobility?*

Our strategy for locomotion error recovery is to find new locomotion modes that allow the robot to reach a situation where the standard locomotion mode is feasible. Multiple locomotion modes ultimately contribute to mobile-robot robustness and autonomy, since the robot can choose between locomotion modes depending on circumstances. Continuing the wheeled robot example, the robot can overcome slip-related locomotion errors using a controller that chooses wheel torques depending on ground traction conditions.

We believe that *mobile-robot error recovery is a domain rich in interesting new problems crucial to robotics*, and we focus on two questions: How can we model mobile robot errors? How can robots recover from locomotion errors? This thesis provides an analysis of two specific legged and wheeled locomotion failures, finds novel recovery modes, and designs gaits to improve robot robustness.

While robotic locomotion fails for many reasons, this thesis focuses on locomotion failures that result from changes in robot-environment interaction. Two prominent examples of robots failing due to environmental interference and unpredictability are: 1) Sandstorm [77], an autonomous Carnegie Mellon humvee developed by the Red Team [3], failed a mission when it became high-centered on a road-side berm; and 2) the Mars rover Opportunity’s wheels sunk into a sand-dune and required three weeks of tele-operation for recovery (see Fig. 1.1 and [31]). In the first case where Sandstorm



Figure 1.1: NASA rover Opportunity gets stuck on Mars.

becomes high-centered, Sandstorm has an undesired contact between the car body and the ground preventing locomotion, while in the second case where the rover becomes trapped in sand, the rover's wheels does not have sufficient ground traction and normal reaction forces.

This thesis develops novel recovery modes for two specific locomotion errors: 1) locomoting a high-centered legged robot; that is, a robot whose body rests on the ground, and its legs do not touch the ground; and 2) getting a wheeled robot out of a slippery pit. In the high-centered legged robot problem, we show that the robot can translate by rocking on its shell using leg swings; and in the stuck-car problem, we show that the robot can take advantage of the dynamic effect of an active suspension to escape. In both cases, the strategy is to move the robot to a configuration where the robot's standard locomotion mode is feasible.

But what are the principles behind control for error-recovery locomotion modes? We now compare control methods for conventional locomotion modes and error-recovery locomotion modes.

Control for Error Recovery Locomotion Modes

Control for error recovery can be complex, because error recovery modes are not as structured as "normal" locomotion. By structure, we refer to the strong relationship that typically exists between locomotion mode design and robot design. In particular, a robot's design is usually based on its anticipated locomotion mode, which also depends on the environment the robot operates in. This feedback between robot design and locomotion mode design permits a systematic analysis of the relationship between the robot's controls and its motion. Here is an example: a car is designed to travel on relatively flat ground using wheel torques, while relying on the wheel-ground traction.

This locomotion mode is straightforward to analyze because the car’s design is customized for its motion—spin the wheels and the car moves forward; rotate the steering, and the car turns. Thus, it is simple to understand how the car’s controls, namely wheel torques and steering, influence its motion.

In contrast, locomotion errors require atypical actuation resulting in motions that are sometimes difficult to anticipate during robot design. Continuing the car example, if the car is stuck on a berm, it needs to rock back and forth to free itself from the undesired contact between the body and the ground. In addition to the atypical wheel torques required for escape, the car wheels may also lose ground contact, thus complicating control. Clearly, a generic car is not designed for such maneuvers, and a human driver may struggle in such circumstances. In general, the pre-specified robot morphology makes finding and analyzing error-recovery modes difficult, and this difficulty appears as a lack of structure in error recovery problems. In future, a robot could be designed anticipating alternate locomotion modes that help in error recovery, but that can prove expensive and is beyond the scope of this thesis.

In the car example, we notice that the error recovery mode takes advantage of an interplay of the kinematics and the dynamics to produce net motion; that is, the car jerks back and forth by varying its wheel torques and the ground contact mode. As the car rocks on the berm, its body also pitches and rolls producing incremental translation. We call such locomotion produced through an interplay of dynamics and kinematics across the robot’s freedoms *dynamically coupled locomotion*. Two other examples of dynamically-coupled actuation are satellite reorientation using spinning reaction wheels (there are no environmental kinematic constraints, only the dynamic effect induced by angular momentum conservation) and a baby rolling over by rocking back and forth (there is an interplay between leg and arm swings and body rolling to produce locomotion).

The error-recovery locomotion modes studied in this thesis use a combination of direct actuation, such as torques to wheels with ground contact, and dynamically coupled actuation, such as robot internal shape changes that induce locomotion. In the high-centered legged robot problem, we show that the robot can control its translation curvature and velocity by varying its leg motions, and we show in the stuck-car problem that the robot can escape from a ditch by making use of an active suspension to control ground traction.

Error recovery modes and normal locomotion modes have differences in the type of robot-environment interaction also. Normal locomotion modes rely on a predictable robot-environment interaction. For example, the standard locomotion mode for wheeled robots assumes slip-free wheel-ground contact, and this allows us to compute the robot’s translation for a given wheel rotation. But

a locomotion error like wheel-ground slip makes motion prediction difficult, since it is difficult to know when exactly slip-free contact breaks. So locomotion modes during locomotion errors can be difficult to model as well as control. While an interesting problem, this thesis does not explicitly model such uncertainty and hybrid dynamics in robot-environment interaction.

The unpredictability in error-recovery locomotion modes appears as an element of "luck" or randomness and is a major hurdle in finding the right control strategy for recovery. For example, in our experiments with a high-centered robot, while we used different leg motions to induce escape, it was difficult to predict when the robot escapes. But a close analysis of the leg and body motions showed that our controls were exciting small body rotations to induce incremental translation. Can we find a low dimensional representation for these pseudo-random controls? Ideally, we want to exploit such structure to define a low dimensional gait space for locomotion errors. Finally, the motion produced by the error recovery modes discussed in this thesis is configuration-dependent, further complicating analysis.

Error-recovery locomotion modes cannot replace conventional locomotion, because they are customized for locomotion errors. Furthermore, error recovery modes may also be inefficient, because the locomotion error typically induces a handicap, such as disabling an actuated freedom. *While standard locomotion modes are easier to model and are designed for generic mobility, understanding dynamically coupled locomotion techniques is crucial to exploring alternate locomotion modes and improving robotic mobility.* Understanding dynamically coupled locomotion may also contribute towards better mechanical designs, since we explore hidden capabilities.

The next two sections introduce the two case studies explored in this thesis: 1) locomotion for high-centered robots and 2) a car escaping from a ditch. The solutions we provide for both problems have one common trait: they both use dynamically coupled locomotion modes that exploit internal mass distribution changes to induce motion.

1.1 Example Problem 1: Locomotion for High-centered Robots

Legged robots offer mobility in diverse terrain, but conventional legged locomotion fails when the robot is stuck. By "stuck", we refer to instances where the robot can no longer use its standard locomotion technique because of external circumstances. One such instance is when a robot is high-centered, that is, when its legs have no environmental contact (see Fig. 1.2). Since a legged robot requires contact between each leg and the surface for locomotion, the robot is stuck when it is high-centered.

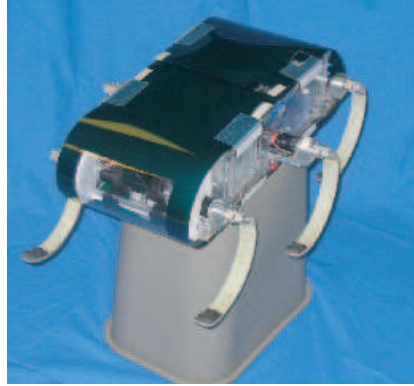


Figure 1.2: The RHex experimental platform high-centered on a block (<http://rhex.net>).

Our interest in locomotion for high-centered robots arose from experience with RHex [68], a simple and highly mobile hexapod robot (see Fig. 1.2). We investigated how RHex gets stuck in rugged terrain through several structured experiments [4]. The environment included various arrangements of cinder-blocks and styrofoam, and we focused on how RHex’s body or legs get trapped. It turned out that RHex was able to escape in most circumstances by sheer strength, except when RHex is high-centered.

A high-centered RHex has few options—it can only swing its legs and rock the robot body back and forth. Also, creating large body rotations is difficult with RHex’s light legs, flat bottom, and heavy body. So producing sufficient dynamic effect through leg motions to cause the robot to flip is not an option. Alternately, the small body rotations that the leg swings produce may induce incremental translation. Such incremental translation can be used to move the robot until, say, it falls off the block. But RHex’s flat body limits such incremental translation, and any translation depends on the friction profile along the body surface and the ground. Finally, since the choice of leg motions to produce maximal translation is unclear, RHex motion when it is high-centered seems “random”.

Suppose, instead of a flat bottom, RHex has a rolling contact between its body and the obstacle, because the robot’s stomach or the obstacle is rounded. Then, even small torques induce the robot’s attitude to oscillate and produce translation until the robot, say, falls off the block. We call this incremental locomotion mode *legless locomotion*.

We project legless locomotion as a technique for escaping when a robot is high-centered; even though the locomotion is slow and inefficient, the strategy is to translate while high-centered and reach a situation where the robot can use its legs in a conventional sense. In RHex’s case, it has

no other option to recover from a high-centered state. In general though, a high-centered legged robot has at least three strategies to translate incrementally by rocking and rolling the body using leg motions (see Fig. 1.3):

1. If the robot has a curved bottom, then the body rotations can incrementally translate the robot assuming slip-free body-ground contact (rolling legless locomotion, see Fig. 1.4).
2. If the robot has an irregular bottom and the protrusions act as ‘feet’, a series of rolling and yawing motions translates the robot by shifting its weight from one ‘foot’ to another (walking legless locomotion, see Fig. 1.5).
3. If the robot has a flat bottom, a net translation force can be produced using jerky leg motions (sliding legless locomotion, similar to the Universal Planar Manipulator concept [62]).

This thesis focuses on the first strategy that uses pure rolling between curved surfaces, because it offers a viable locomotion mode for a robot with inertial properties like RHex. For example, the second strategy requires leg swings to produce sufficient dynamic forces on the body to lift the robot’s weight onto the pivot points, and this is difficult for a robot like RHex with light legs and a heavy body. Also, there are impacts each time the robot’s weight transfers between feet. These impacts produce slip-related translation which is difficult to model analytically. On the other hand, the third strategy requires modeling differences in frictional forces over leg motion cycles, a difficult proposition. In contrast, with a rounded body rolling on a flat surface, even small torques causes the robot’s attitude to oscillate, and these oscillations can generate translation when coupled with the slip-free contact constraints between the robot body and ground surface. Also, since all interactions of the body with the environment are smooth, there are no discontinuities to model. Thus, locomoting a round-bodied robot on its stomach is effective and also easier to model than a robot with a jagged or flat bottom. This thesis focuses on the first strategy, rolling legless locomotion, henceforth referred to just as legless locomotion.

To simplify experimental study of legless locomotion, we need a structured system that allows us to identify legless locomotion’s key elements. This is because RHex is built for legged locomotion and is thus not suited for studying legless locomotion!

We have constructed a simple prototype robot, *The Rocking and Rolling Robot (RRRobot)* (see Fig. 1.6) to study legless locomotion. RRRobot is an unconventional biped: it has legs, but the legs never touch the ground! RRRobot locomotes by rocking on its spherical stomach. Since its legs act only as reaction masses, they are better called *halteres*, after the dumbbells sometimes used by athletes to give impetus in leaping.

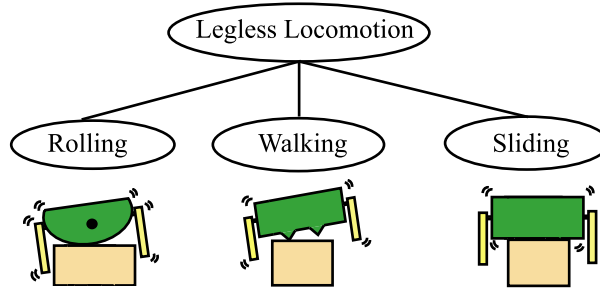


Figure 1.3: Three types of legless locomotion: rolling, walking, and sliding. Rolling legless locomotion (see Fig. 1.5) and walking legless locomotion (see Fig. 1.4) ensue when body rotational oscillations are produced by leg motions. Sliding legless locomotion occurs when a net frictional force is produced over the cycle of planar body motions caused by leg motions. This thesis focusses on rolling legless locomotion.

Considering just RRRobot’s body rotations and sphere-plane contact kinematics while ignoring the dynamics of how the rotations are produced, Fig. 1.4 shows a sequence of interleaved roll and yaw body rotations that induces translation. A challenge in legless locomotion is to find the leg motions that create body attitude oscillations which, when coupled with the nonholonomic contact constraints, cause RRRobot to locomote in the plane. While random leg motions may induce incremental translation, we ideally want a low dimensional gait space for legless locomotion.

Our RRRobot experiments and simulations suggest that body-attitude oscillations produced by leg motions are a practical way of translating in a high-centered state. Furthermore, legless locomotion has many interesting aspects that will offer an understanding of dynamically coupled locomotion, in particular the interaction between the configuration-dependent inertia, the oscillatory dynamics, and the contact kinematics. In this thesis, we study legless locomotion by answering the following questions:

- How can we model legless locomotion? What are the key parameters?
- How is legless locomotion related to other locomotion techniques?
- How can we control legless locomotion?

Chapter 3 presents answers to these questions and suggests interesting future work. We now review the other problem studied in this thesis: a novel locomotion strategy for a car stuck in a ditch.

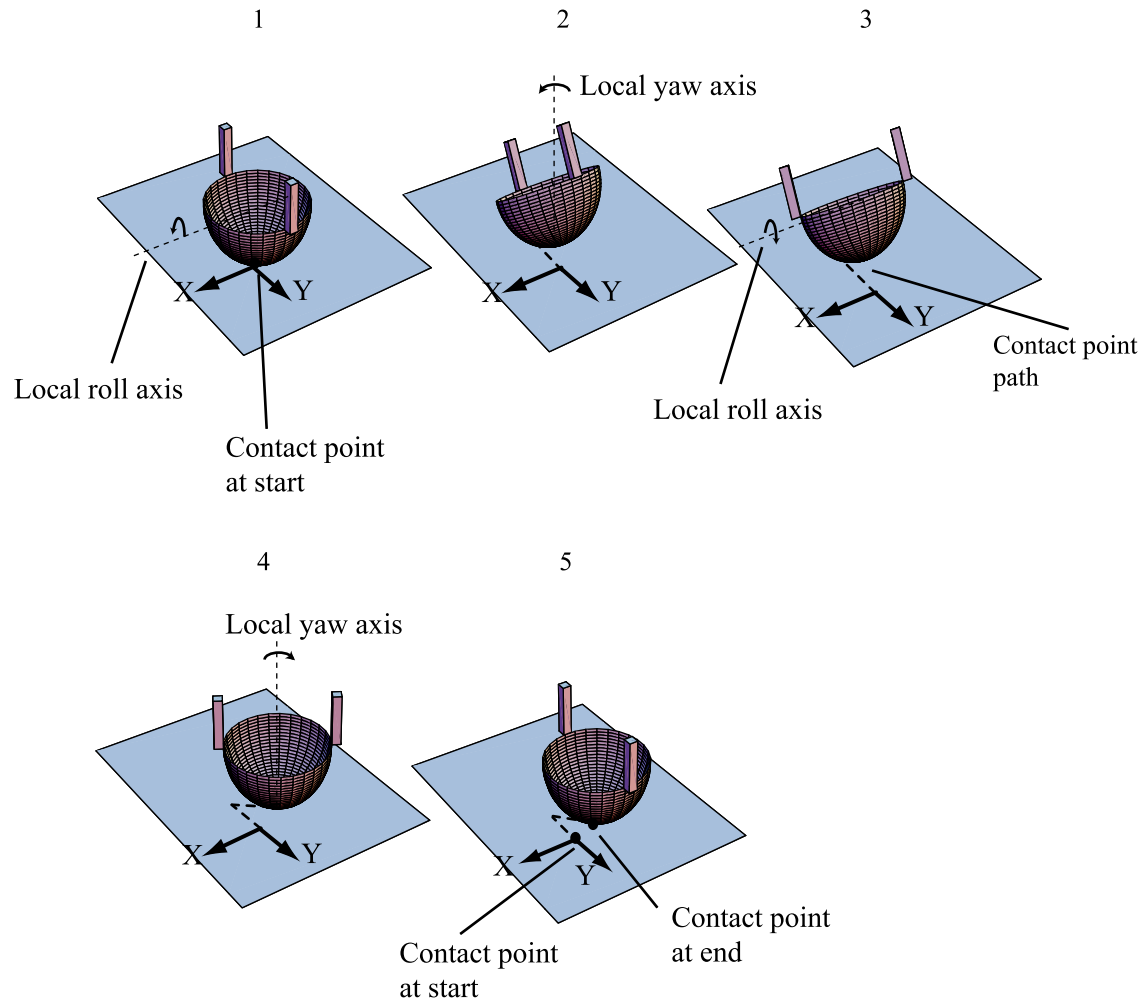


Figure 1.4: Body roll and yaw rotations that produce translation. Motions are represented as rotations about axes attached to the body but aligned with the world coordinate frame. There is a local roll rotation between positions (a) and (b) and positions (c) and (d); there is a local yaw rotation between positions (b) and (c) and positions (d) and (e).

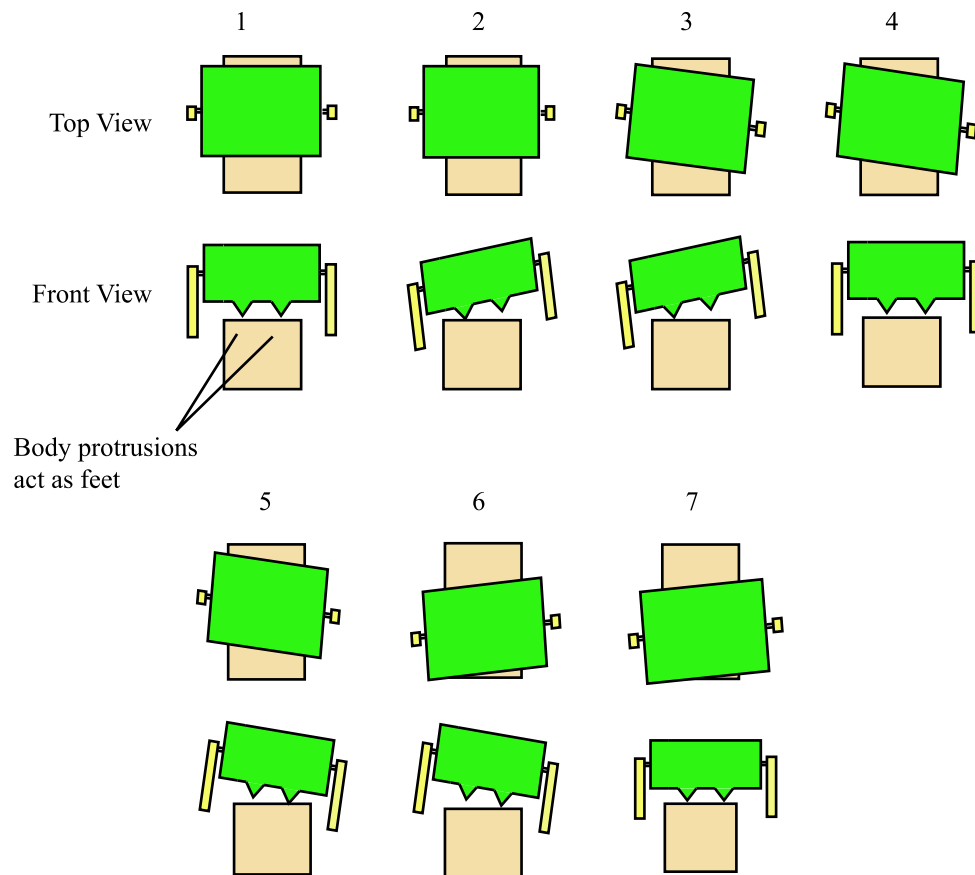


Figure 1.5: Walking legless locomotion: body oscillations can produce locomotion for a robot with body protrusions.

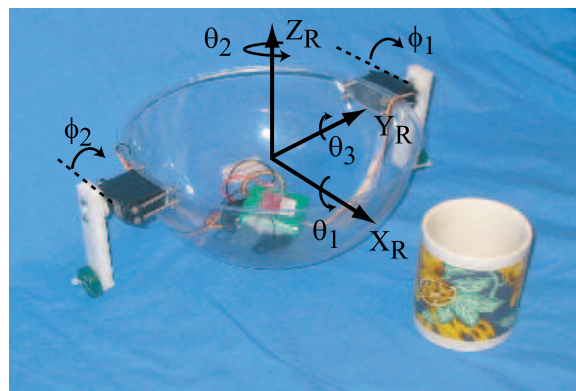


Figure 1.6: The RRRobot experimental platform uses halteres to induce body attitude oscillations leading to body translations.



Figure 1.7: Sandstorm, an autonomous humvee [3], overturned during tests.

1.2 Example Problem 2: A Car Stuck in a Ditch

Wheeled locomotion fails in many ways, such as wheel slip due to poor traction and vehicle overturning (see Fig. 1.7). This section introduces the problem of a wheeled robot stuck in a ditch (see Fig. 1.8), where the robot’s conventional locomotion strategy, namely driving straight out, fails. We propose a control strategy that combines wheel torques with unconventional use of an active suspension.

One approach for escape is to rock the robot back and forth in the ditch building up sufficient momentum to roll out, as though “bouncing” the car out of the ditch. This may work if the robot has sufficient power, and wheel-ground slip is not an issue. But suppose we want the robot to escape without wheel-ground slip, because it is energy conserving as well as easier to model. This restrictive condition becomes particularly acute with the limited traction along the steep slopes, and wheel-torque control may alone be insufficient. In such a situation, can an active suspension help?

As the robot rolls back and forth, the robot’s mass oscillates on the suspension, causing changes to the ground reaction forces. Can we use an active suspension to exert forces on the car mass, changing the ground reaction forces to satisfy the problem constraints? One naive strategy is to keep pushing the robot mass upwards to increase the ground normal reaction force and, hence, the traction limits. But this is impractical because the robot mass’s motion is limited. Thus, while we can *push* the robot mass upwards to increase available traction, we also need to *pull* the robot mass downwards to satisfy its geometric constraints when sufficient traction is available. This idea can be likened to a motorcyclist shifting his weight up and down to control wheel-ground interaction.

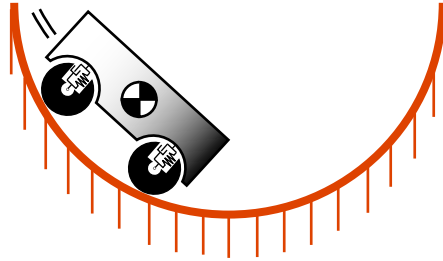


Figure 1.8: Schematic diagram of a wheeled robot in a large ditch. The robot mass sits on an active suspension.

This thesis hypothesizes that unconventional use of components like an active suspension can play a crucial role in mobile-robot error-recovery problems. Furthermore, we show that there exists a structure in this specific locomotion error that can be exploited to find gaits that allow a car trapped in a ditch to escape. Using simplified simulation models, we present an algorithm for coordinated use of wheel torques and active suspension to escape, while minimizing sprung mass oscillations, and also explore solutions using dynamic programming.

We explore the following questions in chapter 4:

- How can we find a locomotion behavior using wheel torques and active-suspension forces to induce the car to escape from the ditch?
- How do the system parameters influence the solution?

Even though we only provide a locomotion behavior for a simplified model using simulation, we expect that the locomotion strategy and the structural analysis we provide is applicable to the practical problem of a wheeled robot stuck in a ditch. We now summarize this thesis's contributions.

1.3 Contributions

This thesis makes three contributions:

1. Finding structure in mobile-robot error recovery.
2. Discovering and understanding legless locomotion.
3. Designing gaits for locomotion error recovery.

We now discuss each of the contributions.

1. *Finding structure in mobile-robot error recovery.*

When mobile robots fail, there is usually little analysis of the circumstances of failure and the controls that the robot has to recover. This is because locomotion errors are difficult to quantify and recovery modes difficult to analyze. We use two examples, the high-centered robot problem and the stuck-car problem, to show that mobile-robot error recovery problems do have structure. By structure, we refer to the low-dimensional gait spaces we have found for the error recovery modes we focus on. For example, to free a high-centered robot, we initially tried different leg motions to induce body rotations and incremental translation. These seemingly random leg-motion behaviors guided us toward thinking about specific leg motions that take advantage of the robot-ground contact and the robot's dynamics. We believe that many locomotion errors offer structured low-dimensional locomotion modes, and glean- ing novel recovery modes from "random" escape attempts requires a careful analysis of the robot's motion in the error circumstances.

We also show that the dynamics and geometry of locomotion errors can be studied using first principles to find alternate control techniques for recovery. In particular, we find new locomotion modes that utilize internal configuration changes and ground contact to create dynamic effect. A key principle we propose is that having a set of such novel locomotion modes ultimately contributes to mobile-robot autonomy, since the robot can choose between different locomotion modes depending on the situation. However, our approach does not model the uncertainty in robot-environment interaction. Future work involves finding a common structure in mobile robot errors using contact analysis and finding new forms of dynamically coupled locomotion.

2. *Legless locomotion: a novel locomotion technique.*

Our study of how mobile robots get stuck led to a high-centered-robot locomotion mode called legless locomotion. Legless locomotion is a result of the interaction between ground contact and body rotations produced by carefully chosen leg motions. Legless locomotion lies in a new space of underactuated locomotion (see Fig. 1.9), because of the combination of its properties: a variable inertia, body-environment contact constraints, and oscillatory motion.

We constructed a simple prototype robot called The Rocking and Rolling Robot to explore the key elements of legless locomotion. We provide a gait for legless locomotion that allows planar translation with variable curvature and velocity by varying the leg trajectories. We

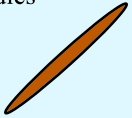

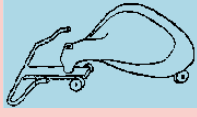

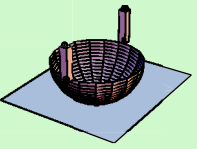
	No contact constraints	With Contact constraints
Constant inertia	Floating rigid bodies Submarines 	Snakeboard  Roller racer 
Configuration-dependent inertia	Floating articulated systems Satellites 	Rocking and Rolling Robot 

Figure 1.9: Robots that use varied forms of underactuated locomotion. Legless locomotion occupies a new domain in the space of underactuated locomotion.

provide results from experiment with RHex [4] and RRRobot and dynamic simulations of RRRobot, and the results match qualitatively.

Understanding legless locomotion using simplified models: Legless locomotion is complex because of its dynamically coupled actuation, namely the complex interaction of two forced oscillators (the body pitch and roll dynamics produced by leg motions), the yaw rotations about the contact point, and the nonholonomic contact constraints. Understanding the interplay of dynamics and contact kinematics, say, between robot locomotion speed or direction and the inputs is difficult.

We provide an analysis of legless locomotion using simplified decoupled models that analyze the elements separately. For example, we quantify the individual contributions of the sphere-plane contact (namely, the relationship between oscillatory body rotations and planar translations) and the relationship between leg motions and body roll, pitch, and yaw rotations using independent models. We go further by understanding the influence of leg motions on each of the body's rotational freedoms. We finally combine these simplified models (body rotation dynamics and contact kinematics) to define the relationship between RRRobot translation and leg inputs for gait design.

3. *Error-recovery Gait Generation.* We solve the gait generation problem for the two unconventional locomotion modes considered in this thesis using dynamic systems theory. The gaits

we find are low-dimensional behaviors inspired by the seemingly random control modes that we find initially.

We define legless-locomotion gait generation as the mapping from planar translation to leg motions; that is, given a desired translation direction, we find the leg motions that track it. Note that we ignore tracking the body rotations specifically and focus only on tracking average contact-point path. Finding such a mapping is challenging because legless locomotion is dynamic, oscillatory, and has variable inertia and nonholonomic constraints. Furthermore, RRRobot's dynamics is dependent on the body and leg configurations and is underactuated. We approach the control problem using decoupled models to provide a mapping between RRRobot motion curvature and leg motions, while showing that the decoupled models provide a good approximation to the full dynamics model.

In the stuck-car problem, we find control techniques that use wheel torques and an active suspension to induce escape while satisfying environmental constraints. We develop solutions for varying problem parameters using our intuition into the system dynamics and dynamic programming. We measure the solution quality with an objective function involving work done and sprung-mass oscillations. Finding metrics for locomotion error-recovery problems is still an open problem though.

We now summarize the similarities and the differences in the two control problems. Both locomotion modes are applicable in error-recovery and allow the robot to reach a situation where the standard locomotion is applicable again. Furthermore, they use internal mass distribution changes to induce motion. In both problems, the locomotion error is the absence of desired contact; in the high-centered robot problem, the robot's legs do not touch the ground, while in the stuck-car problem, the issue is poor wheel-ground traction.

But the two problems have their differences. In the high-centered robot problem, the controls in the legless locomotion problem are used primarily to produce locomotion without focusing on using the controls to satisfy environment constraints. In the stuck-car problem, the controls are used to produce motion as well as satisfy constraints. Also, we choose different approaches to explore the two error-recovery gait spaces, since there is a difference in problem complexity. Since RRRobot's state space is \mathbb{R}^{14} , we use Lagrangian dynamics and decoupled dynamics models. Since the stuck-car problem's state space is \mathbb{R}^4 , we use numerical methods in addition to a Lagrangian analysis.

We now provide the thesis road map.

1.4 Thesis Road Map

After reviewing related work in Chapter 2, we then present legless locomotion in Chapter 3. We discuss our approach to exploring the stuck-car problem in Chapter 4, followed by a discussion of the thesis's contributions and research extensions in Chapter 5.

1.5 Chapter Dependencies

- Chapters 1 and 2 are self-contained.
- Chapters 3 and 4 depends on chapter 1.
- Chapter 5 requires chapters 3 and 4.

1.6 Publication Note

Most of Chapter 3 appears in [17], [16], and [13], while parts of Chapter 1 appears in [14].

Chapter 2

Background

We now review background and related work, organized in four sections.

- Locomotion techniques
- Locomotion error and recovery
- Techniques for modeling mechanical systems
- Control techniques for dynamics systems

2.1 Locomotion Techniques

Locomotion, defined by the Merriam-Webster dictionary as the act or power of moving from place to place (*locus + motion*), is an important problem in robotics. While there are many forms of locomotion, including locomotion modes for land, water, and air, this thesis focuses only on ground locomotion.

Two primary forms of ground locomotion are legged locomotion and wheeled locomotion. Researchers have studied various forms of legged locomotion for the mobility it offers [60, 68, 82, 54, 58, 37, 66, 18, 51]. Similarly, researchers have explored various forms of wheeled locomotion for the efficiency it offers in structured terrain [45, 43, 52, 61, 19, 74, 38, 59]. Recently, undulatory or snake-like locomotion has become popular in the robotics community for the stability and traction it offers [32]. Many interesting aspects of undulatory locomotion have been explored by using the snakeboard [48, 59] and roller racer [44] as examples.

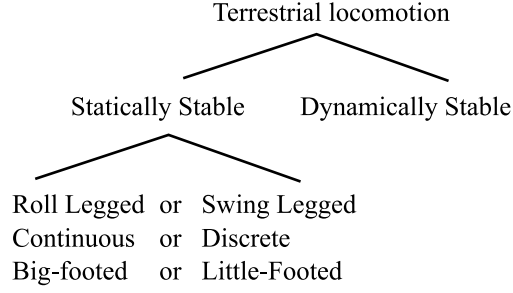


Figure 2.1: Statically stable locomotion classification [81].

Yim [81] provides a classification structure for statically-stable locomotion (see Fig. 2.1), while we provide a classification structure for dynamically stable locomotion using three hierarchical levels (see Fig. 2.2):

- Level 1 discusses the nature of dynamic stability, whether the system is asymptotically stable, neutrally stable, or unstable [64]. A stable system, after a perturbation, returns to a stable limit cycle or an equilibrium point after a perturbation. An asymptotically stable system returns to the original limit cycle or an equilibrium point, while a neutrally stable system returns to a new nearby limit cycle or equilibrium point.
- Level 2 discusses the nature of contact between robot and environment, depending on whether there is slip between bodies. For example, the Universal Planar Manipulator [62] requires slip between objects and the table surface on which the parts are moved around, while legged locomotion requires no slip between the round body and the surface.
- Level 3 discusses the locomotion mode's dynamics. The locomotion dynamics depends on whether there is a flight phase, in which case the dynamics is hybrid. Also, the dynamics depends on the origin of forces used to propel the robot. For example, in walking, reaction forces from direct contact between the ground and a leg induces propulsion, while in satellite reorientation, reaction forces from swinging reaction masses induces body rotation (dynamically coupled locomotion).

Legless locomotion, the locomotion mode we propose for high-centered robots, has properties that differentiate it from conventional locomotion modes. Legless locomotion is dynamic, asymptotically stable (in the presence of small disturbances and external damping, RRRobot behaves as an inverted pendulum), and requires pure rolling contact between its body and the ground. In addition, legless locomotion has a configuration-dependent inertia, lacks a flight phase, and is subject to

Level 1	Nature of Dynamic Stability	
	Asymptotic Stability	or Neutral Stability or Unstable
<hr/>		
Level 2	Nature of Contact	
	Rolling	or Sliding
	Point Contact (Multiple?)	or Surface Contact
<hr/>		
Level 3	Nature of Dynamics	
	Flight Phase (SLIP model)	or No Flight Phase
	Direct actuation	or Dynamically coupled actuation

Figure 2.2: Dynamically stable locomotion classification.

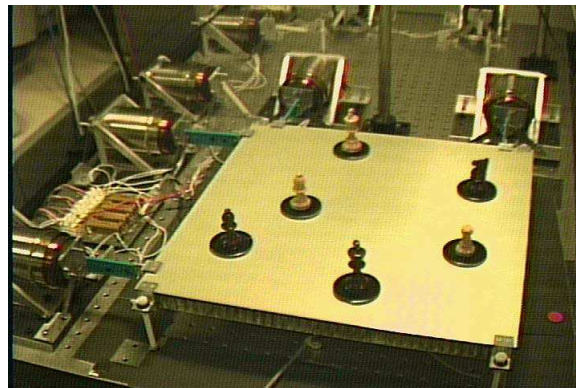


Figure 2.3: The universal planar manipulator [62].

gravitational drift. Legless locomotion’s interesting features resulting from the interplay of contact constraints and body rotational dynamics are explored in chapter 3. We now present related work in analyzing locomotion errors and recovery modes.

2.2 Mobile-Robot Error Recovery

Improving the robustness of robotic locomotion requires an understanding of how robotic locomotion fails. But a generic analysis of these problems is difficult, since how a mobile robot fails and how it recovers depends directly on the specifics of its design, the environment, and an element of luck. Carlson and Murphy [42] present a survey of how unmanned ground vehicles fail, using information from various urban search-and-rescue operations. While stating that mobile-robot reliability was low, they list effector and the control system as two primary failure causes.

Most prior work focuses on error diagnosis using high-level reasoning techniques on sensor data [79] and not on how robots get physically stuck. In this section, we first provide a classification of mobile-robot errors and then review some novel recovery modes.

2.2.1 Locomotion Error Classification

A robot’s locomotion mode depends on the nature of ground contact and the robot’s dynamics and can include many types of locomotion gaits (actuation patterns). For example, running and trotting are gaits that have the following common characteristics: neutral or asymptotic stability, point contact, direct actuation, and a flight phase. But the robot’s leg-ground contact during each leg cycle in running is different from those in trotting. Thus, they belong to different locomotion modes. Similarly, a walking gait for a multi-legged robot with a leg disabled and a walking gait with all legs functioning belong to different locomotion modes.

We define a locomotion error as an undesired external event that forces a change from one locomotion mode to another mode. For example, an animal purposely changing from the trot mode to the running mode is not a locomotion-error, but a multi-legged robot breaking a leg constitutes an error, since this forces the robot to use a different locomotion mode.

Using these notions of a locomotion mode and a locomotion error, we now provide a classification structure for locomotion errors (see Fig. 2.4):

- Level 1 categorizes errors based on whether the cause of the failure is in software or hardware or whether the error is due to the robot’s circumstances.

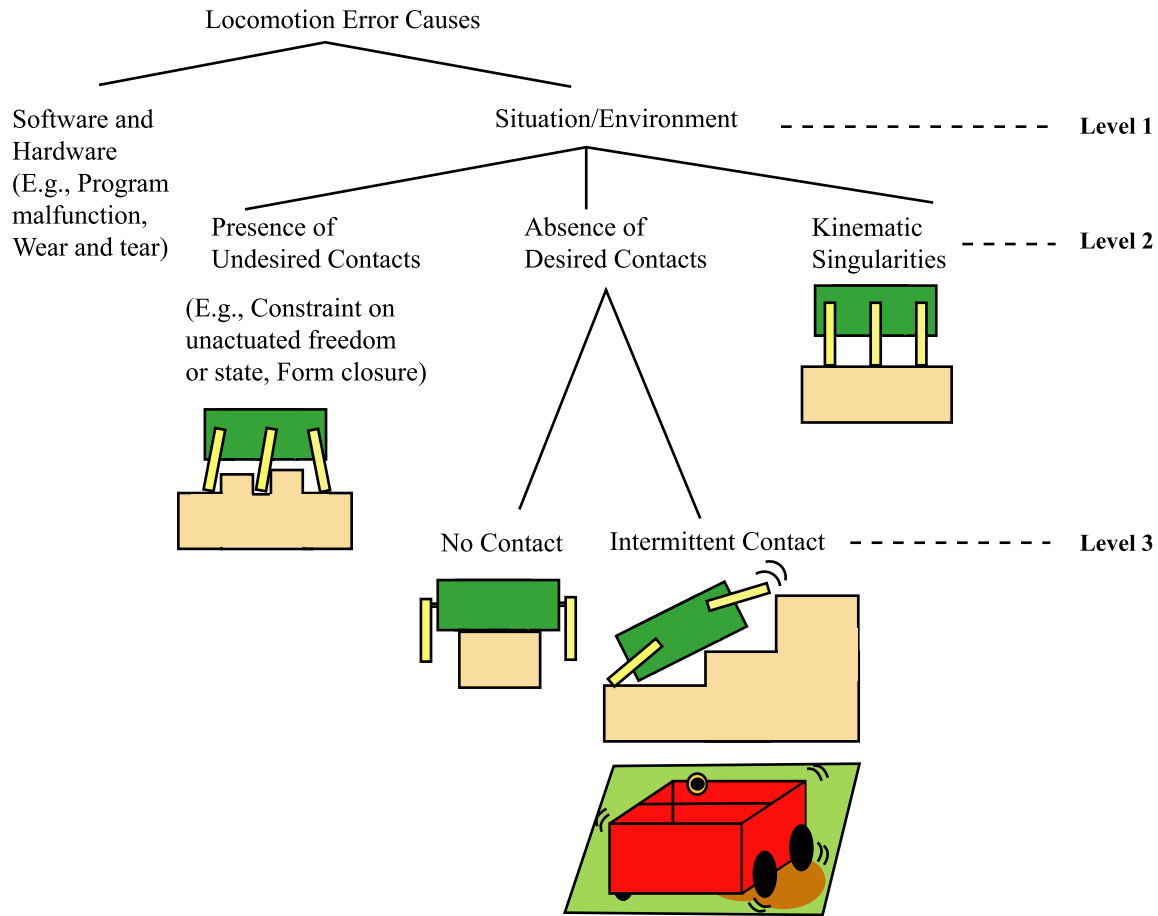


Figure 2.4: Locomotion error classification

- Level 2: Within situational errors, the failure may be due to the presence of an undesired contact, the absence of a desired contact, or the presence of a kinematic singularity.
- Level 3: Absence of a desired contact can occur in two ways: either there is no contact at all (as in the high-centered case), or there is intermittent contact. Intermittent contact can occur in a variety of ways; for example, slipping on a partially greased floor, or partially high-centered robot that has leg-ground contact only at certain body configurations.

This thesis focuses on locomotion errors due to situations that result in the absence of a desired contact, such as high-centered robots (no contact) and cars with poor traction. We now briefly review literature in the area of mobile-robot error recovery.

2.2.2 Locomotion Error Recovery

There has been some work in finding robot maneuvers that could act as error recovery techniques, although they are not portrayed as error recovery techniques. Hale et al. [40] present a singly actuated hopping robot that self-oriens itself before propulsion (see Fig. 2.5). Once the robot lands, it reorients itself in the required direction of motion. Tunstel [76] discusses a genetic programming approach to finding uprighting maneuvers for a nanorover. The algorithm evaluates the maneuver quality using the power consumed, the time elapsed, and the percentage of progress made.

While recovery modes allow a robot to return to its stock locomotion mode, the design of some robots allow some locomotion modes to work even after an error. For example, researchers at Carnegie Mellon have developed a highly maneuverable robot called Spinner [1] that has an actuated suspension and can operate even when inverted. RHex can walk upside-down also, but uprighting is useful when RHex uses automatic vision to navigate. Saranli [69] presents back-flips as a technique for ‘uprighting’ RHex (see Fig. 1.2).

Fujiwara et al. [36] explore safe falling techniques for humanoid robots. A safe falling technique is important because a humanoid robot’s center of gravity is typically high, and sensitive objects like cameras and motors can be damaged during a fall. The key idea they use is to minimize the robot’s vertical velocity, and hence the impact force, by taking advantage of the angular momentum conservation principle. This is achieved by first shortening leg length to minimize the gravitational moment on the body and then extending the leg to minimize angular velocity. This ultimately minimizes

In contrast, four legged robots like the AIBO [5] are not significantly damaged when they fall down while walking, since their center of gravity is low. In fact, they are used in robotic soccer games [78], and one of the “kicking” strategies is to hit the ball with the head, by splaying the front legs and falling forward. After falling the robots recover using a stand-up maneuver.

An interesting open problem is understanding the severity of a locomotion error and deciding on whether to use a recovery technique. The robot may also have to choose the appropriate recovery mode to use based on circumstances.

2.3 Dynamics Systems Modeling Techniques

Finding the dynamic and kinematic models that truly represent a robot’s motion is essential for developing the right control techniques. We now give a glimpse of the range of modeling techniques that have been used for different mobile robots. We show through our work that such structured



Figure 2.5: A single degree of freedom hopping robot that self-oriens itself before propulsion (Courtesy: Caltech/JPL).

modeling techniques help understand complex error-recovery problems also.

Legless locomotion, the primary locomotion mode this thesis explores, involves the interplay between body roll-pitch-yaw attitude dynamics and kinematic nonholonomic contact constraints. Numerous investigators have studied dynamic systems with constraints using Lagrangian dynamics [30] or the energy-momentum method. Lewis et al. [48] study the constrained mechanics of the constant-inertia *snakeboard* (see Fig. 2.6), a modified version of a skateboard in which wheel directions can be controlled. The snakeboard rider locomotes by twisting his/her body back and forth, while simultaneously moving the wheel-directions with a suitable phase relationship. Lewis et al. present numerical simulations of snakeboard locomotion using characteristic wheel motions and discuss a framework for studying mechanical systems with constraints in a coordinate-free form. Zenkov et al. [83] discuss the energy-momentum method for control of dynamic systems with nonholonomic constraints such as the rattleback, the roller racer, and the rolling disk. After identifying system symmetries, Zenkov et al. use momentum equations to analyze the system. We use the Lagrangian method to study RRRobot's dynamics.

While RRRobot's body spatial position and orientation (called the *fiber space* [22]) are not directly controlled, its leg configuration (called the *base space*) is controllable. Ostrowski [59] presents a general framework for studying systems where the fiber space can be represented as a group. Since only the base space is actuated, Ostrowski finds a *connection* relating the base space velocities to the fiber velocities. While Ostrowski focusses on systems with constant inertias, simple constraints, and no gravitational drift, such as the snakeboard and the Hirose snake, RRRobot has a



Figure 2.6: The snakeboard.

spherical contact with the plane, a configuration-dependent inertia, and gravitational drift. Thus, it is unclear if Ostrowski’s framework can be extended to encompass RRRobot behavior.

RRRobot locomotes by rolling its round body on the planar surface. The curvatures of the two surfaces and the type of contact between the two surfaces determine the kinematic constraints and, consequently, the relative motion between the two bodies. Montana [55] derives the equations of motion for the contact point between two moving rigid bodies using differential geometry. Camiciia et al. [26] provide an analysis of the nonholonomic kinematics and dynamics of the *Sphericle* [21], a hollow ball driven on a planar surface by an unicycle placed inside. Bhattacharya and Agrawal [20] present a spherical rolling robot that locomotes using two orthogonal rotors placed inside. They derive driftless equations of motion using the conservation of angular momentum and the contact constraints. The *Sphericle*, Bhattacharya’s robot, and RRRobot have similar nonholonomic contact constraints, but RRRobot’s oscillatory body rotations differentiates its planar motion (see [57] and [50] for more details on nonholonomic constraints).

If we ignore RRRobot’s ground contact constraints and assume RRRobot is floating in space, the problem of controlling RRRobot’s body motion (position and orientation) reduces to simply controlling its body attitude. Fernandes et al. [35] discuss near-optimal nonholonomic motion planning for coupled bodies using Lagrangian dynamics and the principle of angular momentum conservation. Modeling a falling cat as two links attached either through an actuated universal joint or an actuated spherical joint, Fernandes et al. find plans to land a falling cat on its feet from an arbitrary starting point. RRRobot’s body attitude control is complex because both the hip joints are aligned. This results in large reaction forces on the body pitch freedom, while the reaction forces are small on the body roll and yaw freedoms. Also, the roll and yaw body motions are induced by the nonlinear effects that arise from a configuration-dependent inertia. It can be shown that by repeatedly wiggling the legs while exploiting the inertia variances, RRRobot can adjust its orientation. This contrasts with satellite reorientation using spinning reaction wheels—the inertias of the satellite system do not change with rotation of reaction wheels [65], and hence non-aligned reaction wheels are necessary for complete control.

To isolate the contributions of leg motions and body attitude changes to RRRobot’s translation, we approximate the RRRobot-on-a-plane model by decoupling the body attitude dynamics from the contact kinematics (see section 3.5.1 for more details). We also simplify RRRobot’s motion by decoupling the robot’s non-actuated freedoms. For example, we study the influence of the leg motions on the robot’s pitch, roll, and yaw freedoms separately.

This approximate approach of splitting the dynamics from the kinematics is different from the

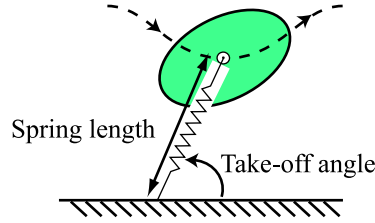


Figure 2.7: The Spring Loaded Inverted Pendulum model is used as a template for RHex’s locomotion.

exact kinematic reduction technique of Bullo et al. [23] for mechanical systems with constraints. In general, it is advantageous to find kinematic reductions, since a kinematically reduced system uses velocity inputs and is easier to control than a dynamic system with force inputs. But it is unclear if we can find a kinematic reduction for RRRobot (see section 2.4 for a brief treatment of kinematic reduction techniques).

Sometimes it is useful to view dynamic systems at a more abstract level by, say, viewing multiple links as a single link, since this eliminates the complexity of joints and serves as a guide for control. For example, Full and Koditschek [63] use models called templates derived by eliminating or combining joints and actuators to understand bipedal locomotion. Similarly, Altendorfer et al. [10] find evidence for the Spring Loaded Inverted Pendulum template (see Fig. 2.7) in RHex’s motion, and Schmitt and Holmes [70] propose the Lateral Leg Spring template to study insect locomotion in the horizontal plane. A legless-locomotion template could possibly be a one-legged model with multiple degrees of freedom (see Fig. 2.8). See section 5.2 for more detail on how the multi-degree of freedom leg can induce body roll, pitch, and yaw rotations that produce translation.

2.4 Control for Dynamics Systems

Control in robotics may be defined as finding a mapping from the desired robot motion to the available inputs (see Fig. 2.9). In general, control is easier if there is an input to control each of the robot’s freedoms (fully actuated) and if the dynamics does not depend on configuration. But many robots are not fully actuated, and the robot’s motion depends on configuration, making control difficult. Furthermore, this thesis explores control strategies for locomotion errors that require a combination of dynamically coupled actuation and direct actuation. We now review related work in two areas: 1) Planning and control for dynamic systems and 2) Dynamics approximations and simplifications.

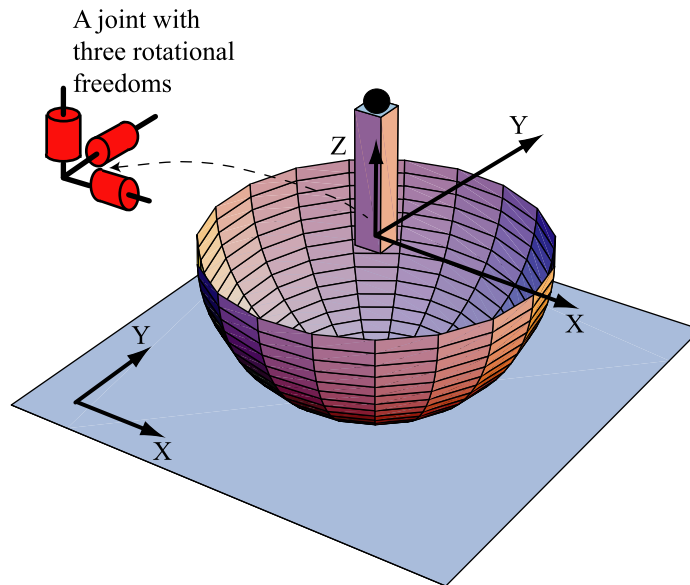


Figure 2.8: Legless locomotion generic model.

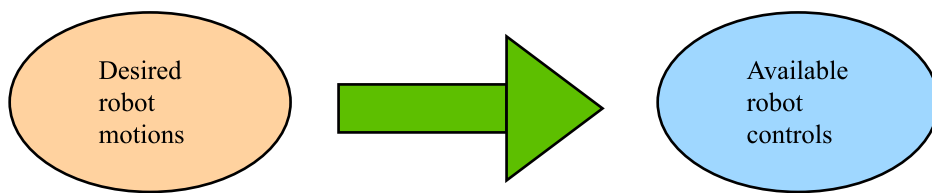


Figure 2.9: The control problem.

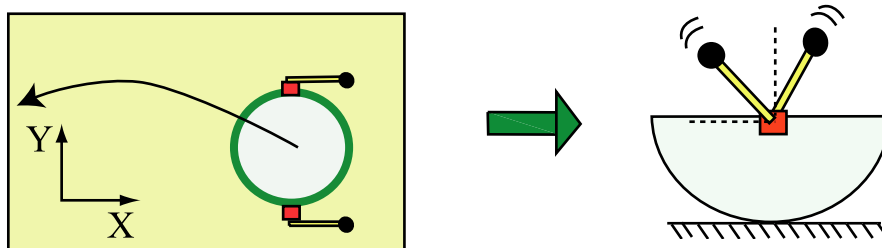


Figure 2.10: The RRRobot control problem.

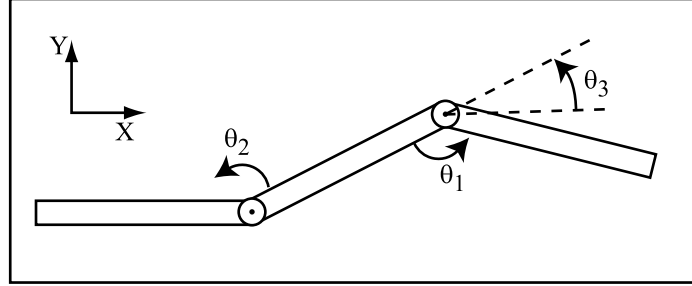


Figure 2.11: The planar skater.

Planning and Control Techniques for Dynamic Systems

Planning techniques are available for underactuated systems with gravitational drift and no nonholonomic contact constraints, and underactuated systems with nonholonomic contact constraints and no gravitational drift. With reference to systems with no nonholonomic contact constraints, Walsh et al. [39] present a planning strategy for a constant-inertia satellite in space with a varying number of controlled rotors. Walsh and Sastry [80] extend this approach to a multi-linked planar skater, whose inertia is configuration-dependent (see Fig. 2.11). Fernandes et al. [35] provide a near-optimal planning technique for a falling cat, and Papadopoulos and Dubowsky [33] state that nearly any planning technique used for fixed-base manipulators can be used for planning free-floating space manipulators. The key idea in these approaches is to use the angular-momentum conservation principle to find a kinematic representation of the dynamic system.

We now look at planning for systems with nonholonomic contact constraints and constant inertia, like the *Snakeboard* [48], and systems with non-holonomic contact constraints and a configuration-dependent inertia, like the *Trikke* [71]. As discussed before in section 2.3, the snakeboard is a variation of the skateboard with controllable wheel directions and a rotor. The *Trikke* consists of two rigid links connected by a revolute joint, and each link has a passive wheel at the distal ends. Both the snakeboard and the *Trikke* locomote through the interplay of stored rotor momentum and varying wheel directions. Bullo and Lewis [34] provide planning primitives for the *Snakeboard* by finding a kinematic representation, while Shamma et al. [71] provide a natural gait generation strategy using height functions for the snakeboard and the *Trikke*. We note that *RRRobot* is in a new space of dynamic underactuated systems with configuration-dependent inertia, contact constraints, and gravitational drift. Currently, it is unclear if we can exploit kinematic reductions and height functions for *RRRobot* control.

We now review some of the control techniques relevant to the stuck-car problem (see section 1.2

for an introduction).

The problem of getting robots to escape from a ditch is similar to the well-studied car-hill problem [75], where the goal is to get a car over a hill by controlling the wheel torques. Many reinforcement learning solutions that build momentum through cyclic application of inputs are available [75][11]. The problem we explore is more complex because of an expanded state and input space—we can change the car’s mass distribution using the active suspension in addition to controlling the robot’s velocity using wheel torques. The expanded state-input space opens the possibility for the controller to utilize “gaits” that synchronize wheel torques and active suspension forces. Also we provide intuition into important aspects of the system dynamics such as available traction and Coriolis effects. In chapter 4, we develop an approximate time-optimal dynamic-programming [46] solution, but one could also imagine using randomized search techniques [47] to find similar strategies. Such numerical techniques help establish the existence of approximate solutions.

Our stuck-car work is also related to Spong’s research on swing-up control of an underactuated revolute-revolute arm [73], where the goal is to swing the arm from the statically stable vertical-down configuration to the inverted position by “pumping in” energy using feedback linearization. Representing the wheel rolling in the ditch by a revolute arm and the mass motion on the active suspension by a prismatic joint, our problem is similar to the swing-up of a fully-actuated revolute-prismatic arm with dynamic constraints represented by the traction limits.

The automotive industry has extensively studied the use of active suspension to improve ride comfort; most research assumes that the car mass is supported vertically on a spring-damper system and develops various control methods to minimize vehicle vibration [29, 28]. Alleyne [9] proposes using an active suspension to minimize the stopping distance on flat ground by pushing down on the wheels to extract greater traction limits. This paper contrasts Alleyne’s work by considering non-flat ground, which induces different dynamics due to Coriolis and centrifugal effects.

System Dynamics Approximations

RRRobot’s dynamically coupled locomotion mode is complex, because of the interplay between body rotational dynamics and the nonholonomic contact constraints. We decouple RRRobot’s dynamics into a set of simplified models that decouple the robot’s motion into its individual freedoms. Another technique for simplifying the dynamics include linearizing the control system about an operating point. Even though an approximation, linearization provides insights into many control systems. Several researchers have linearized a control system’s equations about nominal trajectories to get insights into a robot’s motion [56, 41]. For example, Laumond [45] gives various linearization

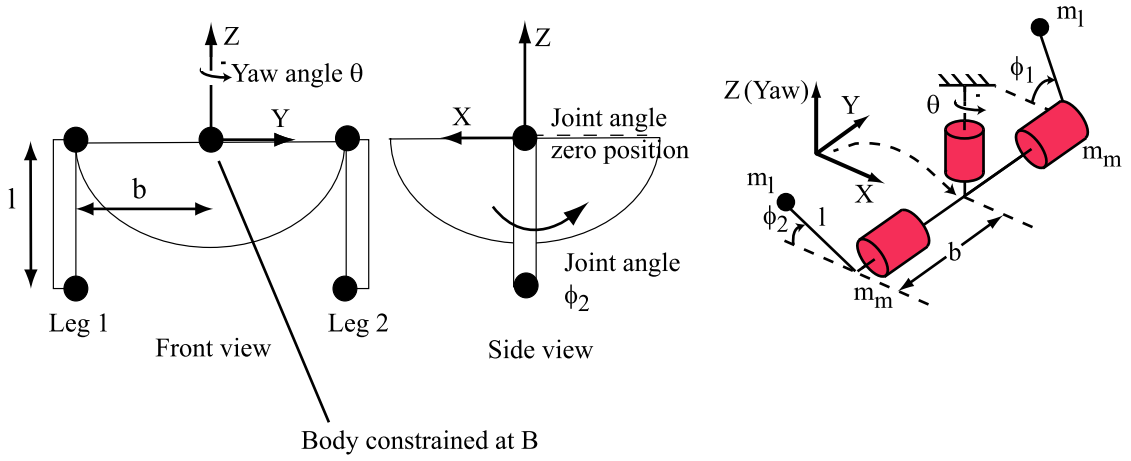


Figure 2.12: The Yaw model: the body, pivoted at its body center, can freely rotate about the yaw axis, and the two legs with point masses at the distal ends are actuated.

techniques to control a nonholonomic car-like robot. RRRobot's dynamics does not lend itself to linearization. While RRRobot's pitch and roll dynamics can be modeled as linear systems, RRRobot's yaw dynamics is inherently non-linear. The non-linearities arising from RRRobot's configuration-dependent inertia are essential to produce net body yaw.

Controlling a dynamic system such as RRRobot is difficult because it has gravitational and velocity-related drift and control inputs that are forces. In contrast, kinematic systems control is easier since the control inputs are velocities and there is no drift. So it is useful to check if a dynamic system can be *reduced* to a kinematic system.

Typically, it is not straightforward to find a kinematic model that completely represents a mechanical system. For a generic mechanical system to admit a kinematic reduction, the system must satisfy certain properties, as expressed in [23]. Bullo and Lynch [24] provide a direct algorithm for finding kinematic reductions by enforcing the condition that the kinematic system must satisfy the mechanical system's dynamic constraints at arbitrary time scaling.

It is unclear if a kinematic reduction exists for RRRobot, but we have found a kinematic reduction for a simplified model, called the Yaw model (see Fig. 3.47), derived by decoupling RRRobot's body rotations [12]. An important property of the Yaw model that helps us derive its kinematic reduction is that the yaw model's Lagrangian is invariant to yaw rotations, that is, there is a *symmetry*. This allows us to plan kinematically for the body's yaw rotation using velocity inputs for leg motions. These kinematic plans can then be converted to dynamic plans. Chapter 3 has more information on the Yaw model and its kinematic reduction.

2.5 Summary

Our work is at the intersection of three areas: 1) dynamic systems modeling, 2) dynamic systems control and planning, and 3) mobile robotics. This chapter touches some of the related work, while maintaining a balance between algorithmic techniques for mechanical systems and experimental mobile-robotics research. We also emphasize that the problems we explore, namely the high-centered robot problem and the stuck-car problem, have not been explored before and offer new insights into control for mechanical systems.

We now analyze legless locomotion in Chapter 3.

Chapter 3

Legless Locomotion: Models, Experiments, and Control

In this chapter, we present a novel locomotion strategy called legless locomotion for high-centered legged robots. We term a legged robot is high-centered when its legs do not touch the ground; this is a locomotion error since a legged robot requires slip-free leg-ground contact. We desire a locomotion strategy that allows us to reach the robot to a situation where it can resume its stock locomotion mode.

Our work is motivated by our experience with RHex, a hexapod robot. When RHex becomes high-centered on, say, a cinderblock, its legs become ineffective (because they cannot produce motion by pushing off the ground). Our initial exploration for escape modes for a high-centered RHex included any leg motion that excites body rotations and incremental translation. The strategy was to use such incremental translation until RHex falls off the block, after which RHex's stock locomotion mode can resume. While random leg swings may induce a high-centered robot to translate, we desire a structured gait that allows us to control a high-centered robot's motion and guide it toward escape.

By carefully analyzing the dynamics and kinematics of high-centered robots, we show in this chapter that there exists a low dimensional gait structure in the space of leg swings that induce a high-centered robot to translate. We call this structured locomotion mode "legless locomotion". Simply put, legless locomotion is a method of translation for high-centered robots using small body rotations induced by leg swings.

To simplify our analysis of legless locomotion, we have constructed a simple robot called The Rocking and Rolling Robot (RRRobot), a round-bodied legged robot that is always high-centered.

We use a round-bodied robot instead of a flat-bottomed robot like RHex, since a rounded bottom permits larger body rotations with a rolling contact than a flat bottom; the larger body rotations can lead to larger translation. Also, even though RRRobot's leg swings may allow the robot to completely tumble over or lose contact, we restrict our analysis to leg motions that induce small body attitude oscillations about a stable equilibrium. This is motivated by RHex's morphology—RHex's legs are too light to induce its heavy body to tumble by swinging its legs when high-centered. Section 1.1 and our ICRA 2004 movie [15] also provide an overview of our motivation for legless locomotion.

We demonstrate legless locomotion through experiments with RHex and RRRobot and simulations and models of RRRobot. We show that out-of-phase leg motions create body attitude oscillations which, when coupled with the slip-free contact constraints, locomote the robot in the plane. Furthermore, we show that by varying the leg trajectories, we can control the robot's planar motion.

We present, in section 3.1, The Rocking and Rolling Robot, the prototype high-centered robot we use to study legless locomotion. In section 3.2, we present a model of RRRobot's dynamics, followed by an analysis of RRRobot's sphere-plane contact kinematics in section 3.3. In section 3.4, we present, using experiments and simulation, legless locomotion gaits—leg motions that induce RRRobot to translate—followed by an exploration of the range of motions available using sinusoidal leg trajectories. In section 3.5, we present a set of simplified models that provide insight into legless locomotion, and show in section 3.6 that the simplified models can be used to develop control for legless-locomotion.

3.1 An Introduction to The Rocking and Rolling Robot

The Rocking and Rolling Robot (*RRRobot*) is a hemispherical shell with two short actuated legs (see Fig. 3.1). RRRobot is always high-centered since its legs never touch the ground, and is thus an unconventional bipedal robot. Note that RRRobot's morphology is similar to a two-legged round-bodied high-centered RHex.

RRRobot's light legs have reaction masses at their distal ends similar to halteres, the dumbbells sometimes used by athletes to give impetus in leaping. The battery and processor are attached to the hemisphere bottom, and we assume the sphere rolls on the plane. RRRobot's mass distribution is designed so that it behaves as an inverted pendulum, and its body attitude oscillates about its stable equilibrium damped by ground friction. The goal is to locomote RRRobot in the plane in its high-centered state through oscillatory body rotations induced by leg swings.

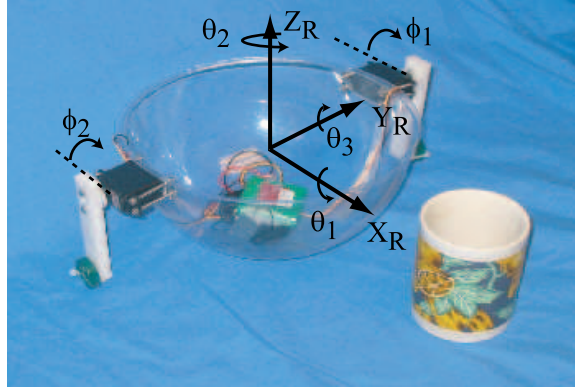


Figure 3.1: The RRRobot is a hemisphere with two short actuated legs [16].

RRRobot's motion structure permits a clean split of the body-ground contact kinematics and the leg-body rotational dynamics for separate analysis. We exploit this division repeatedly to understand RRRobot's complex motion. We first briefly present how the kinematics and dynamics contribute to RRRobot's motion, before exploring them deeply in subsequent sections of this chapter.

A First Look at RRRobot's motion

Since we restrict our analysis to leg motions that induce RRRobot to behave like an inverted pendulum, we focus on body attitude oscillations which when coupled with the nonholonomic contact constraints induce its rounded body to translate in the plane. For example, Fig. 3.2 shows a sequence of interleaved roll-yaw body rotations that cause a sphere to translate along the X-axis. Similarly, Fig. 3.3 shows a sequence of interleaved pitch-yaw body rotations that causes a sphere to translate along the Y-axis. We will see in section 3.3 that the robot translates even if these interleaved body rotations are replaced with smooth out-of-phase body oscillations with the correct phase relationship. Since RRRobot's body is spherical, we restrict our analysis to spherical bodies only; a similar analysis can be performed for any smooth body-ground contact.

Note that body yaw oscillations are a crucial common element of both body roll-yaw and pitch-yaw oscillations that induce a sphere to locomote. Furthermore, if the sequence of rotations in Figs. 3.2 and 3.3 are repeated, the robot continues to move in the same direction. Thus, cyclic body rotations induce RRRobot to translate along a straight line.

One challenge we focus on in legless locomotion is to find the leg trajectories which create the body attitude oscillations to induce RRRobot to locomote in the plane. We now briefly discuss the factors influencing RRRobot's body attitude oscillations and the space of body rotations that can be

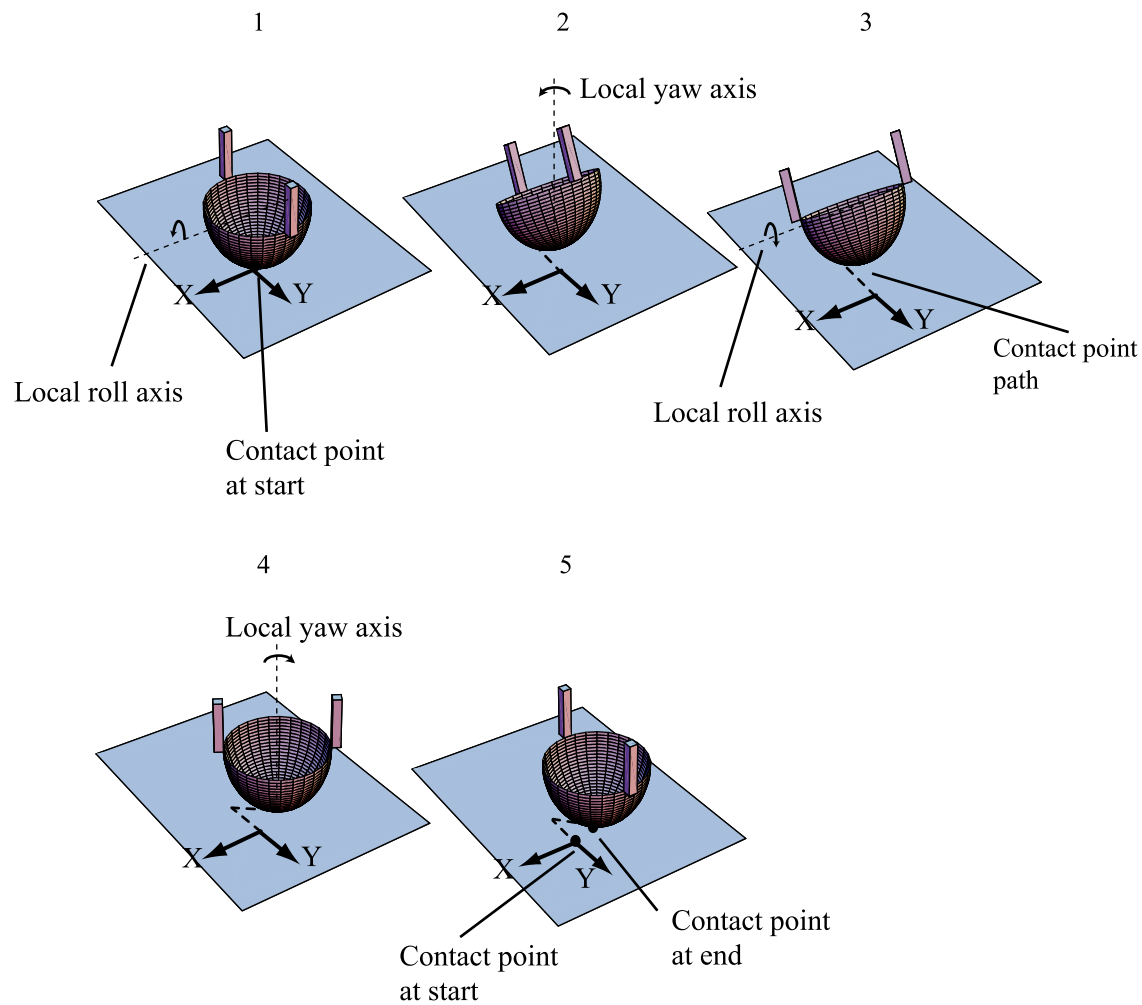


Figure 3.2: Body roll-yaw rotations that produce locomotion. Motions are represented as rotations about axes attached to the body but aligned with the world coordinate frame. There is a local roll rotation between positions 1 and 2 and positions 3 and 4; there is a local yaw rotation between positions 2 and 3 and positions 4 and 5.

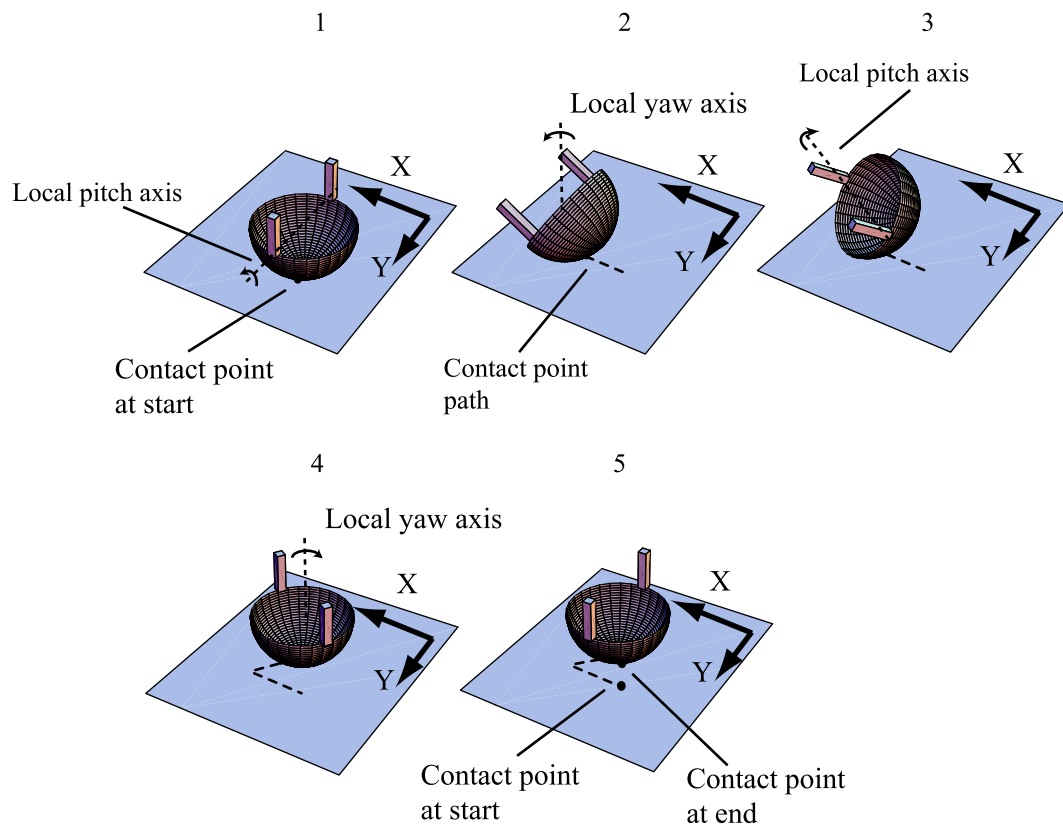


Figure 3.3: Body pitch-yaw rotations that produce locomotion. Motions are represented as rotations about axes attached to the body but aligned with the world coordinate frame. There is a local pitch rotation between positions 1 and 2 and positions 3 and 4; there is a local yaw rotation between positions 2 and 3 and positions 4 and 5.

induced through leg swings.

Since there are no kinematic constraints between RRRobot's legs (the actuated appendages) and the ground (the environment), the internal dynamics (the shape changes defined by leg motions) and gravitational drift define RRRobot's body roll, pitch, and yaw dynamics. RRRobot body roll and pitch attitude motions behave as an inverted pendulum forced by the leg motions, and the body pitch and roll static equilibrium configuration is determined by leg configuration. For example, if the legs are in the vertical configuration, the stable body pitch configuration is the zero pitch configuration; and if the legs are offset from the vertical, the robot's stable pitch configuration is offset in the direction of leg offset. Ground friction provides the damping to bring body pitching and rolling to a stop at the stable configuration.

In contrast to the inverted pendulum behavior of RRRobot's body roll and pitch motions, its yaw configuration is stable, since gravity does not produce any moment about the vertical Z-axis. Body yaw rotations are induced through the reaction forces produced by leg motions only, and the conservation of angular momentum principle provides insight into RRRobot's body yaw dynamics. For example, if the two legs move π out-of-phase with each other, they cause the robot body to yaw. This contrasts with zero phase-difference leg motions that produce no net motion. So, by choosing the correct leg phase difference, we can produce body yaw oscillations.

In addition to leg motions that induce body yaw oscillations, there exist cyclic leg motions that exploit inertia differences to induce net yaw. Producing net body yaw is important because it allows us to control the curvature of robot's path. As the body pitch-yaw oscillations induce linear translation, any net body yaw causes the robot's path to curve. We discuss RRRobot's body yaw dynamics in section 3.5

Finally, the direct alignment between RRRobot's pitch motions and the hip joints produces large-amplitude body pitch rotations when compared with body roll rotations. So the influence of body roll oscillations on RRRobot's motion is comparatively small.

This brief analysis shows that by carefully choosing leg motions, we can induce RRRobot's body roll, pitch, and yaw attitude to oscillate and the body's net yaw configuration to change. Such body attitude oscillations when coupled with the contact constraints offers a structured locomotion mode for a high-centered robot. We now formally explore RRRobot's dynamics using Lagrangian methods.

3.2 RRRobot Dynamics Model

We begin studying *legless locomotion* by modeling RRRobot on a plane. The RRRobot-on-a-plane model is a hemispherical shell with two short actuated legs sitting on a plane (see Fig. 3.1). The massless shell has radius r , and the massless legs have length l . There are five masses on the robot: a mass at the distal end of each leg (representing reaction masses M_l), a mass where each leg is pinned (representing servo mass M_s), and a mass at the bottom of the shell (representing battery and processor mass M_b). Torques τ_1 and τ_2 may be applied at the leg joints, the shell rolls on the plane without slip, and gravity induces natural roll and pitch oscillations.

RRRobot's configuration q_{rr} consists of the sphere's position and orientation (x, y, R) with respect to a spatial frame and the internal configuration of its legs (ϕ_1, ϕ_2) . We also refer to RRRobot's internal configuration as the robot's *shape*. Here $R = R(\theta_y, \theta_p, \theta_r) \in SO(3)$ represents the sphere's orientation according to the yaw-pitch-roll body-fixed angle convention [30]. Thus,

$$q_{rr} = (x, y, R(\theta_y, \theta_p, \theta_r), \phi_1, \phi_2)^T \in \mathbb{R}^2 \times SO(3) \times \mathbb{S}^1 \times \mathbb{S}^1. \quad (3.1)$$

The equations of motion for RRRobot on a plane can be derived using Lagrangian dynamics [30] and take the form

$$M(q_{rr})\ddot{q}_{rr} + C(q_{rr}, \dot{q}_{rr})\dot{q}_{rr} + G(q_{rr}) = \tau + (\lambda_1 \omega^1)^T + (\lambda_2 \omega^2)^T + \zeta_{rr}, \quad (3.2)$$

$$\omega^1 \dot{q}_{rr} = 0, \quad (3.3)$$

$$\omega^2 \dot{q}_{rr} = 0, \quad (3.4)$$

$$\omega^1 = (1, 0, 0, -r \cos \theta_y, -r \cos \theta_p \sin \theta_y, 0, 0), \quad (3.5)$$

$$\omega^2 = (0, 1, 0, -r \sin \theta_y, r \cos \theta_r \cos \theta_y, 0, 0), \quad (3.6)$$

where $M(q_{rr}) \in \mathbb{R}^{7 \times 7}$ is the positive-definite non-diagonal configuration-dependent mass matrix, $C(q_{rr}, \dot{q}_{rr})\dot{q}_{rr} \in \mathbb{R}^7$ is the vector of Coriolis and centrifugal terms, $G(q_{rr}) \in \mathbb{R}^7$ is the vector of gravitational terms, and $\tau = (0, 0, 0, 0, 0, \tau_1, \tau_2)^T$ is the generalized force. The generalized force τ indicates that only the legs are actuated.

The sphere-plane no-slip contact constraints [55] are defined by (3.3) and (3.4). The one-forms in (3.5) and (3.6) define the directions in configuration space along which the tangential contact forces act. The symbols $\lambda_1, \lambda_2 \in \mathbb{R}$ represent the magnitudes of the contact constraint forces, and ζ_{rr} represents viscous damping. Note that we bundle all body-ground contact losses, including dry and

viscous friction, into ζ_{rr} .

Note that RRRobot's equations of motion are complex (over one hundred terms) and understanding the contribution of various elements like robot shape, mass distribution, and control choices is difficult. So we discuss RRRobot's motion by splitting it into two parts: the sphere-plane contact kinematics and the leg-body dynamics. We now discuss RRRobot's contact kinematics.

3.3 Sphere-plane contact kinematics

In this section, we study RRRobot's sphere-plane contact kinematics [55] independent of the leg-body dynamics. The nonholonomic sphere-plane contact constraints given by (3.3) and (3.4) define the relationship between body orientation changes (defined in a body-fixed frame) and planar translation. For example, if the robot's body pitch configuration changes while body roll and yaw configuration are zero, the robot translates along the X-axis. Similarly, if the robot's body roll configuration changes while body pitch and yaw configuration are zero, the robot translates along the Y-axis. However, body yaw rotations produce zero translation, but change the robot's planar orientation. Note that elements 6 and 7 in ω^1 and ω^2 are zero, indicating that leg configuration does not contribute to the contact kinematics.

The constraints in (3.3) and (3.4) are identical for any sphere rolling on a plane, but RRRobot's inverted pendulum behavior and the resulting oscillatory body rotations distinguish its planar motion. First, since we are specifically interested in RRRobot's net translation, we characterize RRRobot's translation speed and direction using its net motion over a body rotation cycle and not its instantaneous state. Thus, even if RRRobot's body attitude oscillations cause it to deviate from its average path, we only use its net translation over a cycle to characterize its speed and direction. Second, since the contact constraints are non-integrable [55], we do not have an algebraic relationship between net translation and cyclic body rotations. This implies that a numerical analysis is required to understand the contact point's net motion for different body attitude trajectories.

An important concept in robotics is the space of configurations a robot can reach. For example, can RRRobot reach any point in the plane using small body-attitude oscillations? We now use specific examples to show that the contact constraints in (3.3) and (3.4) allow full planar accessibility by choosing different body-rotation trajectories. Since we are interested in the net planar translation over each cycle of RRRobot's body attitude oscillation, we restrict our analysis to the influence of body attitude oscillation amplitude and phase on RRRobot's translation. Note that body attitude oscillation frequency only time-scales the path, while body attitude offset does not influence the

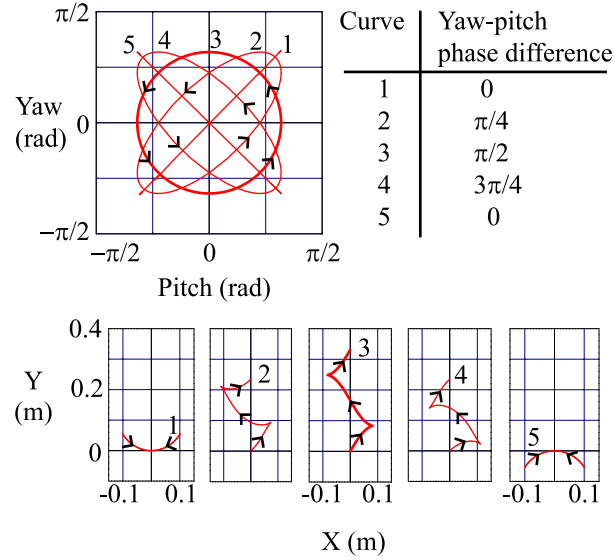


Figure 3.4: The path taken by a sphere rolling on a plane changes with body attitude trajectory. The figure shows how the contact point evolves over one cycle for different pitch-yaw phase relationships, given by $\theta_r = 0$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$.

sphere-plane contact geometry.

Fig. 3.4 shows that net contact-point displacement is restricted to the Y-axis for various unit-amplitude sinusoidal body pitch-yaw phase relationships (with zero body roll). Even though the contact point's X-coordinate oscillates because of the body pitch oscillations, the net displacement along the X-axis is zero. A similar relationship exists between net X-axis displacement and various body roll-yaw phase relationships (with zero pitch). With reference to legless locomotion, this simple analysis alludes to the body rotations that the leg motions must create to induce the robot to move in a specific direction; for example, to induce RRRobot to translate along the Y-axis, we need to find the leg motions that induce body pitch-yaw attitude oscillations.

Note that the net planar displacement is maximum when the body pitch-yaw (see Fig. 3.5) or roll-yaw phase difference is $\pi/2$ and is zero when pitch and yaw or roll and yaw are in phase or π out-of-phase. This implies that there is an optimal choice of body rotation phase relationships that induce maximum translation for each cycle of body rotation. Thus, for legless locomotion, it is advantageous to find the leg motions that induce body rotations with the phase relationship that produces maximum translation for each leg-motion cycle. We explore in section 3.4.2 the space of body attitude oscillations RRRobot's leg motions can produce.

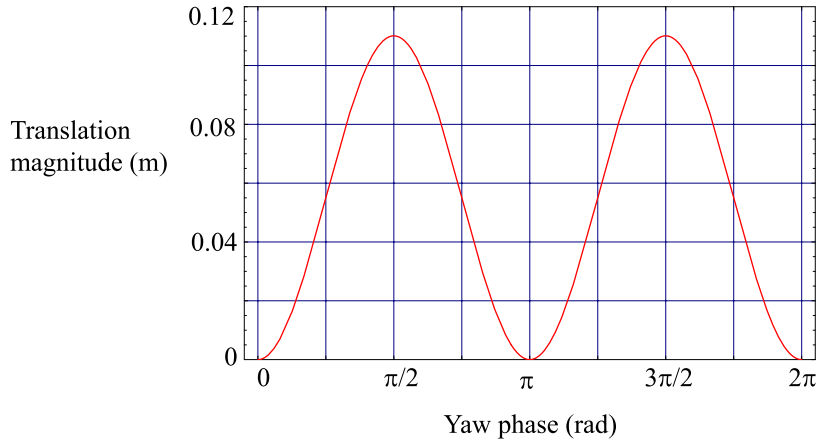


Figure 3.5: Translation magnitude as a function of body yaw phase (for a given pitch oscillation trajectory and zero roll) for a sphere (radius 0.12 m).

If body rotations are non-zero along all three axes, namely the body roll, pitch, and yaw axes, translation in any direction is possible. Fig. 3.6 shows translation direction as a function of the phase of sinusoidal roll and yaw oscillations with respect to pitch oscillations over each cycle. For the same amplitude of roll, pitch, and yaw rotations, the translation direction continuously varies with the relative phase difference between body roll and yaw rotations and body pitch and yaw rotations. For example, if the pitch-yaw phase difference is close to $\pi/2$ and the roll-yaw phase difference is close to zero, then the translation direction aligns closely with the Y-axis. Conversely, if the roll-yaw phase difference is close to $\pi/2$ and the pitch-yaw phase difference is close to zero, then the translation direction aligns closely with the body-fixed X-axis.

Note that the translation magnitude changes with the relative body rotation phase also. Fig. 3.7 shows the translation magnitude as a function of roll and yaw phases for fixed pitch phase.

Translation direction varies as a function of roll and pitch oscillation amplitudes also (see Fig. 3.8). As body roll amplitude increases relative to body pitch amplitude, translation direction shifts from the body-fixed Y-axis. Connecting back to legless locomotion, we note that RRRobot's body pitch rotations are larger than its body roll oscillations, because its hip joints are aligned with

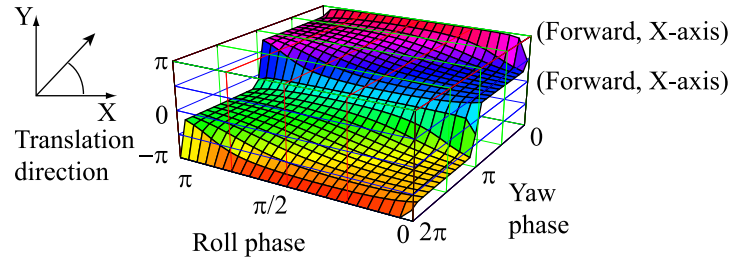


Figure 3.6: Translation direction for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = \sin(t + \beta_r)$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$.

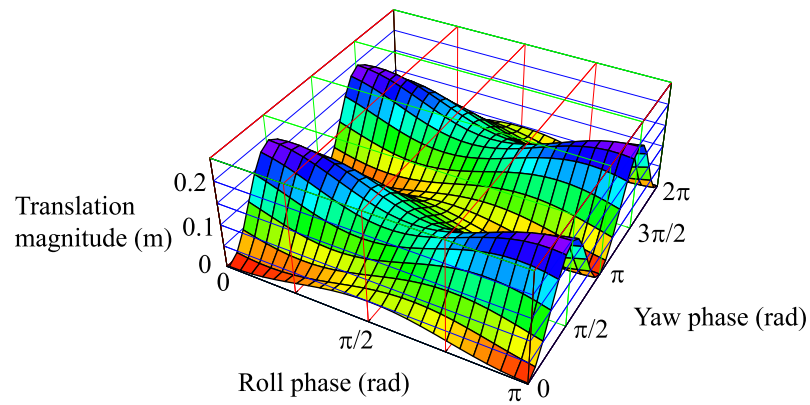


Figure 3.7: Translation magnitude for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = \sin(t + \beta_r)$, $\theta_p = \sin(t)$, and $\theta_y = \sin(t + \beta_y)$.

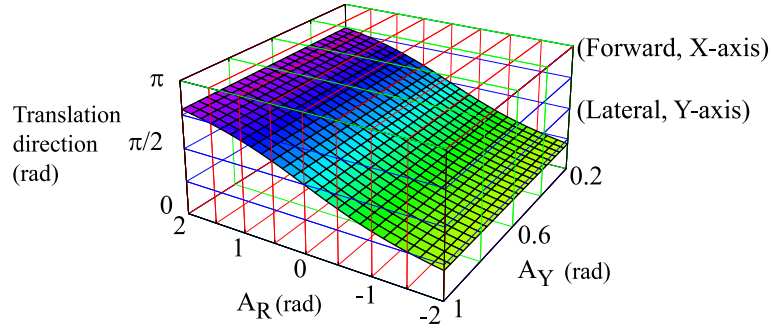


Figure 3.8: Translation direction for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = A_r \sin(t + \pi/2)$, $\theta_p = \sin(t)$, and $\theta_y = A_y \sin(t + \pi/4)$.

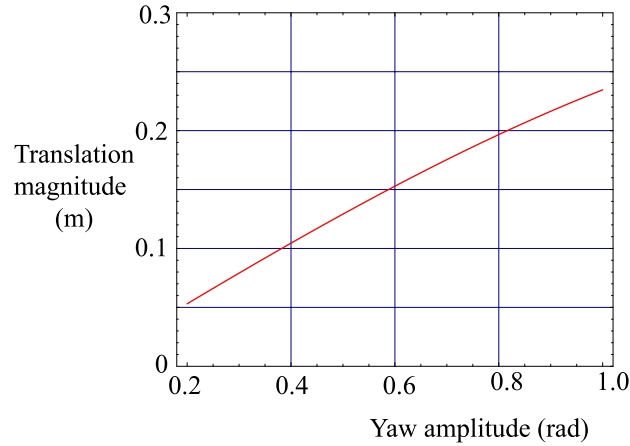


Figure 3.9: Translation magnitude for a sphere (radius 0.12 m) on a plane after one cycle in body-rotation space given by $\theta_r = 0$, $\theta_p = \sin(t)$, and $\theta_y = a_y \sin(t + \pi/4)$.

the body pitch axis. This indicates that RRRobot's predominant translation direction is along its body fixed Y-axis.

Also, translation magnitude depends on roll-pitch-yaw oscillation amplitudes. In particular, Fig. 3.9 shows how translation magnitude changes with yaw amplitude for a given pitch trajectory and zero roll oscillations.

Thus, by varying the body attitude phase and amplitude relationships, a sphere can *rock and roll* along any direction. The important fact here is that translation direction is fixed for a given phase

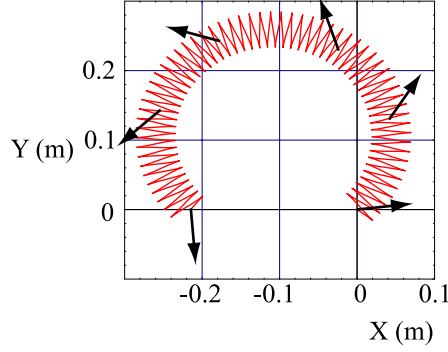


Figure 3.10: Contact-point time history for a sphere (radius 0.12 m) on a plane starting from the origin for the body-rotation phase relationship given by $\theta_r = \theta_p = 0.15 \sin(8t)$, and $\theta_y = 0.1t + 0.15 \sin(8t + \pi/2)$. The arrows indicate robot yaw orientation.

and amplitude relationship; that is, cyclic body attitude oscillations causes the robot to translate along a fixed direction.

So what body rotations cause the robot's path to curve? We get curved paths when the body yaw configuration drifts in addition to body pitch-yaw or roll-yaw oscillations. Fig. 3.10 provides an example of how translation curves when body yaw drifts—the robot moves in a circle. Furthermore, translation curvature increases with yaw drift rate—the circle radius becomes smaller.

Note that while we use sinusoidal body rotations to illustrate the contact kinematics, RRRobot's body attitude oscillations created by the leg motions are not necessarily sinusoidal. Also, even though the contact kinematics permit translation in any direction for different body attitude trajectories, RRRobot's leg-body dynamics limits the range of locomotion. For example, since RRRobot's roll amplitudes are small compared with body pitch oscillation amplitudes, translation along the body-fixed Y-axis dominates translation along the body-fixed X-axis. Similarly, the limited body roll-pitch-yaw phase relationships produced by the leg dynamics limit translation velocities. However, curved translation is possible since we can produce varying body-yaw drift rates using different leg trajectories (see section 3.5.2 and [16]). A challenge in legless locomotion is to find the leg motions that produce the body rotations to induce the desired translation. We now explore gaits for legless locomotion that allow full planar accessibility.

3.4 Legless Locomotion Gaits

This section presents leg motions that induce RRRobot to translate through the interplay of RRRobot's body attitude oscillations and the contact constraints. Many candidate leg motions, such as aperiodic and non-smooth trajectories, exist but we restrict our analysis to sinusoidal leg trajectories for simplicity.

We choose leg trajectories of the form $a \sin(\omega t + \beta) + \gamma$, where a represents the leg amplitude, ω the leg angular frequency, β the leg phase difference, and γ the leg offset. We mainly study the influence of leg offset and leg phase difference on RRRobot's translation, while fixing the leg amplitude and frequency.

We choose leg oscillation amplitudes that keep body oscillation amplitudes small to minimize body-ground contact slip. We set leg oscillation frequency slightly greater than the natural oscillatory frequency to excite reasonable amplitude body attitude oscillations. We avoid high and low frequency leg oscillations, since the body motions induced by such leg motions do not exhibit the slip-free rocking and rolling behavior we are interested in exploring.

In all our experiments, the robot starts from rest at the origin with the legs in the vertical position, and a PD controller tracks the desired leg trajectories. The robot has no feedback about its global position and body orientation and runs open loop.

We now present results from experiments with a tethered RRRobot and an untethered RRRobot, focused on studying legless locomotion's gaits. We simultaneously compare the experimental results to those from simulation (using RRRobot's equations of motion in (3.2)). In section 3.4.2, we then analyze using simulation the range of planar translations that RRRobot's leg motions allow.

3.4.1 Demonstrating Legless Locomotion Using Experiments and Simulation

Our legless locomotion experimental set-up consists of two versions. The first version has a suspended tether providing servo power and control signals. The second version is autonomous with the controller and power supply on-board, thus eliminating disturbances from the tether. We explore the influence of varying the leg offsets on RRRobot's planar translation, while keeping leg frequencies, leg phase difference, and leg amplitude fixed.

We use the following parameter values: servo mass $M_s = 0.053$ kg, leg reaction masses $M_l = 0.057$ kg, leg length $l = 0.1$ m, and gravity $g = 9.81$ m/s. Note that RRRobot's motion is particularly sensitive to mass distribution differences and ground traction, since RRRobot's motion relies on slip-free contact and small inertia differences arising from leg swings. We ensure that these effects

are not overwhelmed by the larger body pitch oscillations and body-ground slip.

With only the servos hinged on the body, we can control the untethered RRRobot mass distribution carefully, but must ensure that the disturbances from the tether are small. The tethered RRRobot’s radius is 0.12 m, and battery and processor are modeled with weights $M_b = 0.3$ kg. The untethered RRRobot presents a different challenge—while the robot is autonomous, its mass distribution must be carefully tuned using weights to ensure symmetry, since the processor and battery weight is fixed to the hemisphere bottom. The untethered RRRobot’s radius is 0.15 m, and battery and processor mass is $M_b = 0.168$ kg.

We explore individual gaits in the tethered version, while we explore gait transitions in the untethered version. In both versions, we keep body oscillations small to minimize wheel-ground slip. This is required for proper comparison with the simulation results, which assume slip-free body-ground contact.

We develop our simulation in Mathematica [2]. While we can model many of legless locomotion’s parameters like RRRobot’s mass distribution, the body shape, and the leg trajectories, the body-ground traction losses are difficult to model. For example, when RRRobot is released from a non-zero body pitch or roll configuration and allowed to oscillate freely without leg swings, the oscillation amplitude decreases with time, indicating that the robot loses energy when rolling on the ground. Also, if RRRobot is spun about its contact point (that is the robot yaws in the plane), the robot stops rotating after some time. This is again because the robot loses energy when spinning about its contact point.

The energy losses when RRRobot rolls on the ground or spins about the contact points arise from a combination of dry and viscous friction. Dry friction results from body-ground slip, while viscous friction results from the body-ground rolling and aerodynamic interaction.

Dry friction is easy to model using, say, a Coloumb friction model, but makes RRRobot’s dynamics more complex when included. This is because RRRobot’s dynamics becomes hybrid; that is, RRRobot’s dynamics transitions between two models depending on whether the traction forces exceed the friction cone—one model when the body slides on the ground (traction forces exceed the friction cone, and there is energy loss), and another when the body rolls on the ground (traction forces inside the friction cone, and there is no energy loss). Since this thesis focuses on understanding RRRobot’s novel coupled motion and not on modeling the contact between two smooth surfaces, we ignore dry friction in our RRRobot simulation, and bundle all the body-ground contact losses into viscous damping coefficients. Furthermore, this implies that we do not model differences between static and dynamic friction also.

We model viscous damping losses as being proportional to configuration velocities and may be included without resorting to hybrid models. We choose viscous damping coefficients that permit a qualitative comparison between RRRobot's motion in experiment and simulation. However, our choice of ignoring dry friction losses results in differences between simulation and experiment.

RRRobot's Linear And Curved Translation Gaits

In this section, we explore RRRobot's linear and curved translation gaits individually using simulations and experiments with the tethered RRRobot. We explore two specific versions of the sinusoidal gait: in Gait 1, the legs oscillate about the vertical position (offset $\pi/2$), and in Gait 2, the legs oscillate about a position $\pi/4$ off the horizontal in Gait 2. The leg amplitude is set to 0.3 rad and leg angular frequency to 7.5 rad/s. The experiments and simulations run for one hundred seconds, and an overhead camera tracks the robot-ground contact point's motion.

Gait 1 produces translation along the Y axis (see Fig. 3.11) by inducing body pitch-yaw oscillations and negligible body roll rotations. The robot translates at about 2.5 mm/sec in experiment and about 8 mm/sec in simulation.

Gait 2 produces counter-clockwise circular translation (see Fig. 3.12) due to a combination of body pitch-yaw oscillations, body yaw drift, and small roll oscillations. The robot covers half a circle in experiment and simulation, translating at about 2.5 mm/sec and turning at about $1.5^\circ/\text{sec}$. In both gaits, swapping the relative phase between the two legs produces translation in the opposite direction. The initial transients produced by ramping the legs from rest into the desired trajectories are visible in experiment and simulation and dampen out after a few cycles.

Our experience indicates that gaits with leg offset close to the horizontal ($0, \pi$) are not reliable. This is because as the leg offset gets closer to the horizontal, the body pitch oscillations become smaller and the body roll oscillation amplitudes are too small to overcome ground traction losses. Thus, RRRobot's translation becomes very small and loses its characteristic pitch-yaw oscillatory rotations.

We thus focus on gaits with offset between $\pi/4$ and $3\pi/4$ (45° either side of the vertical).

The robot's paths in simulation and experiment match qualitatively. In particular, the path curvature in simulation and experiment match well—the robot translates linearly (curvature zero) in gait 1, while it curves with radius approximately 0.1 m in gait 2. While the robot's translation velocity in gait 2 compares well between experiment and simulation, the robot's translation velocity in gait 1 is smaller in experiment than in simulation.

The difference in gait 1's translation velocity between experiment and simulation is probably

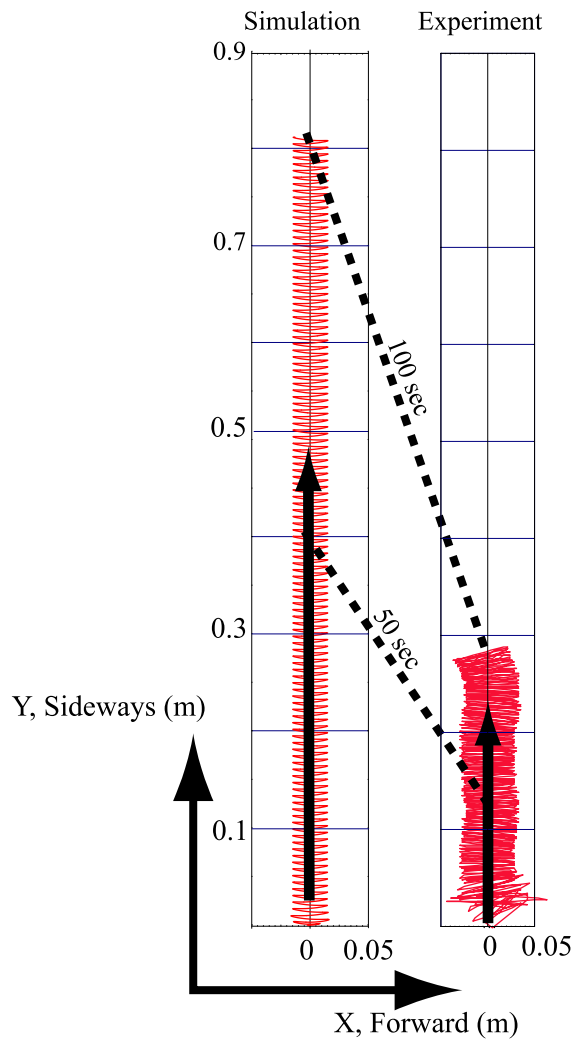


Figure 3.11: Planar plots of contact-point time history during sideways locomotion produced by Gait 1 in RRRobot-on-a-plane simulation and RRRobot-on-a-plane experiment. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.

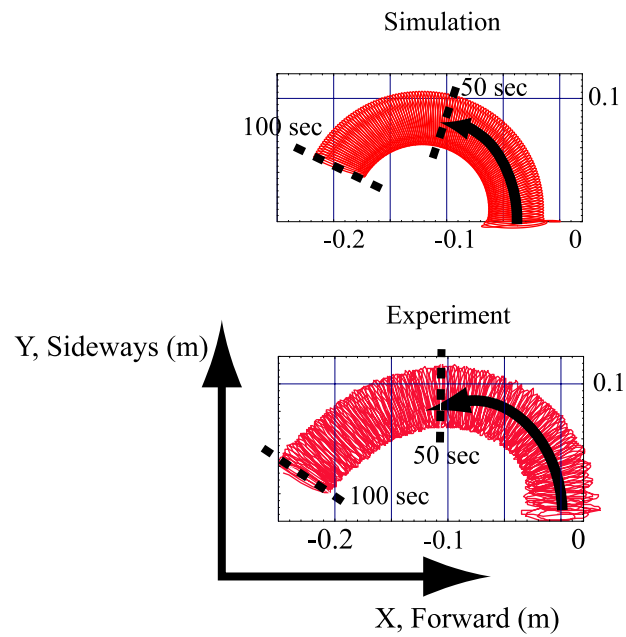


Figure 3.12: Planar plots of contact-point time history during counter-clockwise circular locomotion produced by Gait 2 in RRRobot-on-a-plane simulation and RRRobot-on-a-plane experiment. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.

due to errors in modeling ground friction. Body-ground friction involves a combination of dry friction (Coulomb friction) and viscous friction (velocity dependent). These elements impede motion by dissipating the system's energy. Since we bundle all the ground friction effects into viscous friction in simulation, some discrepancy between experiment and simulation exists.

For example, in simulation, we assume that the robot can freely yaw about the contact point with a little viscous damping. Also, simulation does not include dynamic and static friction effects. In experiment, however, the body cannot freely yaw due to a combination of dry and viscous friction. This causes the robot's back-and-forth planar motions to be more closely spaced in experiment than the back-and-forth motions seen in simulation.

This effect is particularly severe for gait 1, because the translation produced by gait 1 is a result of the body-pitch oscillations, and the body is required to yaw when the body pitch and roll velocities are zero. The static friction at the extreme pitch and roll configurations impedes body yaw motion. This effect is smaller in gait 2, since the robot rolls, pitches, and yaws simultaneously. Thus, the pitch-roll velocity is never zero and the dynamic friction does not impede yaw rotations as much as static friction. This effect is apparent in the spacing between RRRobot's back-and-forth paths in the plane—the paths are closely spaced in gait 1, when compared to the paths in gait 2. This difference in ground resistance to yawing at a point is similar to the relative difficulty of turning a car's steering wheel when the car is stationary when compared to turning the car's steering wheel when the car is moving.

There are many parameters in legless locomotion, namely the body shape, the leg trajectories, the mass distribution, and body-ground friction and slip. We have mostly eliminated differences between experiment and simulation in all these parameters except in body-ground friction and slip, since modeling body-ground friction is difficult. Our simulations bundle all losses into the viscous damping coefficients by setting viscous damping ζ_{rr} to

$$\zeta_{rr} = (-0.6\dot{x}, -0.6\dot{y}, -0.05\dot{\theta}_r, -0.0045\dot{\theta}_y, -0.01\dot{\theta}_p, -0.01\dot{\phi}_1, -0.01\dot{\phi}_2)^T. \quad (3.7)$$

Note that the viscous damping coefficients in the translation and rotation freedoms are coupled. For example, the damping losses along the Y axis are related to the damping losses along the pitch axis, but it is difficult to find the exact mapping between the losses in each freedom. We choose damping coefficients to provide a qualitative comparison between the robot's motion in experiment and simulation.

In summary, since legless locomotion is a dynamically coupled locomotion mode where the

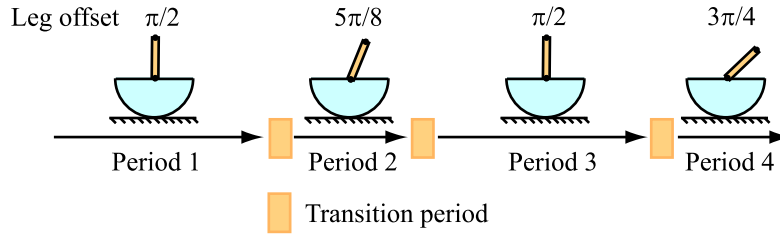


Figure 3.13: Studying gait transitions in legless locomotion: the offset of the sinusoidal gaits change from $\pi/2$ to $5\pi/8$ to $\pi/2$ and then to $3\pi/4$.

dynamic effect of the leg swings is transferred through the contact geometry to produce motion, the losses are larger and more difficult to model when compared with a conventional locomotion mode like walking. Given that legless locomotion is an unconventional locomotion mode designed for locomoting a high-centered robot and that legless locomotion is slow anyway, we believe that the difference in translation velocities between experiment and simulation is not significant.

We now present experiments and simulation to explore gait transitions for RRRobot.

RRRobot Gait Transitions

While we presented an individual analysis of RRRobot's linear and curved translation gaits in the previous subsection, we now present an analysis of RRRobot's behavior when its gaits are sequenced. In particular, we use computer simulations and experiments with an untethered RRRobot to study how leg trajectory transients affect RRRobot's translation when changing from a linear translation gait to a curved translation gait and vice versa. We use sinusoidal leg trajectories (amplitude 0.65 rad, phase difference $\pi/2$, and frequency 7.5 rad/s) with the following sequence of leg offsets (see Fig. 3.13): leg offset $\pi/2$ for eighty cycles (linear translation, period 1), leg offset $5\pi/8$ for forty cycles (curved translation, period 2), leg offset $\pi/2$ for eighty cycles (linear translation, period 3), and leg offset $3\pi/4$ for forty cycles (curved translation, period 4). Smooth transitions from one gait to another are executed inside ten cycles. A Vicon motion-capture system [7] tracks the motion of the robot body (six markers on the hemisphere rim) and the legs (two markers on each leg) at 120 frames/sec. The motion-capture data permits us to perform a complete geometric analysis of RRRobot's motion.

Fig. 3.14 shows the time history of planar translation in experiment, and Fig. 3.15 shows the time history of planar translation in simulation. Since RRRobot runs open-loop, the robot drifts from the expected path in experiment due to unmodeled disturbances and transients. This is seen

particularly in the periods where the robot is supposed to translate linearly (periods 1 and 3). Note that such drift can be corrected with feedback using computer vision or inertial sensors.

The robot's planar path in experiment and simulation match qualitatively. But one difference is RRRobot's slow linear translation velocity in experiment when compared with its velocity in simulation. We attribute this to the errors in ground traction modeling (as discussed in the previous subsection). RRRobot's linear translation gait experiences significant dry ground friction, which is difficult to model. Since we bundle dry and viscous ground friction into viscous damping coefficients in our simulations, differences between RRRobot's translation in experiment and simulation are inevitable, and we notice this in the linear translation gait. We also notice that the transients dampen out more quickly in simulation than in experiment. The important point from Figs. 3.14 and 3.15 is that RRRobot can translate with variable curvature and has complete planar accessibility using different leg trajectory offsets.

In addition to measuring RRRobot's planar translation, the motion capture data allows us to quantify RRRobot's body rotations and leg motions seen in our experiments and simulations. Figs. 3.16 and 3.17 show the leg trajectories executed during the experiment. The change in leg offsets from $\pi/2$ to $5\pi/8$ to $\pi/2$ and then to $3\pi/4$ between the periods is evident. Fig. 3.18 shows the leg phase differences during each of the periods; while we expect to see circles because the commanded leg trajectories are $\pi/2$ out-of-phase with each other, the distorted circles indicate that there are errors in tracking the commanded leg trajectories.

Fig. 3.19 shows the time history of body pitch rotations in experiment, while Fig. 3.20 shows the time history of body pitch rotations in simulation. In both experiment and simulation, the mean body pitch configuration during periods 2 and 4 are different from those in periods 1 and 3, since the leg offsets are different and gravity causes the robot to lean (pitch) in the offset direction. However, while the mean pitch configuration during periods 1 and 3 is zero in simulation, the robot pitch configuration is offset from the vertical in experiment due to small leg position and mass distribution errors in RRRobot. This causes RRRobot to translate along a curve during periods 1 and 3 in experiment, rather than along a straight line. Also, the pitch offset transients take longer to dampen out in experiment than in simulation. For example, the transients during period 3 dampen out slowly in experiment, causing the body pitch oscillations to slowly settle into the limit cycle. This causes the translation path to curve during period 3 before settling into a linear translation path.

There is a strong similarity in RRRobot's body yaw motion between the experiment and simulation (see Figs. 3.21 and 3.22), except during period 3 when the body pitch transients take a long time to settle. RRRobot translates linearly during period 1 and curves during periods 2 and 4. Simply

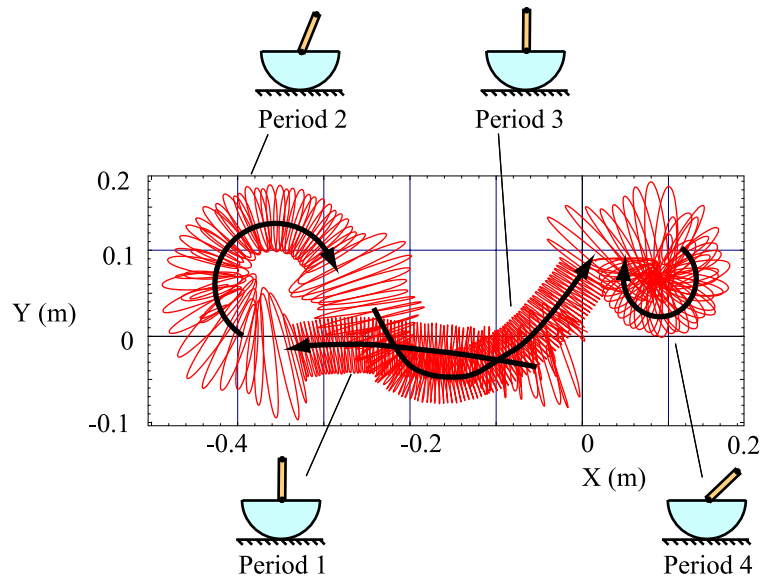


Figure 3.14: Untethered RRRobot experiment: planar translation.

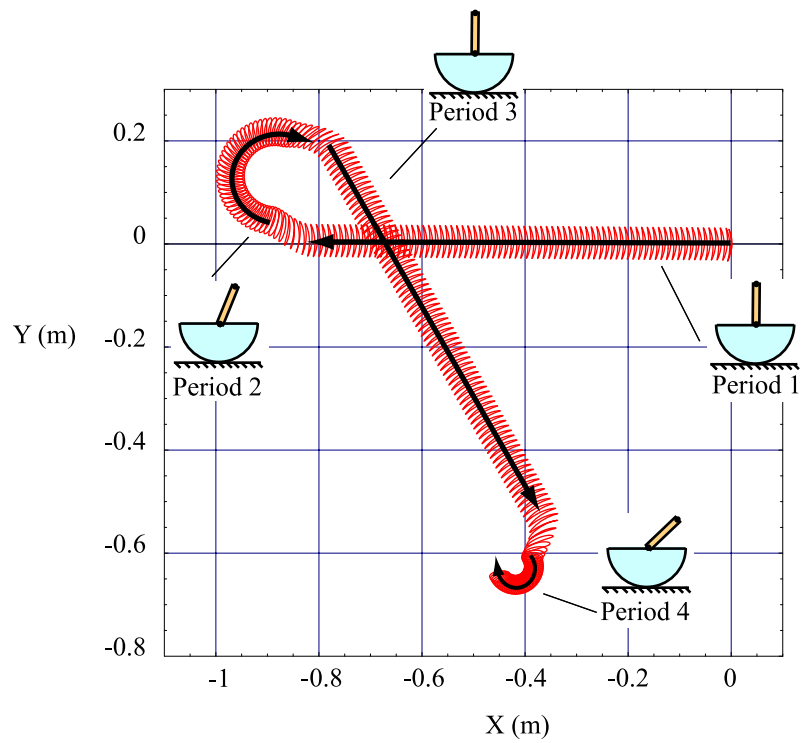


Figure 3.15: RRRobot simulation: planar translation.

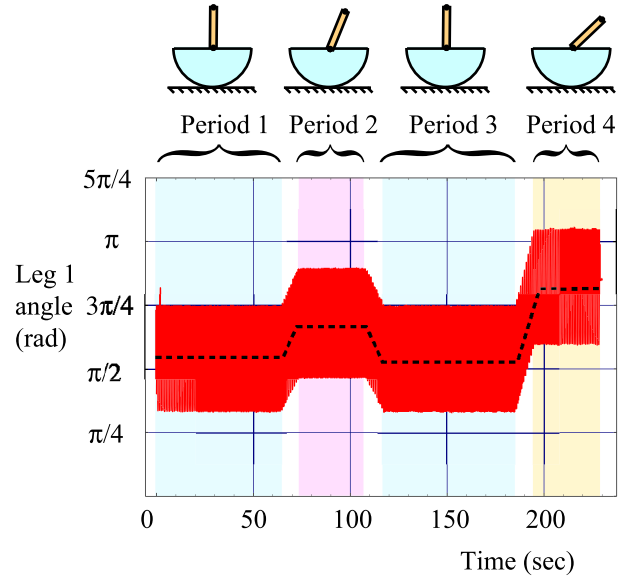


Figure 3.16: Untethered RRRobot experiment: leg 1 trajectory. The dotted lines indicate leg offset position.

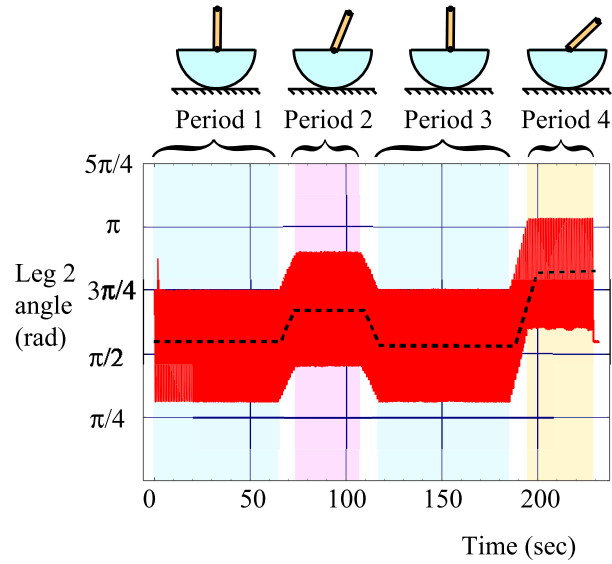


Figure 3.17: Untethered RRRobot experiment: leg 2 trajectory. The dotted lines indicate leg offset position.

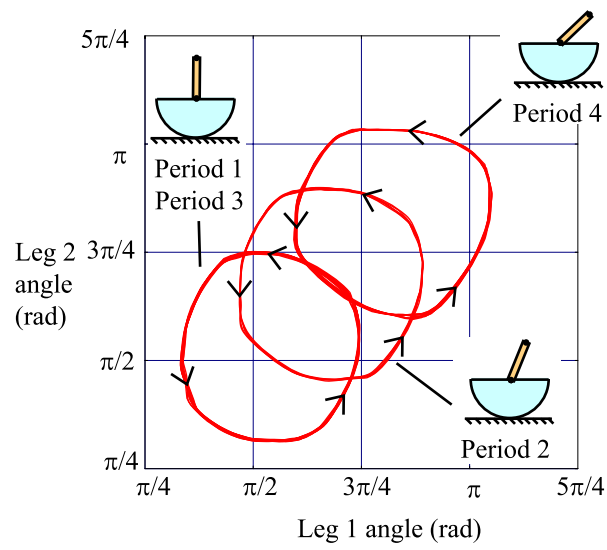


Figure 3.18: Untethered RRRobot experiment: phase relationship between leg motions during each period.

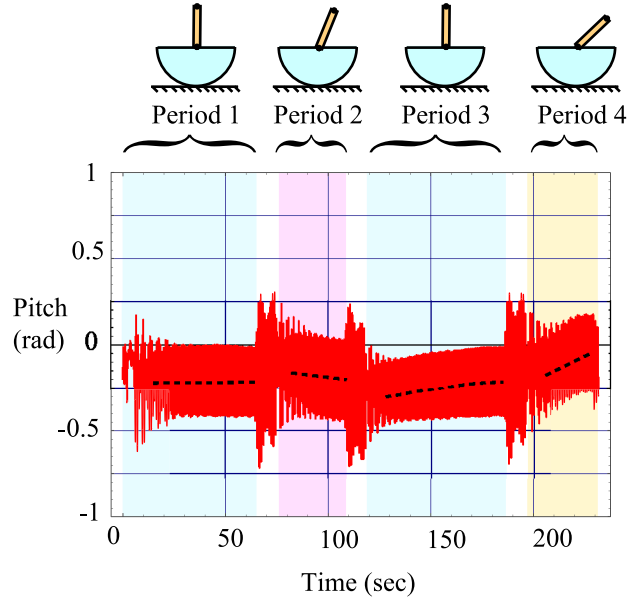


Figure 3.19: Untethered RRRobot experiment: body pitch trajectory. The dotted lines indicate body pitch offset position.

put, RRRobot's translation curves in the direction the robot leans.

Figs. 3.23 and 3.25 show the relationship between RRRobot's body pitch and yaw rotations during period 1 and period 4 in experiment, while Figs. 3.24 and 3.26 show the relationship between RRRobot's body pitch and yaw rotations during the same periods in simulation. First, we note that the body pitch and yaw rotations are out-of-phase with each other in period 1, as evidenced by the non-zero area under the curve. As presented in section 3.3, such out-of-phase body attitude oscillations when coupled with the nonholonomic contact constraints produce net translation; this compares well with the linear translation induced by body pitch-yaw oscillations in Fig. 3.4. Second, we note that body yaw configuration drifts during period 4, due to the out-of-phase leg oscillations offset from the vertical. Body yaw drift combined with body pitch-yaw oscillations induces RRRobot's translation to curve. This compares well with the example seen in Fig. 3.10 where body yaw drift causes the robot to locomote in a circle. Third, we note that the pitch-yaw body rotations are not identical in simulation and experiment due to modeling errors. In simulation, the yaw oscillation amplitude is bigger; also the pitch-yaw phase relationship is closer to $\pi/2$ in experiment than in simulation.

Fig. 3.27 shows the time history of body roll rotations in the experiment. Note that the roll-oscillation amplitudes are significantly smaller than the pitch oscillation amplitudes and have small

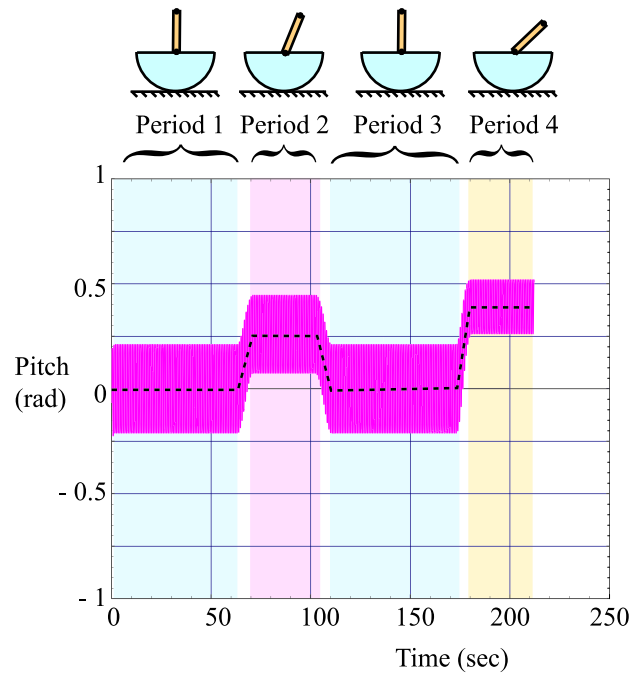


Figure 3.20: RRRobot simulation: body pitch trajectory. The dotted lines indicate body pitch offset position.

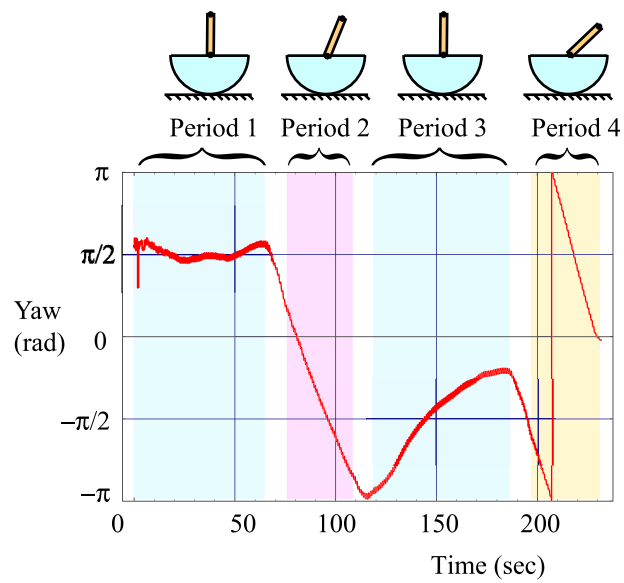


Figure 3.21: Untethered RRRobot experiment: body yaw trajectory.

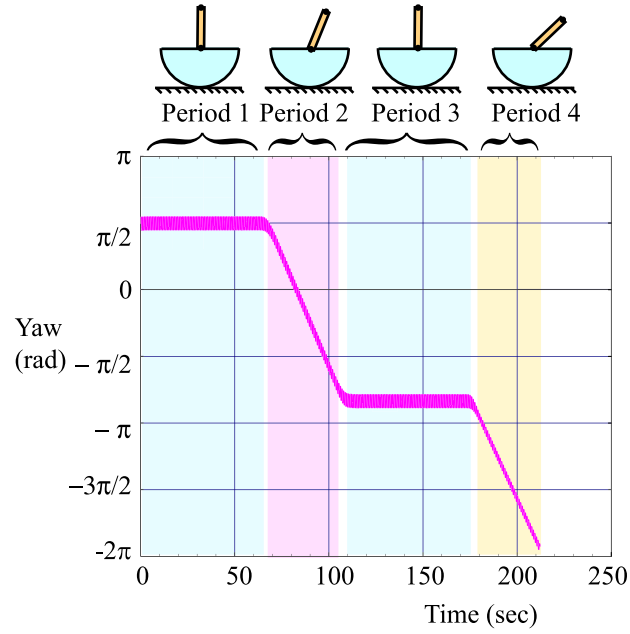
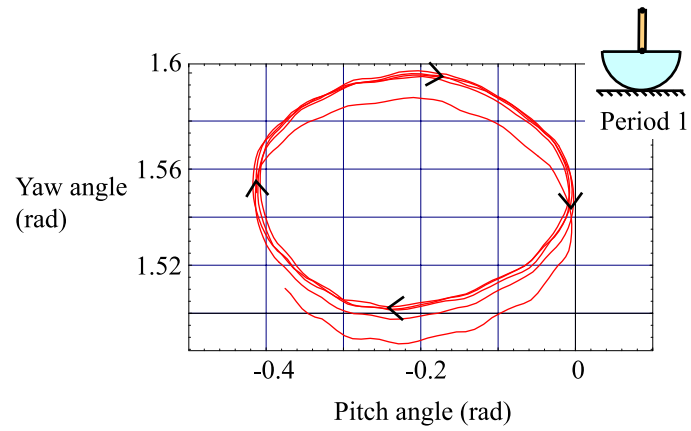


Figure 3.22: RRRobot simulation: body yaw trajectory.

Figure 3.23: Untethered RRRobot experiment: phase relationship between pitch and yaw rotations during period 1 ($t \in [37.5, 41.7]$ seconds).

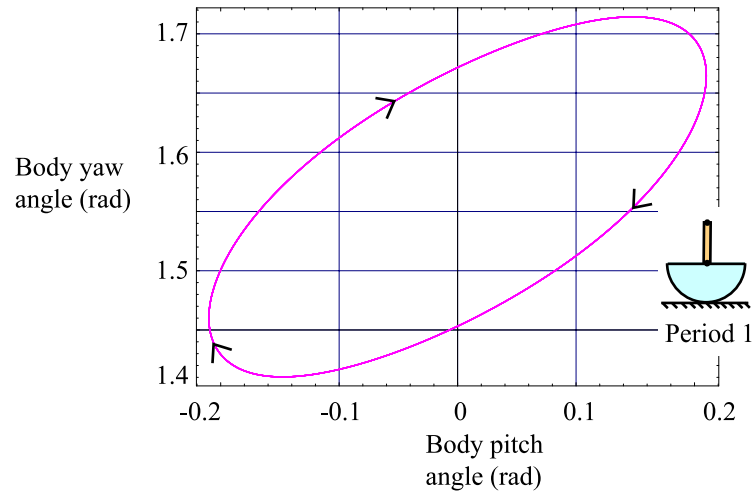


Figure 3.24: RRRobot simulation: pitch-yaw phase relationship during period 1 ($t \in [20, 30]$ seconds).

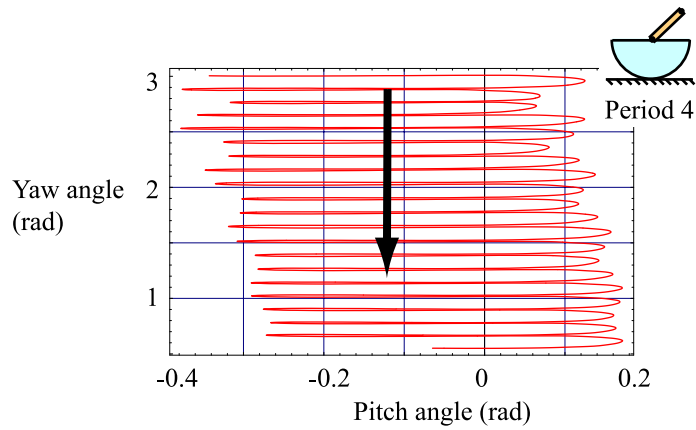


Figure 3.25: Untethered RRRobot experiment: phase relationship between pitch and yaw rotations during period 4 ($t \in [208, 225]$ seconds).

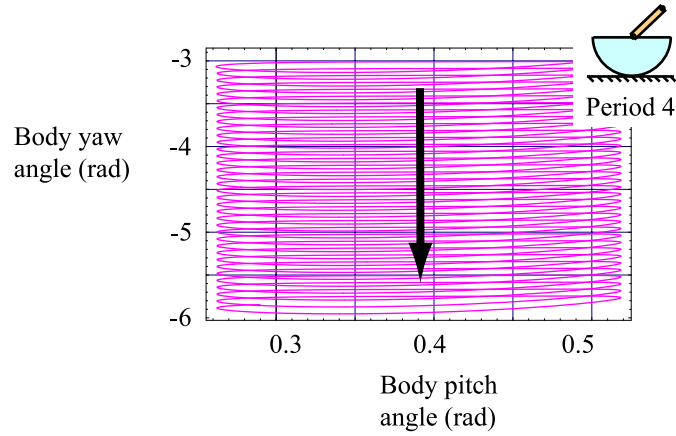


Figure 3.26: RRRobot simulation: pitch-yaw phase relationship during period 4 ($t \in [180, 210]$ seconds).

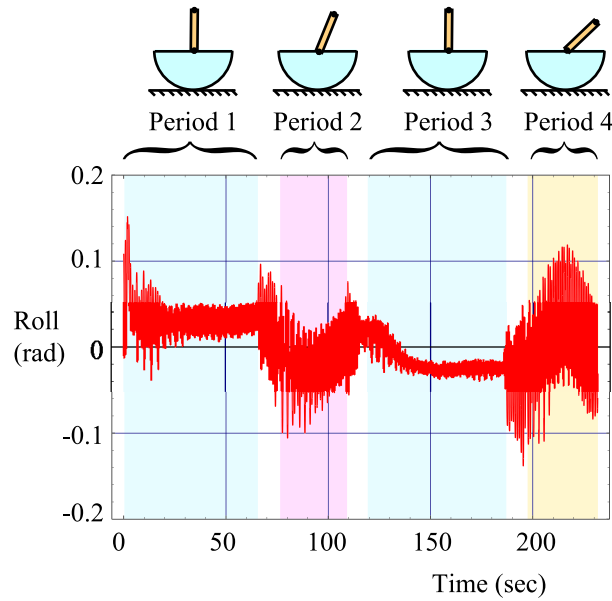


Figure 3.27: Untethered RRRobot experiment: body roll oscillations.

impact on RRRobot's translation direction. The small roll oscillations are important though in the curved translation gaits (as discussed in the previous subsection).

We also scanned RRRobot's body curvature to see if the plastic hemisphere deforms with the weight, since body deformations near the contact point influence the contact kinematics. In the nominal operating configuration, the sphere radius is 0.156 m, whereas the sphere in an inverted configuration has radius 0.162 m (we fit a sphere to a set of points using Chang and Pollard's

code [27]). Note that we assume sphere radius is 0.15 m in our simulations. Despite the discrepancies in robot radius, the planar paths traversed in simulation and experiment qualitatively match.

Our simulations model ground traction by setting viscous damping ζ_{rr} to

$$\zeta_{rr} = (-2.0\dot{x}, -2.0\dot{y}, -0.01\dot{\theta}_r, -0.004\dot{\theta}_y, -0.01\dot{\theta}_p, -0.01\dot{\phi}_1, -0.01\dot{\phi}_2)^T. \quad (3.8)$$

Note that these damping parameters are different from those used in simulating the smaller radius RRRobot (as discussed in the previous subsection), since the ground traction losses differ between the two cases. Again, note that the translation and rolling damping coefficients are coupled, and it is difficult to find the exact mapping between the losses in different freedoms. We note that the damping coefficients only approximate the ground traction forces seen in experiment to find a qualitative agreement between the robot's translation in experiment and simulation.

Finally, note that the large ground friction forces in experiment (and modeled using viscous damping in simulation) cause RRRobot to move slowly. In the following section, we present simulation results with lesser viscous damping to model slip-free body-ground contact. As expected, this allows RRRobot to translate faster.

3.4.2 Exploring Legless Locomotion Capabilities Using Simulation

In this subsection, we explore using simulation the full range of motions available to RRRobot by varying its leg trajectories. We use the following parameter values: servo mass $M_s = 0.053$ kg, leg reaction masses $M_l = 0.057$ kg, battery and processor mass $M_b = 0.3$ kg, sphere radius $r = 0.12$ m, leg length $l = 0.1$ m, and gravity $g = 9.81$ m/s. We use leg amplitude 0.3 rad and leg frequency 8 rad/s and set viscous damping to

$$\zeta_{rr} = (0, 0, -0.01\dot{\theta}_r, -0.01\dot{\theta}_y, -0.01\dot{\theta}_p, -0.01\dot{\phi}_1, -0.01\dot{\phi}_2)^T. \quad (3.9)$$

Note that these damping coefficients are different from those we use to match experimental results in section 3.4.1. The damping coefficients used in the previous section try to model the dry friction from body-ground slip and the viscous damping from ground interaction, while from this section onwards we assume slip-free body-ground contact and only model the viscous damping.

Fig. 3.28 shows RRRobot's translation induced by $\pi/2$ out-of-phase leg motions about the vertical configuration (simulation duration twenty five seconds), while Fig. 3.29 shows RRRobot's

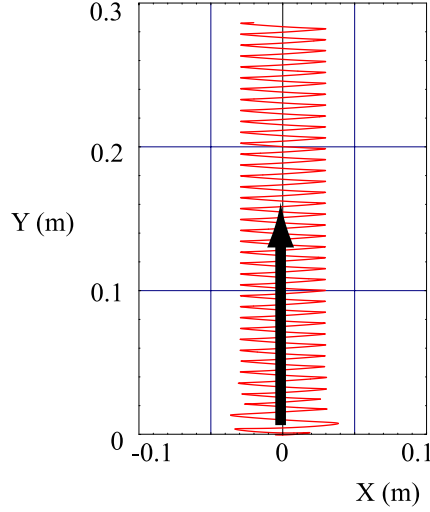


Figure 3.28: RRRobot simulation: time history of contact-point motion induced by out-of-phase leg motions about the vertical configuration over thirty seconds.

translation induced by $\pi/2$ out-of-phase leg motions offset $\pi/4$ from the vertical configuration (simulation duration twenty five seconds). Note that the translation velocity is lesser in the curved translation gait. This is because the robot pitches, yaws, and rolls simultaneously, causing the contact point to trace loops in the plane. In contrast, RRRobot's body only pitches and yaws in the linear translation gait and does not trace loops.

Fig. 3.30 shows how RRRobot's planar translation changes with varying leg offset and phase difference, based on RRRobot simulations using leg amplitude 0.3 rad and leg frequency 8 rad/s (simulation duration one hundred seconds). Note that translation curvature depends predominantly on leg offset, and RRRobot curves in the direction it leans (pitches) in. For example, the robot curves to the left with leg offset $\pi/4$ and curves to the right with leg offset $3\pi/4$. Also, translation curvature is symmetric as leg offset varies either side of the vertical configuration. Thus, the simulations indicate, like the experiments, that translation with variable curvature is possible.

This compares well with the intuition offered by the contact kinematics analysis in section 3.3, where we showed that pitch-yaw oscillations induce lateral translation and such oscillations when combined with yaw drift induce curved motion. This is exactly what we see in RRRobot's dynamic motion—out-of-phase leg motions about the vertical produce body pitch-yaw oscillations which induce RRRobot to translate along its body-fixed Y-axis; when RRRobot's leg offset shifts from the vertical, RRRobot's body yaws and induces curved translation.

Translation velocity varies depending on both leg offset and phase difference. Gauging from

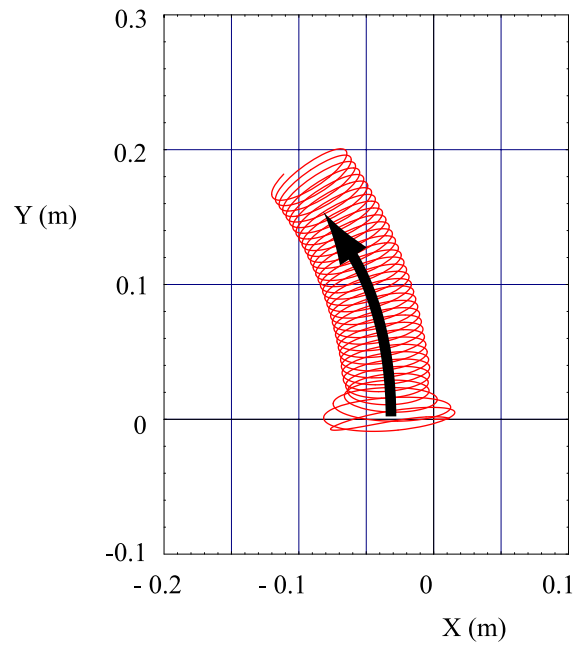


Figure 3.29: RRRobot simulation: time history of contact-point motion induced by out-of-phase leg motions offset $\pi/4$ from the vertical configuration over thirty seconds.

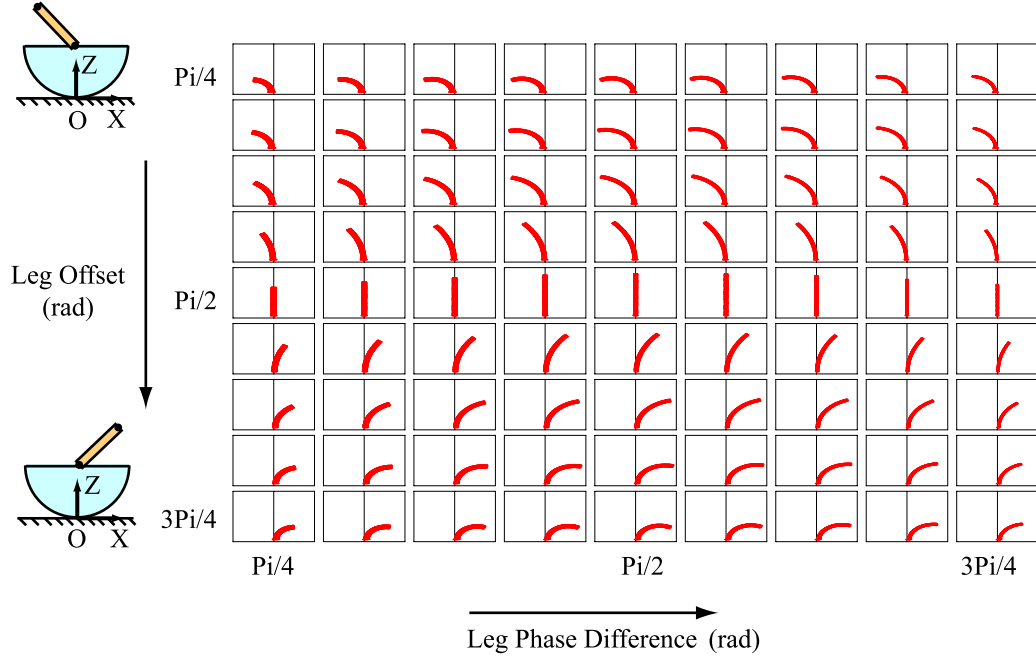


Figure 3.30: Translation curvature as a function of leg trajectory offset and phase difference: each plot shows the time history of contact-point motion over one hundred seconds. The X-axis range is $[-0.6, 0.6]$ m, and the Y-axis range is $[0, 1.1]$ m.

the path lengths in Fig. 3.30, we notice that translation velocity is maximum (1 cm/sec) with leg offset $\pi/2$ and phase difference $\pi/2$, since this induces large body pitch-yaw oscillations with body pitch-yaw phase-difference close to $\pi/2$.

Figs. 3.31, 3.32, 3.33, and 3.34 analyze RRRobot's steady-state body rotations as a function of leg offset and phase difference. Notice that body roll oscillation amplitudes are much smaller than body pitch oscillation amplitudes. Large body pitch and yaw amplitudes induce greater translation, as discussed in section 3.3. RRRobot's body pitch and yaw amplitudes do not vary much with leg offset, but change significantly with leg phase difference—as leg phase difference approaches zero, pitch oscillation amplitude increases and yaw oscillation amplitude decreases; as leg phase difference approaches π , pitch oscillation amplitude decreases and yaw oscillation amplitude increases.

Also the body yaw-pitch phase difference varies little from $\pi/2$. We noticed in section 3.3 that this body pitch-yaw phase difference produces maximum translation velocity. So this choice of parameters, namely RRRobot's body masses, leg masses, and body curvature, induce RRRobot to translate with maximum linear velocity naturally.

We now summarize our experimental simulation results so far.

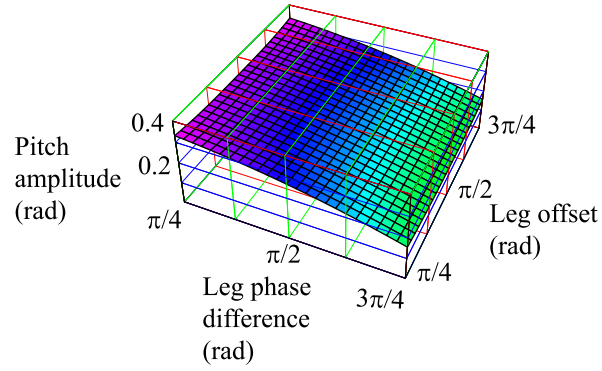


Figure 3.31: RRRobot pitch rotation amplitude as a function of leg trajectory offset and phase difference.

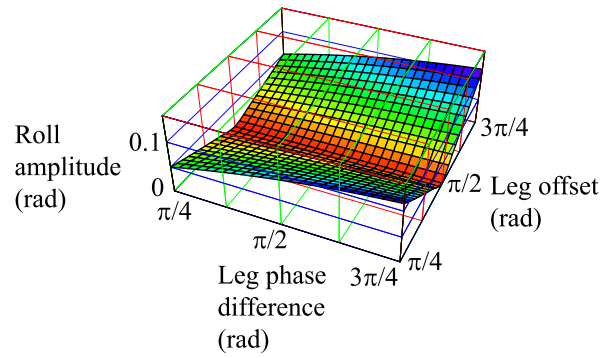


Figure 3.32: RRRobot roll rotation amplitude as a function of leg trajectory offset and phase difference.

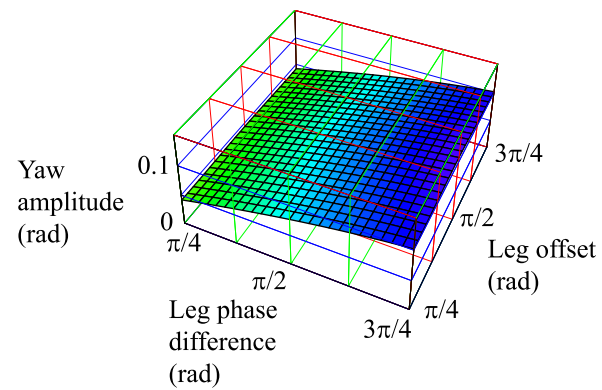


Figure 3.33: RRRobot yaw rotation amplitude as a function of leg trajectory offset and phase difference.

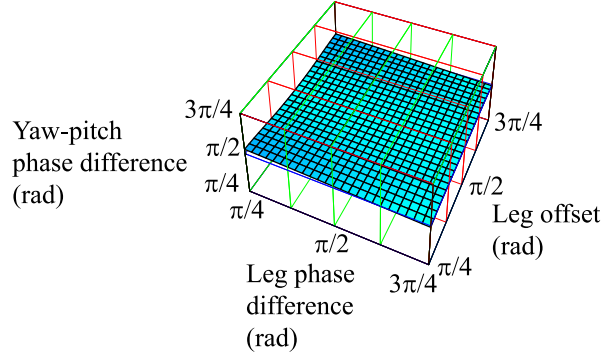


Figure 3.34: RRRobot pitch-yaw phase difference as a function of leg trajectory offset and phase difference.

3.4.3 Summary of RRRobot Experiments and Simulations

In this section, we have explored legless locomotion through experiments and simulation. Using sinusoidal leg trajectories that induce variable-curvature translation, we have shown that RRRobot has full planar accessibility. The simulation and experiment results match qualitatively, the main difference being the slow linear translation velocity in experiment compared with simulation. This may be because of the unmodeled body-ground dry friction.

The intuition we have gained from the contact kinematics analysis in section 3.3 has allowed us to quantify RRRobot's planar motion. But while we have presented methods to demonstrate legless locomotion, many difficult questions arise from the simultaneous interaction of complex phenomena, such as the variable inertia rotational dynamics and the nonholonomic contact constraints. For example, why does RRRobot's body yaw? Can we find RRRobot's leg motions to track a specific planar path? We now describe simplified models that provide insight into legless locomotion's dynamics.

3.5 Simplified Legless Locomotion Dynamics Models

The key to understanding a complex mechanical system is to find simple models that capture the essence of its motion. We then can use the simplified models to understand the contribution of various elements to the system's properties. In this section, we present three types of simplifications to understand legless locomotion as demonstrated by RRRobot:

1. Decoupling the system's internal dynamics and external contact kinematics, analyzing their

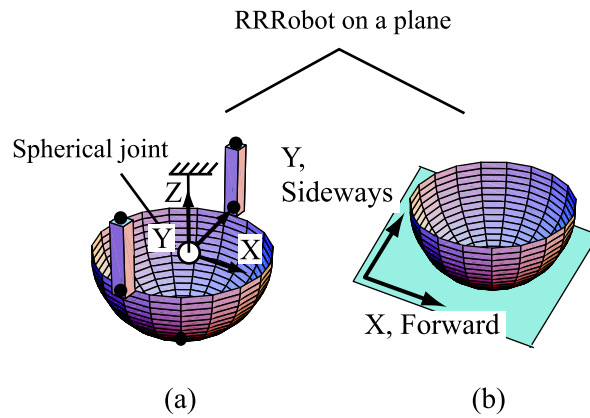


Figure 3.35: The Pivoting Dynamics model simplifies the RRRobot model (see Figure 3.1) into two parts: (a) RRRobot pivoted at its geometric center on a spherical joint and (b) a sphere on a plane.

individual properties, and then recombining them. We call this collection of models the Pivoting Dynamics model.

2. Studying the system dynamics along non-actuated freedoms of the system separately and recombining the individual motions using the contact kinematics. We call this collection of models the single-axis models.
3. Exploring if the system can be modeled as a drift-free kinematic system with velocity inputs rather than a dynamic system with drift and force/torque controls. Such a model if it exists is called a kinematic reduction.

The first type of simplification, decoupling the system dynamics and external contact kinematics, helps understand the individual influence of the dynamics and the kinematics on the robot's motion. RRRobot's motion structure lends itself to such an analysis, and we explore the leg-body rotation dynamics separately from the sphere-ground contact kinematics. We achieve this by pivoting the robot at its geometric center, allowing isolated study of the leg-body rotational dynamics since the sphere has no ground contact. We then use the sphere-plane contact kinematics (discussed in section 3.3) to compute the motion produced by the body rotations. We call this model the *Pivoting Dynamics* model (see Fig. 3.35 and section 3.5.1). This simplification assumes that the dynamics and kinematics are decoupled, and we discuss the implications.

The second type of simplification, analyzing the dynamics along the body's non-actuated freedoms separately, allows us to focus on one specific freedom by disabling the remaining non-actuated freedoms. This reduction in the body's freedoms will help understand the coupling (or lack of it)

between the different robot freedoms. For example, consider a satellite in space with three perpendicular reaction wheels. Disabling the satellite's body roll and pitch freedoms allows us to explore the influence of the reaction wheel motions on the yaw rotations. If we understand the influence of controls on the various passive freedoms individually, a superposition of the individual motions allows us to approximate the robot's motion. In the case of RRRobot, we study the influence of leg motions on body rotations along the roll, pitch, and yaw axes; we then use the sphere-plane contact kinematics to compute robot translation (see section 3.5.2). This simplification assumes that the individual freedoms are decoupled, and we discuss the implications.

The third type of simplification, finding kinematic reductions, is based on techniques developed by Bullo, Lewis, and Lynch [23]. It involves identifying if the dynamic system with acceleration inputs and drift can be modeled as a driftless kinematic system with velocity inputs. This is useful because control and planning is easier for kinematic systems than for dynamic systems. For example, planning and control is easier for a vertical unicycle when it is viewed as a kinematic system rather than as a dynamic system. But only some dynamic systems that satisfy certain properties admit kinematic reductions. Section 3.5.2 explores kinematic reductions for RRRobot's single-axis models.

Note that the first two approaches are approximate simplifications and are straight-forward to implement, while the third approach is an exact simplification and difficult to derive. The key reason for exploring these simplified models is that understanding RRRobot's motion and finding a control method using its full dynamics is difficult; so we approximate the full model using simpler models, quantify the robot's motion in a decoupled manner, and find control strategies for the simpler models. We then show that the control strategies for the simpler models help develop qualitative control the full dynamics model. RRRobot's motion structure lends itself nicely to such a decoupled analysis, and it is unclear if such analysis is possible for other systems. We now discuss the three simplifications in greater detail.

3.5.1 Pivoting Dynamics Model

The RRRobot dynamics model presented in section 3.2 includes the interplay between body dynamics and contact kinematics, and RRRobot's equations of motion are provided by (3.2). It is difficult to understand the contribution of the dynamics and the kinematics to RRRobot's complex motion. Section 3.3 explores the contact kinematics, by studying RRRobot's translation as a function of body rotations while ignoring the leg-body dynamics. To analyze just the interaction between leg motion and body attitude, we pivot RRRobot at its geometric center, a simplification since we ignore

the influence of RRRobot's planar translation on the dynamics. Once we compute the body attitude motion for a certain leg trajectory, we use the contact kinematics equations to approximately predict RRRobot's translation in the plane. Thus, this model, called the Pivoting Dynamics model, approximately *reduces* the RRRobot system into two parts (see Fig. 3.35): 1) the dynamics of RRRobot rotating about a spherical joint, and 2) the contact kinematics of a sphere on the plane. We now present a mathematical model for the Pivoting Dynamics model.

The configuration q_{pd} of the Pivoting Dynamics model consists of the sphere's orientation $R(\theta_y, \theta_p, \theta_r)$ with respect to a inertial frame and the configuration of its legs (ϕ_1, ϕ_2) . We use the body-fixed yaw-pitch-roll Euler angles to represent robot orientation. Thus,

$$q_{pd} = (R(\theta_y, \theta_p, \theta_r), \phi_1, \phi_2)^T \in SO(3) \times \mathbb{R}^2. \quad (3.10)$$

The equations of motion for the Pivoting Dynamics model take the form

$$M_{pd}(q_{pd})\ddot{q}_{pd} + C(q_{pd}, \dot{q}_{pd})\dot{q}_{pd} + G(q_{pd}) = \tau_{pd} + \zeta_{pd}, \quad (3.11)$$

where $M_{pd}(q_{pd}) \in \mathbb{R}^{5 \times 5}$ represents the positive-definite non-diagonal variable mass matrix, $C(q_{pd}, \dot{q}_{pd}) \in \mathbb{R}^5$ represents the vector of Coriolis and centrifugal terms, $G(q_{pd}) \in \mathbb{R}^5$ represents the vector of gravitational terms, $\tau_{pd} = (0, 0, 0, \tau_1, \tau_2)^T$ represents the generalized force, and $\zeta_{pd} \in \mathbb{R}^5$ represents the viscous damping to model any losses. The generalized force τ_{pd} indicates that only the legs are actuated, and there are no external constraints on the system. Note that (3.11) does not include the influence of the contact kinematics and differs from RRRobot's dynamics modeled in (3.2).

Once we compute the changes in body configuration for a certain leg trajectory, we use the kinematic contact equations

$$\begin{pmatrix} \omega^1 \\ \omega^2 \end{pmatrix} \dot{q}_{rr} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (3.12)$$

to compute the velocity of the contact point in the plane, where

$$\begin{aligned} \omega^1 &= (1, 0, 0, -r \cos \theta_y, -r \cos \theta_p \sin \theta_y, 0, 0), \\ \omega^2 &= (0, 1, 0, -r \sin \theta_y, r \cos \theta_r \cos \theta_y, 0, 0), \end{aligned} \quad (3.13)$$

and $q_{rr} = (x, y, q_{pd}^T)^T$ represents the configuration of RRRobot on a plane. While the Pivoting Dynamics model is fictitious, we can use it to approximate RRRobot's motion.

Why do we say “approximate”? What is the difference between the Pivoting Dynamics model and RRRobot? The main difference between the Pivoting Dynamics model and the RRRobot is the rotational axes's location, arising from the rolling contact in the full dynamics model and the spherical joint (at the sphere center) in the Pivoting Dynamics. RRRobot's center of mass is oscillating about the moving contact point, whereas the Pivoting Dynamics model's center of mass is oscillating about the sphere's fixed geometric center. Thus, if we consider just one axis of rotation for the body, RRRobot behaves like an inverted (rolling) pendulum (see Fig. 3.36), and the Pivoting Dynamics model behaves like a simple pendulum (see Fig. 3.37). The different rotational axes result in different effective rotational inertias and, consequently, different rotational time periods, radii of gyration, and oscillation amplitudes. In particular, the rolling inverted pendulum's oscillation time-period for small amplitudes is

$$T_{ip} = 2\pi\sqrt{\frac{\rho^2}{g(r-\rho)}}, \quad (3.14)$$

where ρ is the radius of gyration, and g is gravity, while the simple-pendulum oscillation time-period is

$$T_{sp} = 2\pi\sqrt{\frac{\rho}{g}}. \quad (3.15)$$

Table 3.1 compares the time-periods for RRRobot and the Pivoting Dynamics model.

As a result of these inertia differences, RRRobot and the Pivoting Dynamics model have different translation and yaw rates for the same leg motions. Thus, the Pivoting Dynamics model can only qualitatively approximate RRRobot's translation.

So what is the advantage in using the Pivoting Dynamics model? The Pivoting Dynamics model approximately reduces RRRobot's seven second-order equations to five second-order equations and two first-order equations, a slightly simpler system. Furthermore, the Pivoting Dynamics model allows us to quantify the influence of the dynamics and kinematics on RRRobot's translation and also investigate the influence of system inertia on motion.

Figs. 3.38 and 3.39 compare RRRobot's motion in simulation with the motion predicted by the Pivoting Dynamics models. The translation produced in the Pivoting Dynamics model and in the RRRobot model match qualitatively; the contact point follows similar paths, but the Pivoting Dynamics model translates slightly faster and curves sharper. This is because the Pivoting Dynamics

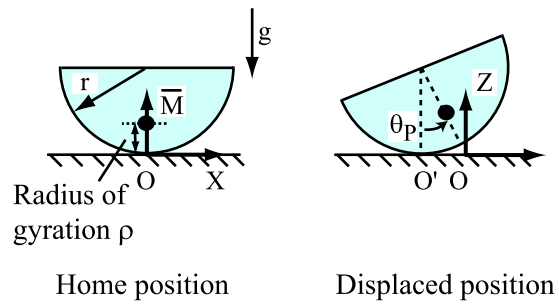


Figure 3.36: A planar eccentric-mass wheel performs harmonic oscillations for small amplitude.

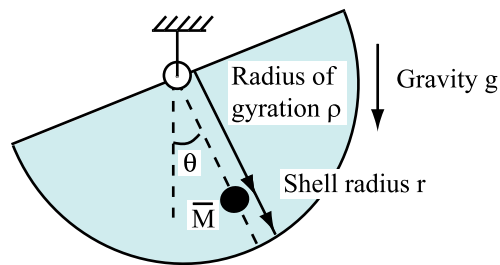


Figure 3.37: The simple pendulum performs harmonic oscillations for small amplitude.

Table 3.1: Rotation Time-Periods for the RRRobot-on-a-plane model and the Pivoting Dynamics model

	Roll Rotations (sec)	Pitch Rotations (sec)
RRRobot-on-a-plane	1.29	1.07
Pivoting Dynamics	1.19	0.96

Model is pivoted at its geometric center, while in the RRRobot-on-a-plane Model, the robot has a rolling contact. Thus, for a given change in attitude, the contact point moves faster in the Pivoting Dynamics model than in the RRRobot-in-a-plane Model. In summary, we can use the Pivoting Dynamics model to approximate RRRobot’s planar translation.

Our Pivoting Dynamics simulations use the damping parameters

$$\zeta_{pd} = (-0.01\dot{\theta}_r, 0, -0.01\dot{\theta}_p, -0.01\dot{\phi}_1, -0.01\dot{\phi}_2)^T, \quad (3.16)$$

and in our RRRobot simulations, we use the damping parameters

$$\zeta_{rr} = (0, 0, -0.01\dot{\theta}_r, -0.01\dot{\theta}_y, -0.01\dot{\theta}_p, -0.01\dot{\phi}_1, -0.01\dot{\phi}_2)^T. \quad (3.17)$$

We use different yaw damping values, because yaw damping destroys any net yaw produced by leg motions in the Pivoting Dynamics model.

The difficulty with the Pivoting Dynamics model is that the body rotations and leg motions are still coupled and (3.11) is complex. We now discuss our second approach to simplifying RRRobot’s motion: analyzing its dynamics along each unactuated freedom separately.

3.5.2 Single-Axis-Rotation Models

RRRobot’s two forced oscillators—the roll oscillator and the pitch oscillator—and its yaw freedom are simultaneously controlled by RRRobot’s leg motions. These body rotations are coupled through the body-ground rolling contact, making dynamics analysis difficult. While the Pivoting Dynamics model (see section 3.5.1) decouples RRRobot’s constrained rotational dynamics from the contact kinematics, the Pivoting Dynamics model’s rotational dynamics about the spherical joint is itself hard to analyze because of body roll-pitch-yaw coupling.

We propose decoupling the body rotational dynamics and studying the relationship between leg motions and body motions along each axis separately (see Figs. 3.40, 3.41, and 3.42). For example,

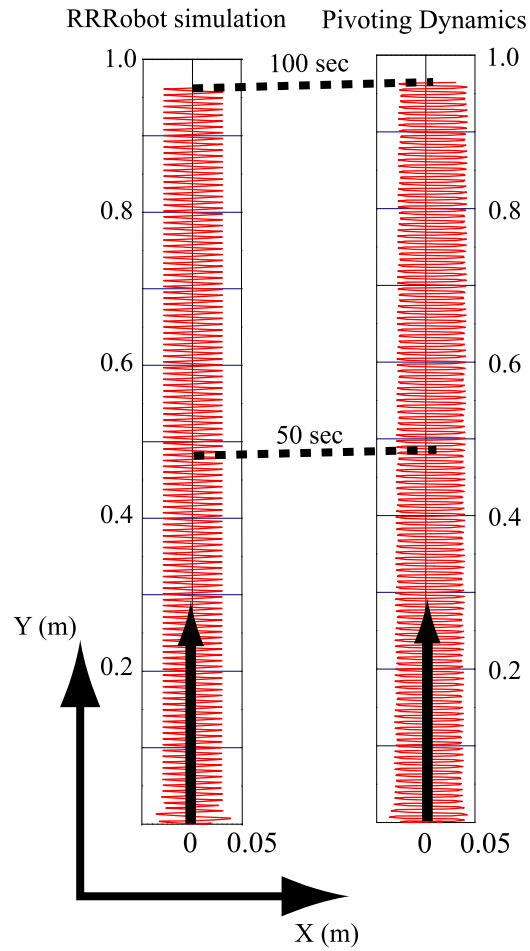


Figure 3.38: Planar plots of contact-point time history during sideways locomotion produced by Gait 1 in RRRobot-on-a-plane simulation, RRRobot-on-a-plane experiment, and Pivoting Dynamics simulation. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.

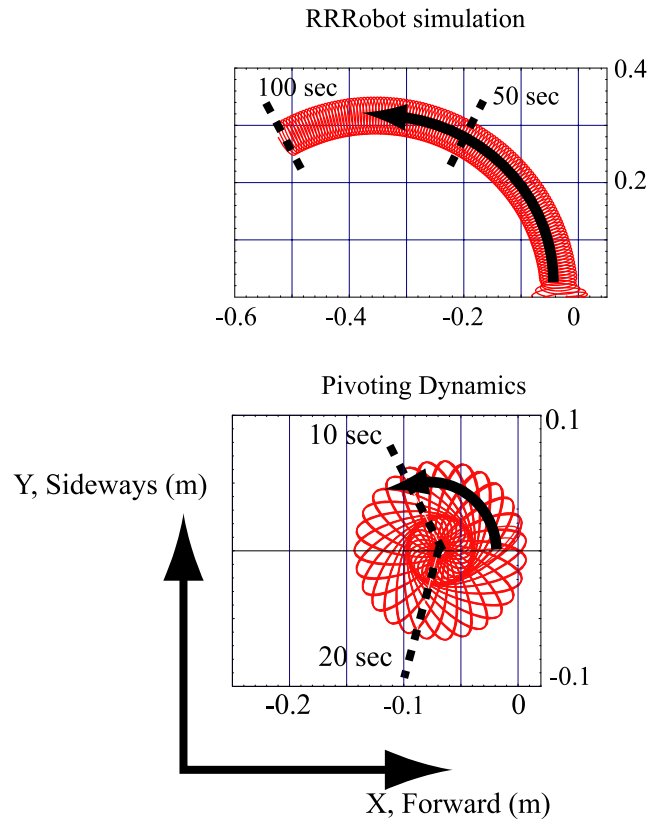


Figure 3.39: Planar plots of contact-point time history during counter-clockwise circular locomotion produced by Gait 2 in RRRobot-on-a-plane simulation, RRRobot-on-a-plane experiment, and Pivoting Dynamics simulation. The solid arrow gives robot motion direction, and the dotted lines indicate the robot position at the specified time.

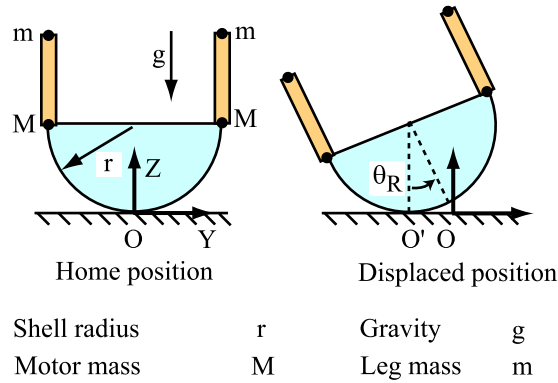


Figure 3.40: RRRobot's roll freedom (side view).

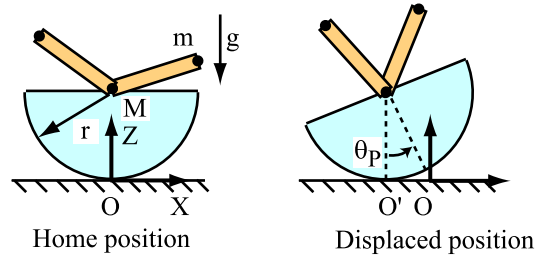


Figure 3.41: RRRobot's pitch freedom (side view).

we create the Pitch model by disabling RRRobot's roll and yaw rotations and allowing only pitch rotations. This allows us to study RRRobot's body pitch motion independent of the other rotational freedoms. Similarly, we create the Roll model by allowing only body roll rotations and the Yaw model by allowing only body yaw rotations. We call this collection of models the Single-Axis models. Note that the roll and pitch models have a rolling contact, while the Yaw model is pivoted. This contrasts with the Pivoting Dynamics model where the body rotates about a spherical joint at the geometric center.

The decoupled oscillatory dynamics models approximate RRRobot's dynamics assuming zero body roll-pitch-yaw coupling (see Fig. 3.43). We will highlight where and why this assumption breaks after discussing the single-axis models's dynamics.

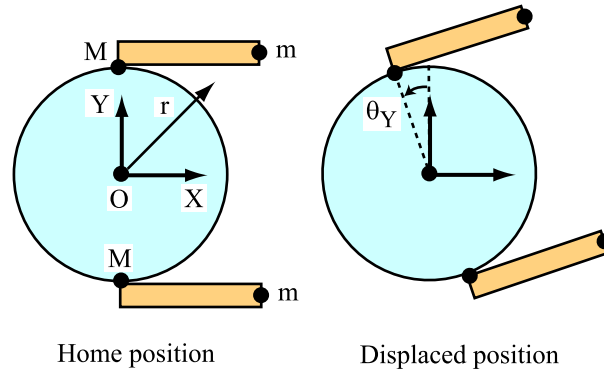


Figure 3.42: RRRobot's yaw freedom (top view).

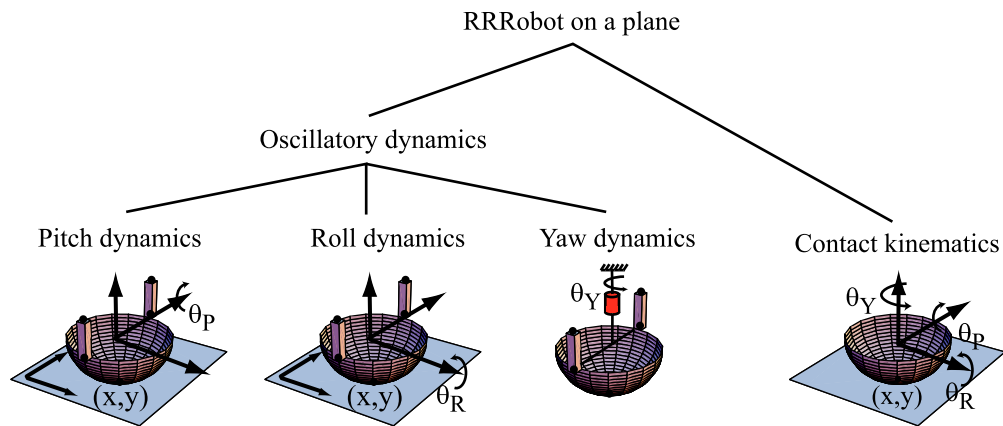


Figure 3.43: Decoupled RRRobot dynamics.

The equations of motion for these fictitious single-axis models take the form

$$M(q_{sa})\ddot{q}_{sa} + C(q_{sa}, \dot{q}_{sa})\dot{q}_{sa} + G(q_{sa}) = \tau_{sa} + \zeta_{sa}, \quad (3.18)$$

where $q_{sa} = \{\theta, \phi_1, \phi_2\} \in \mathbb{R}^3$, and ζ_{sa} represents damping.

The first element of q_{sa} is θ , the body roll, pitch, or yaw configuration depending on the model, while the last two elements represent leg configuration. The symbols $M(q_{sa}) \in \mathbb{R}^{3 \times 3}$, $C(q_{sa}, \dot{q}_{sa})\dot{q}_{sa} \in \mathbb{R}^3$, and $G(q_{sa}) \in \mathbb{R}^3$ represent standard mechanical-system terms, and $\tau_{sa} = (0, \tau_1, \tau_2)^T$ is the generalized force. The input torques τ_1 and τ_2 are applied to the legs, while body rotation is not actuated. We use the sphere-plane contact kinematics given by (3.12) to approximately compute RRRobot's translation for the body rotations induced in the decoupled models.

Fig. 3.44 shows one example of the strong match between translation predicted by the single-axis models and RRRobot's translation for small amplitude $\pi/2$ out-of-phase leg oscillations about the vertical. These leg motions produce body pitch and yaw oscillations about zero, while roll rotation is negligible. There is a strong match between the body rotation trajectories for the decoupled models and the full dynamics models, indicating that pitch and yaw oscillations are decoupled.

Fig. 3.45 shows one example of the strong match between translation predicted by the single-axis models and RRRobot's translation for small amplitude out-of-phase leg oscillations offset $\pi/4$ from the vertical. These leg motions produce pitch and yaw body oscillations primarily and small roll oscillations. In addition, the robot body leans from the vertical, and each leg cycle produces net body yaw in both the full dynamics model and the decoupled dynamics model.

But there is a difference between the single-axis models and the full-dynamics model: the yaw inertia in the full dynamics model is a function of body pitch (due to the rolling contact) and leg configuration, while the yaw inertia in the decoupled Yaw model is only a function of leg configuration (the yaw pivot prevents body pitch). This causes the yaw drift rates in the full-dynamics model to be different from the yaw drift rates predicted by the Yaw model for different leg trajectories. If we want to use the decoupled dynamics models to approximate RRRobot's motion, some adjustment is required to match the yaw drift between the Yaw model and the full dynamics model. In our work, we vary leg amplitude as a function of leg offset for the decoupled Yaw model to make the yaw drift rate match with RRRobot's dynamics. This approximation allows us to model RRRobot's translation with the decoupled models, and we discuss this aspect in the Yaw model and the legless locomotion control subsection. Note that we use the following damping parameters in

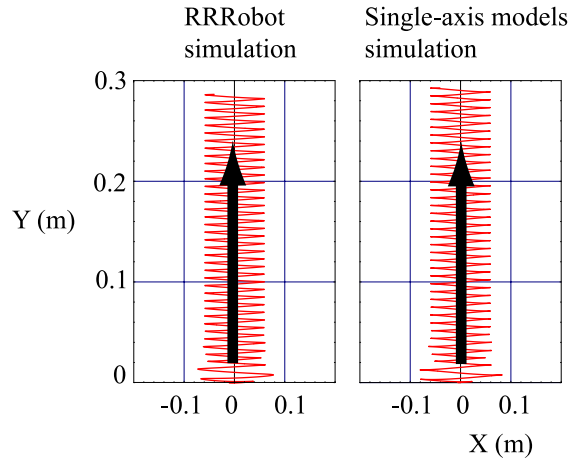


Figure 3.44: The lateral translation gait: comparison of RRRobot's motion with motion predicted by the single-axis models over thirty seconds. Leg 1 trajectory: $\pi/2 + 0.3 \sin(8t)$, and leg 2 trajectory: $\pi/2 + 0.3 \cos(8t)$.

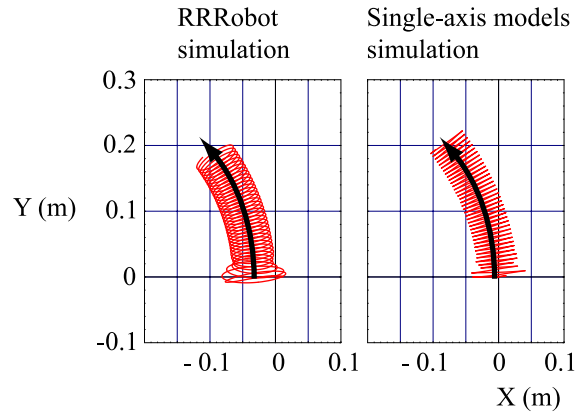


Figure 3.45: The circular translation gait: comparison of RRRobot's motion with motion predicted by the single-axis models over thirty seconds. Leg 1 trajectory: $\pi/4 + 0.3 \sin(8t)$, and leg 2 trajectory: $\pi/4 + 0.3 \cos(8t)$.

our Single-Axis simulations:

$$\zeta_P = (-0.01\dot{\theta}_p, 0, 0)^T, \zeta_R = (-0.01\dot{\theta}_r, 0, 0)^T, \zeta_Y = (0, 0, 0)^T. \quad (3.19)$$

We now discuss the dynamics of the Pitch model.

The Pitch Model

The Pitch model is derived by restricting RRRobot's body rotational freedoms to only the pitch freedom (see Fig. 3.41), with the goal of using a simpler model to capture RRRobot's body pitch rotations. The Pitch model behaves as a forced inverted pendulum—the robot body oscillates about a mean pitch configuration depending on the choice of leg motions and the natural oscillatory dynamics and settles into a limit cycle due to frictional damping. Note that the Pitch model better represents RRRobot's pitch motion than the Pivoting Dynamics models, since the rolling contact is retained.

When viewed as a control system, the pitch model's inputs are the leg torques and the outputs are the body oscillation amplitude, frequency, phase, and offset. Note that body pitch oscillation frequency equals the leg frequency, since the dynamics is approximately linear for small leg trajectory amplitudes, and the mean body pitch offset may be determined by a statics analysis.

Fig. 3.46 shows how body pitch oscillation amplitude relates to leg trajectory controls. Comparing with Fig. 3.31, we note that the Pitch model captures RRRobot's pitch-oscillation amplitudes well. The remaining parameter, the body pitch phase, is not important in an absolute sense; rather the pitch phase value relative to the yaw phase value is important, since the relative phase influences RRRobot's translation (as discussed in section 3.3). We discuss the relation between body pitch and yaw phase in the Yaw model subsection.

The Yaw model is more complex than the Pitch model, since the leg motions induce both net body yaw in addition to body yaw oscillation. We now discuss the Yaw model in greater detail.

The Yaw Model

The Yaw model is derived by pivoting RRRobot at a revolute joint which is placed at the sphere's geometric center and whose axis is aligned with the body Z-axis (see Figs. 3.42 and 3.47). The Yaw model's motion helps us understand RRRobot's yaw rotations. The yaw model is similar to the Pivoting Dynamics model, except that the body can spin about the yaw axis only.

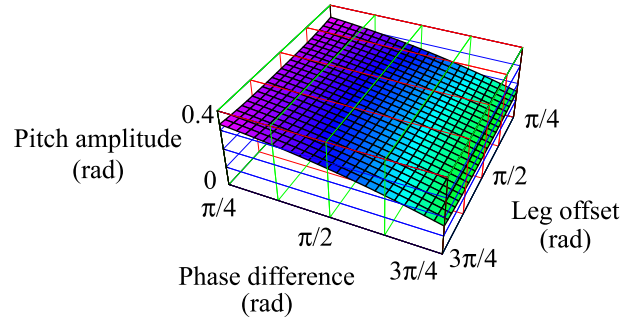


Figure 3.46: The Pitch model: Variation in body pitch oscillation amplitude as a function of leg offset and leg phase difference.

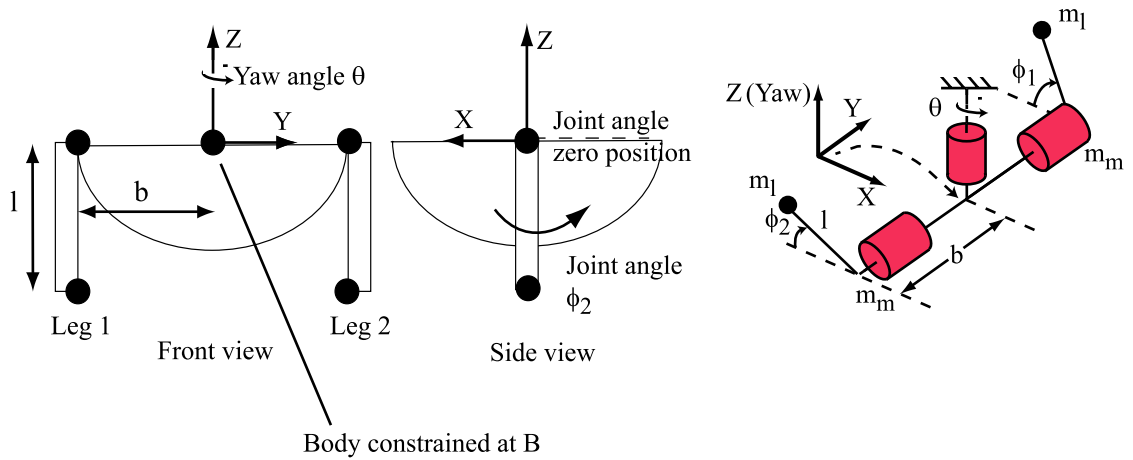


Figure 3.47: The Yaw model: the body, pivoted at its body center, can freely rotate about the yaw axis, and the two legs with point masses at the distal ends are singly actuated.

The Yaw model body has two masses, each m_m , at the ends of a diameter. Each massless leg has an actuated hip joint and a point mass m_l at the distal end. The Yaw model configuration is represented by $q_y = (\theta_y, \phi_1, \phi_2)^T \in \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$, where θ_y denotes the body configuration, ϕ_1 leg 1's joint configuration, and ϕ_2 leg 2's joint configuration.

The Yaw model has no gravity, there are no joint limits, and torques u_1 and u_2 can be applied at leg joints 1 and 2. The mass matrix $M_y(q_y)$ associated with the Yaw model and describing the system kinetic energy is

$$M_y(q_y) = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix}, \quad (3.20)$$

where

$$\begin{aligned} g_{11} &= 2(m_m + m_l)b^2 + m_l l^2 + \frac{1}{2}m_l l^2(\cos 2\phi_1 + \cos 2\phi_2), \\ g_{12} &= -m_l l b \sin \phi_1, \\ g_{13} &= m_l l b \sin \phi_2, \\ g_{21} &= -m_l l b \sin \phi_1, \\ g_{22} &= m_l l^2, \\ g_{23} &= 0, \\ g_{31} &= m_l l b \sin \phi_2, \\ g_{32} &= 0, \\ g_{33} &= m_l l^2. \end{aligned}$$

Note that the mass matrix $M_y(q_y)$ depends on leg configurations, but is independent of yaw rotations. Such an invariance is called a symmetry, implying the existence of a conserved quantity [25] in the Yaw model. In the Yaw model, this conserved quantity is the yaw angular momentum; in the absence of external disturbances, the total angular momentum of the body and the legs about the Z-axis is constant.

The Yaw model equations of motion [8] are given by

$$M_y(q_y)\ddot{q}_y + C_y(q_y, \dot{q}_y)\dot{q}_y = \tau_y, \quad (3.21)$$

$\tau_y = (0, \tau_1, \tau_2)^T$ is the control and $C_y(q_y, \dot{q}_y)$ contains velocity products. The control τ_y indicates that the Yaw model is underactuated. Also, if the system's initial velocity \dot{q}_y is zero, then the body must be stationary when the legs are stationary (along any trajectory).

Table 3.2: Incremental motion of the yaw model.

Time interval	$\phi_1(t)$	$\phi_2(t)$	Change in yaw
0–1	$0 \rightarrow \pi/2$	0	ϵ_1
1–2	$\pi/2$	$0 \rightarrow \pi/2$	$-\epsilon_2$
2–3	$\pi/2 \rightarrow 0$	$\pi/2$	$-\epsilon_2$
3–4	0	$\pi/2 \rightarrow 0$	ϵ_1
Net change in yaw			$2(\epsilon_1 - \epsilon_2)$

The key question with the underactuated yaw model is whether the body can reach arbitrary configuration using leg motions. It is apparent from the angular momentum conservation principle that the body yaws from the reaction forces of leg swing. For example, if we move the left leg forward from the vertical configuration while keeping the other leg stationary, the body spins clockwise instantaneously. However, if the leg makes a complete rotation, the body returns to its start configuration.

Can we get net body yaw using cyclic body motions? That is, if we move the legs and return them to the start configuration, can we achieve net body yaw motion? An important property of the Yaw model that we can use to produce net body yaw is the variable inertia.

Here is a simple thought experiment to illustrate this. We will move each leg back and forth between extremes of 0 and $\pi/2$. Each leg will dwell at the extreme for one second and will take one second to transition between angles following a cubic spline. The result is a Lie bracket-inspired [57] smoothed square wave, with the two legs out-of-phase with each other (see Fig. 3.48). This sequence of leg motions yields a net yaw motion, as shown in Table 3.2 and can be confirmed by studying the table and thinking about the yaw angular inertia of the system. Suppose the body yaw is ϵ_1 during interval $t = 0$ to $t = 1$, and is ϵ_2 during interval $t = 1$ to $t = 2$. The net yaw during the two motion segments is different, because the yaw angular inertia varies depending on whether the leg is stretched out or tucked in. This difference produces net yaw at the end of the motion sequence. This same property of producing net yaw motion using yaw inertia differences is seen in RRRobot also, but RRRobot is more complex because of the coupling between body pitch and yaw oscillations.

We now discuss control for the Yaw model; that is, finding leg motions (gaits) that allow the robot to reach any configuration.

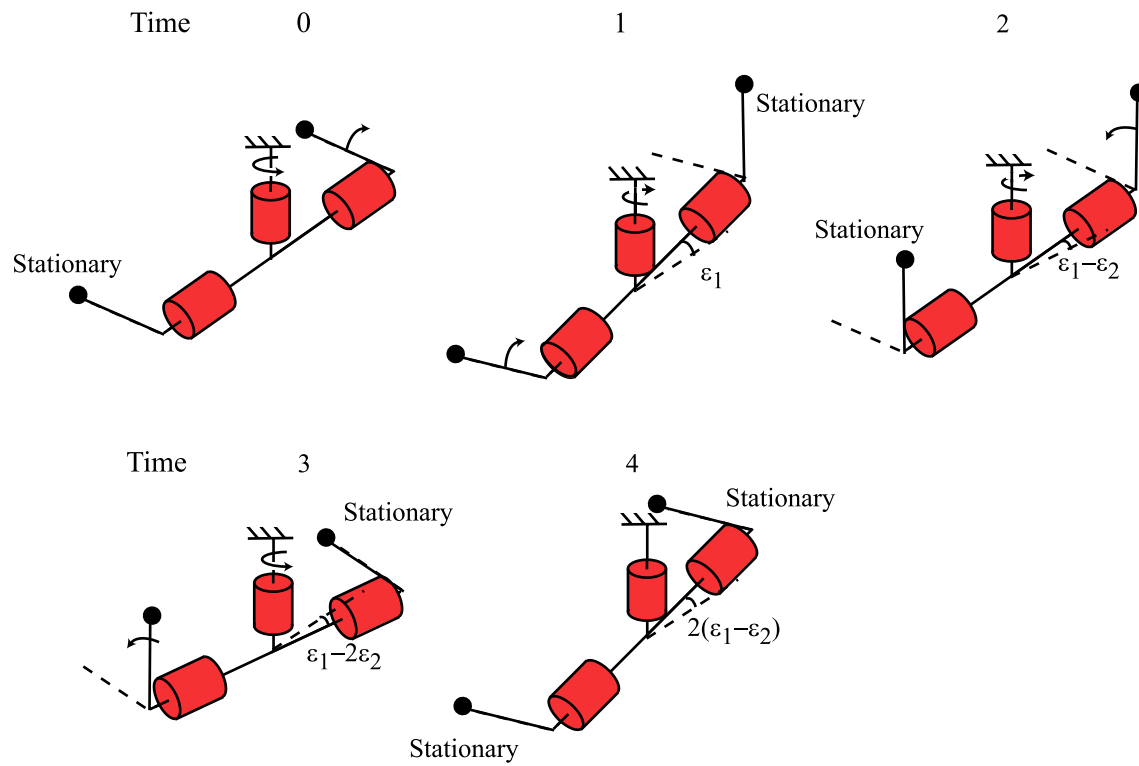


Figure 3.48: Incremental motion of the yaw model using Lie bracket-inspired leg motions (see Table 3.2).

Control for the Yaw Model

Finding characteristics like that shown in Table 3.2 for mechanical systems from equations of motion like (3.21) is difficult. Also, planning and controlling system trajectories using (3.21) is difficult, because of the velocity-related terms and the torque inputs; that is, there is no systematic analytic procedure to find torque inputs to achieve a given goal trajectory.

In [12], we present a *kinematic reduction* for the yaw model, derived from the angular momentum conservation principle. A kinematic reduction model is a kinematic version of the full dynamic system. The primary condition for a kinematic model to become a kinematic reduction of a mechanical system is that there must exist controls for the dynamic model that can track the kinematic model's trajectory.

Using techniques in [23], we find two gaits for the yaw model that allow the kinematic model full configuration controllability (the ability to reach any configuration at rest), while ensuring that the trajectories can be tracked by the mechanical system. The first gait involves moving one leg while keeping the other leg stationary and produces net yaw for acyclic leg motions. The second gait involves moving both legs in out-of-phase sinusoids. The right phase relationship and leg offset produces net yaw. Thus, if we want to move the yaw model from one configuration to another, we apply the second gait followed by applying the first gait to both legs.

The interesting gait is the second gait, since it produces body yaw oscillations and net yaw using cyclic leg motions. Fig. 3.49 shows how body yaw oscillation amplitude relates to leg oscillatory trajectories for the yaw model. This compares favorably with RRRobot's body yaw oscillation amplitudes shown in Fig. 3.33, except for the spike near leg offset $\pi/2$ in the yaw model. The absence of a spike in the full dynamics models is attributed to the pitch-yaw coupling—as the leg offset shifts from the vertical ($\pi/2$), the robot pitches from the vertical. This causes the yaw inertia about the rolling contact to increase, since the battery mass is offset from the axis, and consequently, produces smaller yaw oscillations.

In contrast, since the Yaw model's body pitch configuration is fixed, the Yaw model's yaw inertia depends only on leg configuration. Hence, there is a spike in the body yaw oscillation amplitude mapping. This effect carries over to the relationship between net body yaw induced and the leg trajectories also. For example, the leg trajectories that produce maximum net body yaw is different for the full dynamics model and the Yaw model.

However, techniques developed by Shamma et al. [72] allow us to compute net body yaw in the Yaw model for different leg trajectories (see Fig. 3.50). Cyclic leg motions about the vertical configuration produce zero net body yaw, while cyclic leg motions about leg configurations offset

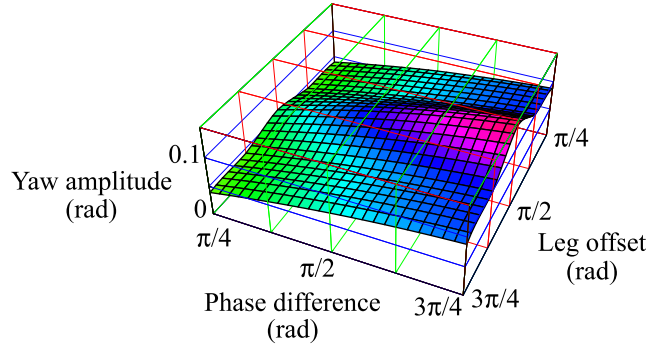


Figure 3.49: The Yaw Model: Variation in body yaw oscillation amplitude as a function of leg offset and leg phase difference.

from the vertical produce net body yaw. The height functions in Fig. 3.50 allow us to compute the net body yaw induced in the Yaw model for any leg trajectory, and any difference between the Yaw model and the full dynamics model can be adjusted for, say, using the leg amplitude changes.

Even though we have found kinematic reductions for the yaw model, it does not extend directly to RRRobot, because of RRRobot's gravitational drift as well as the coupling between the body pitch and yaw rotations. In particular, the leg cycles that produce maximum body yaw motion is different in the two systems. But we can still use the kinematic reduction for the yaw model as an approximate model of RRRobot's yaw orientation by making the leg amplitude a function of leg offset in the yaw model (see section 3.6 for more details).

Also, while the phase relationship between body pitch and yaw oscillations predicted by the single-axis models is different from the phase differences in the full dynamics model (see Fig. 3.51 and compare with Fig. 3.34), the relative phase difference between body pitch and yaw oscillations only influences translation velocity and not curvature. Since RRRobot's velocity is small, this discrepancy does not impact control significantly if we track translation curvature only. Furthermore, the difference in linear velocities may be corrected using feedback. Finally, while we have only discussed kinematic reduction results for the simple Yaw model, finding kinematic reductions for complex systems such as the legless locomoting RRRobot is an open problem.

3.5.3 Summary

We now summarize the utility of using the simplified models discussed in this section.

First, we developed the Pivoting Dynamics model by decoupling the body-rotation dynamics

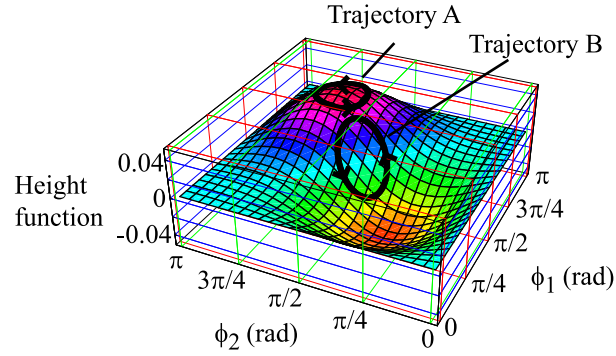


Figure 3.50: Yaw model height function. Trajectory A (leg 1: $5\pi/8 + 0.15 + 0.3 \sin(8t)$ and leg 2: $5\pi/8 + 0.15 + 0.3 \cos(8t)$) produces net body yaw, while trajectory B (leg 1: $\pi/2 + 0.3 \sin(8t)$ and leg 2: $\pi/2 + 0.3 \cos(8t)$) does not produce net yaw.

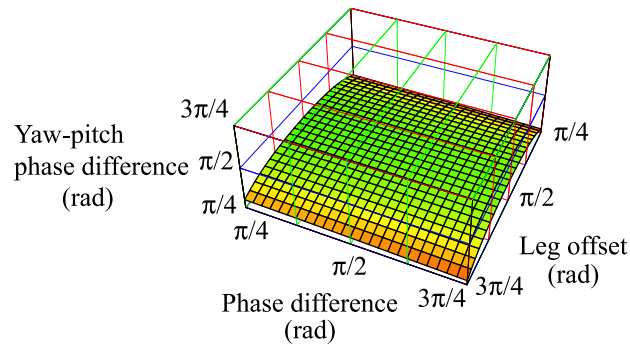


Figure 3.51: RRRobot yaw-pitch phase difference as a function of leg trajectory offset and phase difference, as predicted by the single-axis models.

from the contact kinematics, allowing us to study the dynamics and the contact kinematics separately. This allows us to quantify the dynamic relationship between leg motions and body rotations and the kinematic relationship between body motions and robot translation separately. We recombine these decoupled models to approximate RRRobot’s motion qualitatively. This simplification is enough to isolate the contact kinematics, but the rotational dynamics of a spherical-joint pivoted body is still complex. Furthermore, the pivoted body rotations do not accurately represent RRRobot’s body rotations, which utilizes a rolling contact.

So, we then consider the single-axis models to quantify the dynamics of body rotations along each freedom individually. This allows us to study the relationship between leg motions and body roll, pitch, and yaw rotations separately. This gives us significant insights into the body pitch oscillation amplitudes, frequency, phase, and offset, and the body yaw oscillations and drift. Recombining the single-axis models with the contact kinematics provides an approximation to RRRobot’s dynamics. We now present a control strategy using the single-axis models and the contact-kinematics model.

3.6 Toward Legless Locomotion Control

Section 3.1 introduced legless locomotion and section 3.5.2 introduced the simplified (decoupled) models that provide insights into legless locomotion’s dynamics. Furthermore, we showed using simulation that the simplified models provide a good approximation to RRRobot’s locomotion. In this section, we find an approximate control solution for RRRobot—a mapping between planar translation and leg motions—using these models.

Legless locomotion has properties that make control difficult, namely underactuation (two controls and seven degrees of freedom), a configuration-dependent inertia, velocity-related and gravitational drift, and contact constraints. Furthermore, legless locomotion has a dynamics structure that is difficult to integrate symbolically even for one specific input. So, in this section, we numerically solve a reduced problem—finding the leg motions that produce the desired robot velocity and fixed-curvature path when the robot settles into its limit (steady state) oscillatory cycle. In section 3.6.2, we suggest an approach to the general legless-locomotion control problem of tracking a variable curvature path.

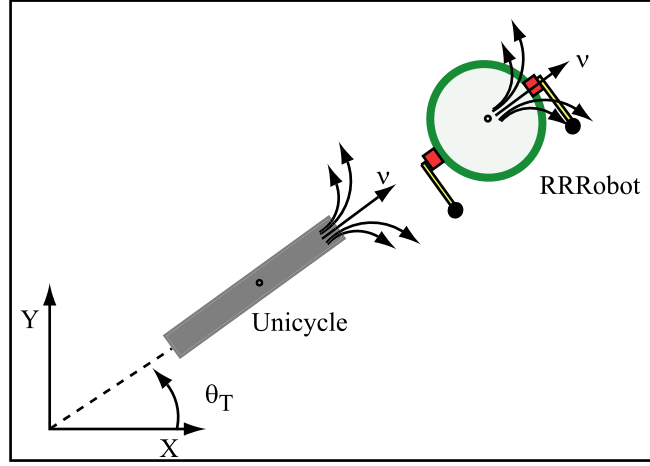


Figure 3.52: Similarity in planar translation between a vertical unicycle and RRRobot (top view).

3.6.1 Legless locomotion control

In this subsection, we present RRRobot control, that is, a mapping from planar translation to leg motions. We also show that this control mapping is qualitatively similar for the decoupled models and the full dynamics models. The key idea is that we can use the decoupled models to predict motion in the full-dynamics model and also develop more advanced control methods in the future.

An analysis of RRRobot's planar motion (see Fig. 3.30) shows that the predominant translation mode is translation along the body-fixed Y-axis with variable curvature. Small velocity variations are also possible. Such motion is similar to a unicycle with a limited velocity range and turning radius (see Fig. 3.52). As discussed in section 3.1, RRRobot's motion results from the limited body rotational dynamics that the leg motions can produce. The leg motions produce body pitch, yaw, and (small) roll rotations using different leg offsets and phase differences. Inertial differences during out-of-phase leg motions produce body yaw drift, which results in translation curvature.

Figs. 3.53 and 3.54 show how translation velocity v , yaw velocity α (the turning rate), and curvature $K = \alpha/v$ depend on leg offset and phase difference (for sinusoidal leg motions) in the full RRRobot dynamics model and the decoupled dynamics models (using leg amplitude 0.3 rad (see caveat below), angular frequency 8 rad/s, and measured at the mean of a cycle). The magnitudes and structure of yaw velocity and curvature match well, but there is a structural difference in the linear-velocity mapping.

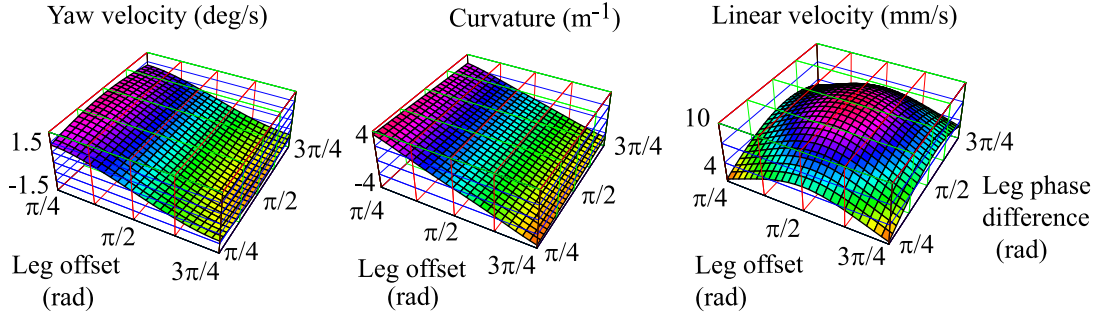


Figure 3.53: RRRobot translation as a function of offset and leg phase difference.

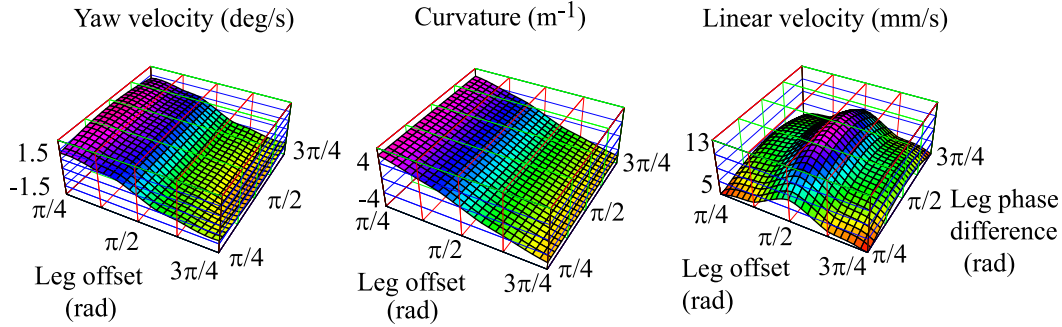


Figure 3.54: RRRobot translation as a function of offset and leg phase difference as predicted using the decoupled models (compare with Fig. 3.53)

This difference in the linear-velocity mapping is because of structural differences in the yaw inertia between RRRobot and the Yaw model (see section 3.5.2). To overcome this difference, we define leg amplitude in the Yaw model as a function of leg offset to get a favorable comparison in curvature control for the full dynamics model and the Single Axis models. Here is one implementation: in the decoupled Yaw model, the leg motion amplitude is not fixed at 0.3; rather it is defined as a function of leg offset (see Fig. 3.55).

We compute the inverse of the mappings in Figs. 3.53 and 3.54 to derive a control relationship for the full dynamics RRRobot model (see Fig. 3.56) and the Single Axis models (see Fig. 3.57). Again, there are some discrepancies in the linear velocity mapping; but if we track only path curvature (since RRRobot's linear velocity is small), then the decoupled models provide a good approximation to the full dynamics model. Thus, given a desired linear and yaw velocity, we can find the

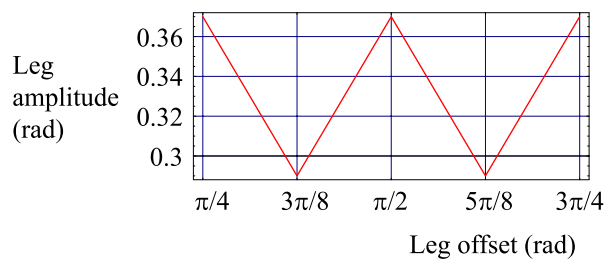


Figure 3.55: Amplitude modulation used in the decoupled Yaw model.

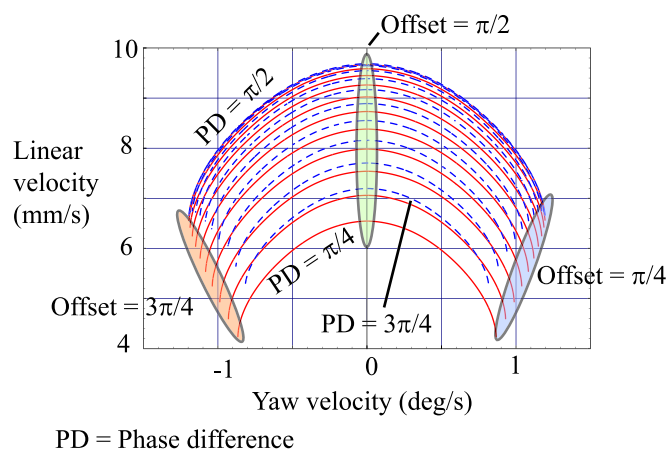


Figure 3.56: Mapping between RRRobot linear velocity and yaw velocity and leg offset and phase difference.

sinusoidal leg trajectory (in particular, the leg offset and phase difference) that tracks it.

This subsection provides a geometrical solution to RRRobot control by finding an approximate mapping between legless-locomotion translation and leg trajectories at steady state using dynamics decoupling. Note that we resort to a numerical comparison between the full dynamics and the simplified dynamics, since the structure of dynamics and nonholonomic kinematics makes a symbolic comparison difficult. The next subsection presents some preliminary work on tracking varying curvature paths.

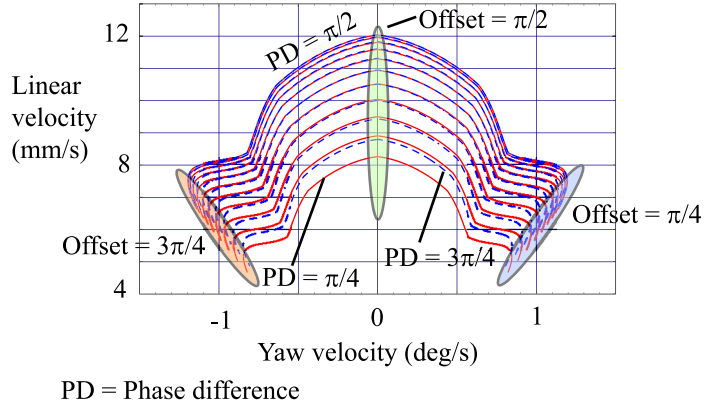


Figure 3.57: Mapping between RRRobot linear velocity and yaw velocity and leg offset and phase difference as predicted by the decoupled models.

3.6.2 Tracking Varying Curvature Paths

We now briefly present a method to find the leg motions to track a path with varying curvature. We assume that the transient dynamics is small by ensuring that the leg trajectory changes are slow and take advantage of legless locomotion's smooth dynamics.

RRRobot's net angular velocity α , linear velocity v , and curvature K are functions of the leg trajectories's phase difference ϕ and offset Γ (see Figs. 3.53 and 3.54); that is

$$\alpha = \alpha(\phi, \Gamma) \quad (3.22)$$

$$v = v(\phi, \Gamma) \quad (3.23)$$

$$K = \frac{\alpha(\phi, \Gamma)}{v(\phi, \Gamma)} \quad (3.24)$$

These relationships are expressed numerically in the previous section, since these properties are computed by integrating RRRobot's motion over a cycle of oscillations. We are interested in computing the leg trajectory changes for a given rate of change in curvature; that is, given $\frac{dK}{dt}$, we wish to compute $\frac{d\Gamma}{dt}$. We know that

$$\frac{dK}{dt} = \frac{\partial K}{\partial \phi} \frac{d\phi}{dt} + \frac{\partial K}{\partial \Gamma} \frac{d\Gamma}{dt}. \quad (3.25)$$

Now, from Figs. 3.54 and 3.53, we notice that $\frac{\partial K(\phi, \Gamma)}{\partial \phi}$ is approximately equal to zero. Thus,

$$\frac{dK}{dt} \approx \frac{\partial K}{\partial \Gamma} \frac{d\Gamma}{dt}. \quad (3.26)$$

Furthermore, we know from differentiation rules that

$$\frac{\partial K}{\partial \Gamma} = \frac{\frac{\partial \alpha(\phi, \Gamma)}{\partial \Gamma} \cdot v(\phi, \Gamma) - \frac{\partial v(\phi, \Gamma)}{\partial \Gamma} \cdot \alpha(\phi, \Gamma)}{v(\phi, \Gamma)^2}, \quad (3.27)$$

and $\frac{\partial \alpha(\phi, \Gamma)}{\partial \Gamma}$ is approximately a constant (from Fig. 3.53), say, C_Γ . Thus,

$$\frac{\partial K}{\partial \Gamma} = \frac{C_\Gamma \cdot v(\phi, \Gamma) - \frac{\partial v(\phi, \Gamma)}{\partial \Gamma} \cdot \alpha(\phi, \Gamma)}{v(\phi, \Gamma)^2}, \quad (3.28)$$

and

$$\frac{d\Gamma}{dt} = \frac{\frac{dK}{dt}}{\frac{C_\Gamma \cdot v(\phi, \Gamma) - \frac{\partial v(\phi, \Gamma)}{\partial \Gamma} \cdot \alpha(\phi, \Gamma)}{v(\phi, \Gamma)^2}}. \quad (3.29)$$

Thus, we can track varying curvature paths by varying the leg offset. Now, the key question is how large a rate of change of curvature can we track by this method? Note that the rate of change of leg offset $\frac{d\Gamma}{dt}$ increases with the rate of curvature change $\frac{dK}{dt}$, and the magnitude of transients increase with the magnitude of leg offset changes. Thus, transients can become significant with large $\frac{dK}{dt}$, and our analysis assumes that RRRobot has settled into a steady-state gait. As long as the rate of change of phase difference and offset are small, the dynamics transients are small and are damped out quickly.

Figs. 3.15 and 3.14 show plots of the time histories of RRRobot translation for the slow leg offset changes shown in Fig. 3.13 from simulation and experiment. Fig. 3.58 shows using simulation how planar translation changes for the constant-rate phase-difference changes in Fig. 3.59. Thus, we can vary the leg trajectories slowly to track varying curvature paths.

3.7 Summary

This chapter presents legless locomotion in detail: the concept, the models, the parameters, simulation and experiment results, and control techniques. There are other novel control strategies for locomoting high-centered robots, such as gaits that induce a robot to flip over; but this paper focuses

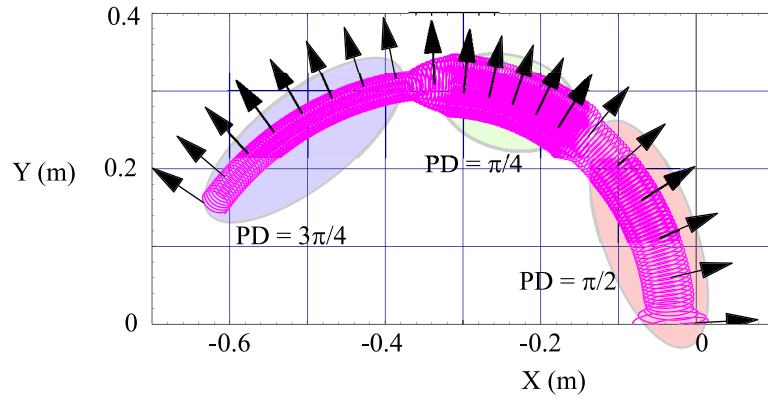


Figure 3.58: RRRobot planar translation as a function of leg phase-difference β shown in Fig. 3.59. The leg trajectories take the form $0.3 \sin(8t) + \pi/4$ and $0.3 \sin(8t + \beta) + \pi/4$.

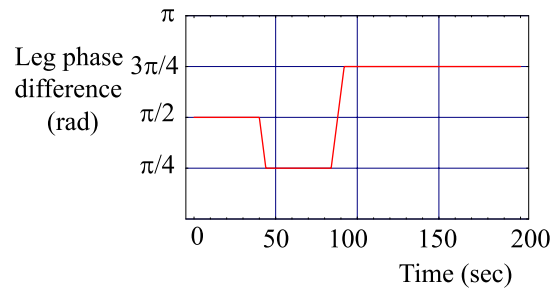


Figure 3.59: Variable RRRobot leg-trajectory phase differences.

only on legless-locomotion control. More control strategies are required for situations where conventional locomotion fails to improve the robustness of mobile robots. Future work also includes studying the dynamics of RRRobot without simplifications and other ideas outlined in chapter 5.

A key principle to note in legless locomotion is the deliberate use of direct actuation, dynamically coupled actuation, and exploiting environmental interaction to induce incremental motion. Even though legless locomotion is slow and inefficient, its structured behavior inspired by our initial experiments with RHex permits a unique control strategy for error recovery. Legless locomotion involves carefully choosing robot shape changes, namely out-of-phase leg motions, to produce translation. We now turn to the other mobile-robot error problem: freeing a wheeled robot from a ditch. While direct actuation using wheel torques produce robot motion, we show that dynamically coupled actuation through coordinated robot-shape changes may be necessary for error recovery.

Chapter 4

Dynamic Feedback Strategy for a Car in a Slippery Ditch

In this chapter, we continue with our research motivation of finding unconventional and dynamically coupled locomotion modes for mobile-robot error recovery by exploring a new problem—inducing a wheeled robot trapped in a slippery ditch to escape (see Fig. 4.1). We examine strategies for coordinated use of wheel torques and an active suspension to enable escape by overcoming insufficient traction. In addition to hand-tuned gaits, we also explore the solution space using dynamic programming.

Chapter 3 presents legless locomotion as a recovery mode for a high-centered robot using a combination of robot shape changes (out-of-phase leg motions), gravity-induced drift, and environmental interaction (slip-free contact). In this chapter, we present a method that enables a car to escape from a ditch using a combination of direct actuation (through wheel torques) and dynamically coupled actuation (using an active-suspension). The wheel torques build system kinetic energy, and the active suspension influences wheel-ground contact interaction by pushing and pulling on the car body.

There are many ways to model the problem of a car trapped in a ditch. We focus on the situation of a car trapped in a large slippery ditch, such as the situation in Fig. 4.1 and work with a simplified model, the Normal Car Body model (see Fig. 4.2), that captures the problem’s salient features.

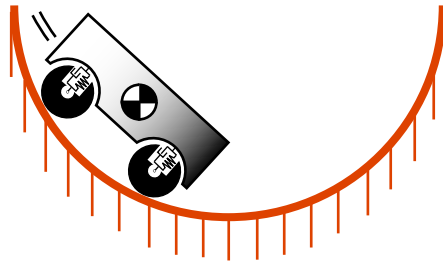


Figure 4.1: Schematic diagram of a wheeled robot in a large ditch. The robot mass sits on an active suspension.

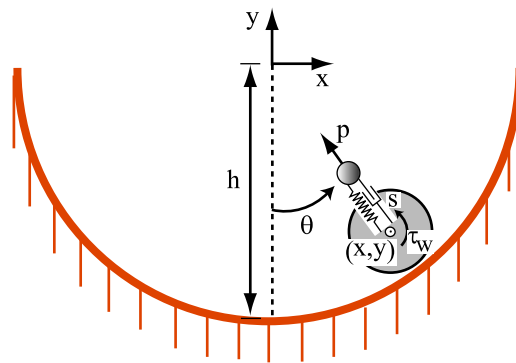


Figure 4.2: Schematic diagrams of the Normal Car Body model. The wheel, propelled by wheel torques, rolls in the ditch, and the car body is supported by an active suspension.

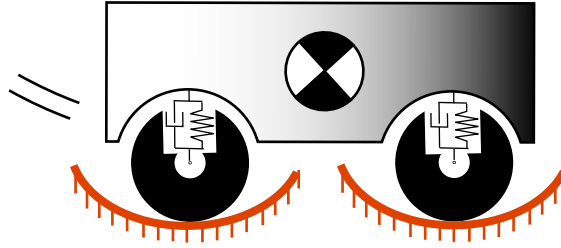


Figure 4.3: Schematic diagram of a wheeled robot in a small ditch. The robot mass sits on an active suspension.

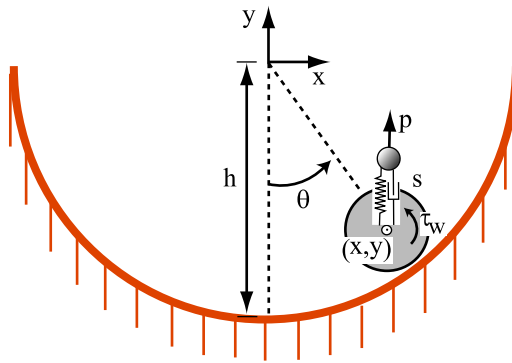


Figure 4.4: The vertical car body model.

Alternately, Fig. 4.3 shows the situation where a car is stuck in two small ditches. This problem can be represented using a Vertical Car Body model (see Fig. 4.4). While there may be differences between the dynamic structure between the Normal Mass model and the Vertical Mass model, we believe our approach applies to both models.

Working with the Normal Mass model, we explore the “stuck-car” problem by finding a control strategy that works for different problem parameters, such as varying ditch size and control limits. We also outline a dynamic programming implementation. We emphasize that this chapter contains only a preliminary analysis of the problem using simulation, and there is significant scope for future experimental analysis. The key contribution we make is an analysis of the problem structure (namely, the environmental contact and the available controls) to find novel control strategies.

We first present the Normal Car Body model in section 4.1 and then use thought experiments to explore the problem dynamics in section 4.2. We then present gait design and a discussion in

section 4.3. Note that there is a brief review of related work (such as automotive active-suspension use and feedback control) in chapter 2.

4.1 Modeling a Stuck Car in a Ditch

To study the problem of a wheeled robot stuck in a slippery ditch, we construct a simplified planar model called the Normal Car Body Model. The model consists of a wheel trapped at the bottom of a round ditch (see Fig. 4.2). The vehicle mass is supported at the wheel center by an active suspension, modeled as a spring and a prismatic actuator. The car body is constrained to move along the wheel-ground contact-normal direction (a body-fixed axis), similar to the car mass's motion in Fig. 4.1. We assume a Coulomb friction model [53] for wheel-ground traction.

The configuration $q_{nsm} = (\theta, s)^T$ of the Normal Car Body Model consists of the angular position of the wheel in the ditch θ and the normal car-body position s . The robot's state x_{nsm} consists of the robot's configuration and configuration velocities; that is,

$$x_{nsm} = (q_{nsm}^T, \dot{q}_{nsm}^T)^T = (\theta, s, \dot{\theta}, \dot{s})^T. \quad (4.1)$$

We assume we have complete state information, and the inputs available are the wheel torque τ and the active suspension force p that is applied between the the car body and the wheel.

The Normal Car Body model's equations of motion, derived using Lagrangian techniques [22], are given by

$$\ddot{\theta} = \frac{-R((h-R)^2(\tau + (m+M)gR\sin\theta) + MR(-h+R)(gs\sin\theta - 2(-h+R+s)\dot{s}\dot{\theta}))}{h((h-R)^2(I + (m+M)R^2) + MR^2s(-2h+2R+s))}, \quad (4.2)$$

$$\ddot{s} = g\cos\theta - (h-R)\dot{\theta}^2 + s\dot{\theta}^2(k(l-s) + p)/M. \quad (4.3)$$

The wheel-ground normal reaction force N is given by

$$N = N(p) = m(g\cos\theta(h-R)\dot{\theta}^2) + p + k(l-s), \quad (4.4)$$

and the tangential frictional force f is given by

$$\begin{aligned} f &= f(\tau), \\ &= \frac{(-MR\tau s^2 + M(h-R)s(2R\tau - Ig \sin \theta + 2I\dot{s}\dot{\theta}))}{((h-R)^2(I + (m+M)R^2) + MR^2s(-2h + 2R + s))} \\ &\quad + \frac{((h-R)^2((m+M)(-R\tau + Ig \sin \theta) - 2IM\dot{s}\dot{\theta}))}{((h-R)^2(I + (m+M)R^2) + MR^2s(-2h + 2R + s))}. \end{aligned} \quad (4.5)$$

Note that we consider N and f as functions of τ and p only, since we can choose the controls τ and p at any state x_{nsm} to control the ground reaction force and friction. Furthermore, the normal reaction N is independent of wheel torque τ , and the tangential friction f is independent of active suspension p . This allows us to control the normal reaction force and the friction forces independently using the two inputs instantaneously.

However, there are two constraints on the system, namely the contact constraint given by

$$N = N(p) > 0, \quad (4.6)$$

and the traction constraint (defined by the friction cone constraints) given by

$$\|f(\tau)\| \leq \mu N(p). \quad (4.7)$$

The controller must choose inputs (τ, p) to induce the robot to escape ($\|\theta\| \geq \pi/2$ and $\theta\dot{\theta} > 0$), while satisfying the constraints in (4.6) and (4.7).

We focus on solutions with no wheel-slip and no lift-off. In such cases, there is no energy loss as the robot passively rolls back and forth in the ditch. Losses occur only when the inputs perform negative work, such as when applying wheel torques against the wheel's motion, or when applying a force against the car body's motion. Note that solutions involving wheel-slip and lift-off do exist. For example, we can apply bang-bang wheel torques without worrying about wheel slip or allow the car body to deviate by large amounts to induce lift-off. Such solutions involve more energy losses and hybrid dynamics (for example, two sets of equations of motion, one for zero wheel-slip and another for non-zero wheel-slip) and are more difficult to model. We focus on solutions without wheel-slip and contact loss.

Now, suppose we find a control strategy for escape that satisfies the contact and traction constraints in (4.6) and (4.7); that is, a policy that returns a wheel torque and an active-suspension force given the system state. How can we measure if the policy is “good” or “bad”? What are the

Table 4.1: Normal Car Body Model Parameters

Model parameter	Symbol	Value
car body	M	1000 Kg
Wheel mass	m	100 Kg
Wheel inertia	I_0	12.5 Kg m/s ²
Wheel angular position in ditch	θ	$\in [-\pi/2, \pi/2]$
Wheel radius	R	0.5 m
Ditch radius	h	Varies: 6 m or 4 m
Spring constant	k	30000 N/m
car body position	s	$\in [-0.5, 1.5]$
Equilibrium car body position at $\theta = 0$	s_0	0.5 m
Gravity	g	9.81 m/s ²
Coefficient of friction	μ	0.1
Wheel torque	τ	Varies: $\in [-200, 200]$ or $[-300, 300]$ Nm
Active suspension force	p	$\in [-35000, 35000]$ N

problem's metrics? In the stuck-car problem, some quantities of interest are the time to escape, the work done by the inputs, and car body deviations. While it is evident that the solution is better if the time to escape, work done, and car-body deviations are all small, it is difficult to find a control policy that optimizes all the quantities of interest. Furthermore, the correct metric depends on the robot's circumstances.

In this chapter, we use a metric minimizing input impulses and car-body excursion from the nominal position. Minimizing the input impulses is similar to minimizing work done, while minimizing the car-body oscillations is similar to minimizing disturbances to the robot body.

Table 4.1 presents the values we use for the Normal Car Body model's parameters. These parameter values model a small car in a ditch. Note that the robot mass M is significantly larger than the wheel mass m . We explore the problem for ditch radius values set to 6 m or 4 m and wheel torque ranges set to $[-200, 200]$ Nm and $[-300, 300]$ Nm.

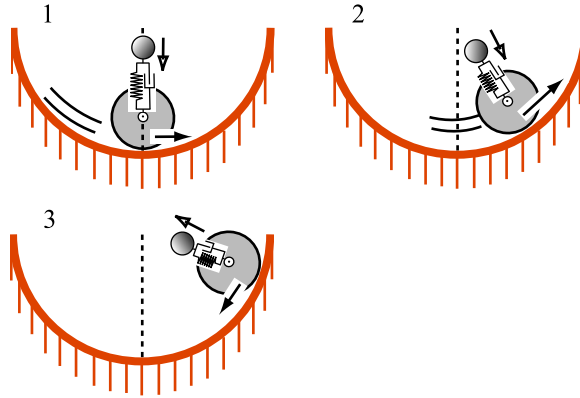


Figure 4.5: Schematic diagram of the interaction of wheel motion and car body motion. The white-headed arrows indicate the direction of car body motion, and the black-headed arrows the traction force.

4.2 Intuition into the Normal Model's Dynamics and Controls

In this subsection, we first discuss the dynamics of the interaction between wheel motion and car-body motion in the Normal Model and then explore some strategies for wheel torque and active-suspension use.

Suppose the wheel is rolling to the right at the ditch bottom, and the car body is at rest relative to the wheel ($\theta = 0, \dot{\theta} > 0, \dot{s} = 0$; stage 1 in Fig. 4.5). We set both inputs to zero ($\tau = p = 0$) and ignore centrifugal effects for the moment. As the wheel moves up the slope, the car body compresses the spring, since the car body has a tendency to stay at rest even if the wheel moves (inertia). When the wheel comes to rest along the slope, the spring is fully compressed (stage 3 in Fig. 4.5). When the wheel rolls back down the slope, the spring extends and is fully extended again at the bottom of the ditch $\theta = 0$. Thus, in the absence of any actuation or losses, the motion of the robot moving up the slope, including the car body's motion, is identical to the motion moving down the slope.

The effective system inertia (about the ditch center) changes as the car body moves, resulting in Coriolis effects (namely the $s\dot{\theta}$ terms in the equations of motion). For example, the inertia decreases as the car body approaches the ditch center. These inertia effects become significant as the ditch radius decreases and velocities increase, resulting in variations in the normal force acting on the wheel and influencing the traction force. For example, as the car body moves away from the ditch center, the traction forces must increase to accelerate the larger inertia and vice versa (stage 2 in

Fig. 4.5). To prevent slip when the Coriolis effects are large, the active suspension may be required to increase ground reaction forces. Note that for the parameter values we choose (for example, ditch radius 4 or 6 m), the Coriolis effects are not significant.

The Normal Car Body Model has a degenerate configuration at $s = h - R$, where wheel motion produces zero car-body translation. This degeneracy can be exploited to produce a solution with small time-to-escape and work done, since the heavy robot mass does not translate for any wheel motion. We explicitly avoid solutions that include this degenerate configuration, since it represents an infeasible motion for a practical system; for example, if the ditch radius is 2 m, the car mass would need to be moved an impractical 1.5 m!

Since the tangential gravitational force (acceleration) is large and the normal gravitational force is small along steep slopes, larger friction forces and wheel torques are required to spin the wheel closer to escape. This sometimes requires a large active-suspension force to prevent slip. Finally, as system velocity increases, centrifugal effects increase ground normal reaction and contribute to larger traction limits.

With the above relationship between car-body dynamics and the wheel motion in mind, we can choose how to use the wheel torques to accelerate the robot and the active suspension to push or pull the robot mass. One strategy for wheel-torque use is to build robot kinetic energy quickly using "bang-bang" torques—controls that alternate between extremes—while modulating the torques to accommodate the contact and traction constraints. One strategy for active-suspension use is to prevent the robot mass from oscillating. This prevents transferring wheel kinetic energy to the car body, thus building wheel momentum quickly and minimizing work done by the active suspension. But large fine-resolution active-suspension forces may be required to precisely maintain the car body at the desired configuration. Also, torque modulations may be necessary to satisfy the friction cone constraints with this strategy.

An alternate strategy for active-suspension use is to push against the car body to increase the ground reaction force and available traction. At such instances, the active suspension can perform negative work on the system, for example when applying an upward force when the car body is moving downward. Clearly, other strategies exist to use wheel torques and active-suspension forces in a coordinated manner, but we use the above two strategies in designing gaits in section 4.3.

Finally, it is difficult to identify the metric—for example, the work done or time to escape—that a chosen control policy optimizes, particularly for a dynamically coupled locomotion mode. Furthermore, measuring the quality of an error-recovery mode is an open problem, since dynamically coupled locomotion modes can be inefficient. We choose one specific metric involving work done

and car-body deviations to measure solution quality.

4.3 Gait Generation for the Stuck-Car Problem

We define a *gait* in the stuck-car problem as a coordinated use of wheel torques and active suspension forces. We are interested in gaits that utilize state information $(\theta, \dot{\theta}, s, \dot{s})$ to induce escape while respecting the traction constraints. In this section, we use the intuition developed in section 4.2 to find a feedback policy based on complete state information and then briefly explore gaits using dynamic programming.

4.3.1 A Hand-tuned Gait

Our hand-tuned gait presented in algorithm 1 is one method to choose the inputs τ and p to enable escape while respecting the contact and traction constraints. Put simply, the algorithm first tries to maintain car body position using the active suspension while modulating wheel torques to satisfy the friction-cone constraints. A stiff proportional-derivative controller with gains $K_P = 10^7$ and $K_D = 5 \times 10^6$ maintains car body position using active suspension forces

$$p = K_P(s - s_d) + K_D(\dot{s}). \quad (4.8)$$

Second, if the wheel-torque modulations are insufficient, we use the active suspension to control ground traction. Third, if both active-suspension and wheel-torque modulations are insufficient, the car cannot escape without wheel-slip; that is, there exist no wheel torques and active-suspension forces to satisfy the constraints. Note that this strategy is local in nature; that is, the controller returns the controls given a state and has no memory.

The results of using algorithm 1 are shown in Figs. 4.6, 4.10, 4.8, and 4.9 for different problem parameters, namely ditch radius and wheel torque limits. Table 4.2 presents a summary of the work done (measured using wheel-torque and active suspension impulses) and car-body oscillations in the four cases, using the following metric:

$$\text{Cost} = \int_{\text{Start}}^{\text{Goal}} (w_1 \|\tau\| + w_2 \|p\| + w_3 \|s - s_d\|) dt. \quad (4.9)$$

We set weights w_1 to 10, w_2 to 100, and w_3 to 35000. Note that a large w_3 penalizes car-body deviation from the nominal position s_d . For example, applying a wheel torque of 170 Nm is less

Algorithm 1 Algorithm to choose wheel torques and active-suspension forces to induce escape

- 1: Find active suspension force p_N to maintain $N = 0$ using (4.4).
 - 2: Find active suspension force p_S to maintain the sprung mass at the nominal position using the proportional derivative controller in (4.8).
 - 3: **if** $p_S > p_N$ **then**
 - 4: Set $p = p_S$,
 - 5: **else**
 - 6: Set $p = p_N$
 - 7: **end if**
 - 8: Compute $N = N(p)$ from (4.4).
 - 9: Compute $f_{lim} = \text{sgn}(-\dot{\theta})\mu N$.
 - 10: Compute $\tau = \tau(f_{lim})$ from (4.5).
 - 11: **if** $\tau \in [\tau_{min}, \tau_{max}]$ **then**
 - 12: Return (τ, p) .
 - 13: **end if**
 - 14: Set $\tau_{lim} = \text{sgn}(-\dot{\theta})\tau_{max}$.
 - 15: Compute $N_{reqd} = f(\tau_{lim})/\mu$ from (4.5).
 - 16: Compute $p_{reqd} = p(N_{reqd})$ from (4.4).
 - 17: **if** $p_{reqd} \in [p_{min}, p_{max}]$ **then**
 - 18: Return (τ_{lim}, p_{reqd}) .
 - 19: **end if**
 - 20: Return “Car cannot escape without wheel-ground slip”.
-

Table 4.2: Evaluation of algorithm 1 using the metric $\alpha\|\tau\| + \beta\|p\| + \gamma\|s - s_d\|$.

System	Solution cost
Ditch radius 6 m, $\tau_{max} = 300$	1114 (823 + 97 + 194)
Ditch radius 6 m, $\tau_{max} = 200$	2402 (845 + 149 + 1407)
Ditch radius 4 m, $\tau_{max} = 300$	868 (640 + 82 + 146)
Ditch radius 4 m, $\tau_{max} = 200$	1624 (619 + 107 + 898)

than half the cost of allowing the car body to deviate 0.1 m from the nominal position s_d .

We now review the solution in Fig. 4.6. The robot escapes in less than 30 seconds, and the car body moves little from the desired position $s_d = 0.5$ m. This implies that the active suspension can be replaced with a rigid link, and the robot can still escape. The wheel torques are mostly “bang-bang”, building kinetic energy quickly. The controller reduces the wheel torque magnitude though, when the friction forces reach the friction-cone limits. Thus, wheel torque modulations are alone sufficient for the robot to escape for these specific problem parameters.

The magnitude of active suspension forces used increase in magnitude as the robot nears the top of the ditch. This is because the normal gravitational component decreases along steep slopes, and the controller compensates for the weak normal gravitational component by pushing the car body upward. Also as the robot’s velocity increases, the centrifugal forces influence car-body motion. Thus to counter the car body motion, larger active suspension forces are required.

The solutions in Fig. 4.10, 4.8, and 4.9 are similar, but the problem parameters require the active suspension to control traction. For example, we see a spike in the car-body position close to escape. This indicates that the robot cannot escape without active-suspension use using algorithm 1.

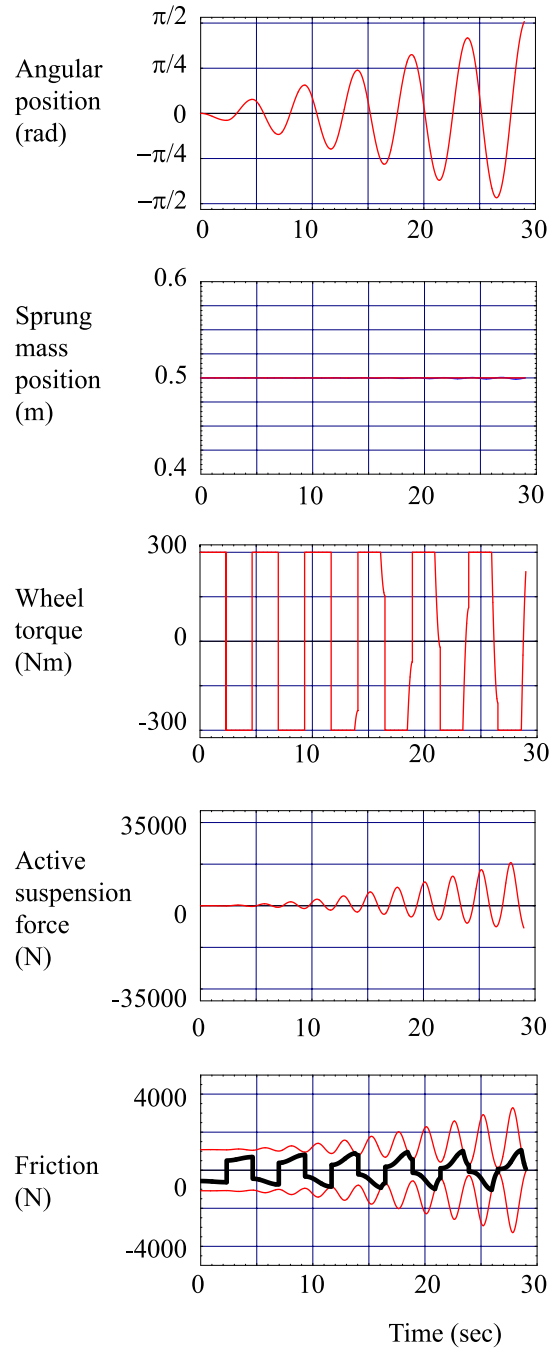


Figure 4.6: Algorithm 1 applied to the Normal Car Body model (ditch radius 6 m and maximum wheel torque magnitude 300 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.

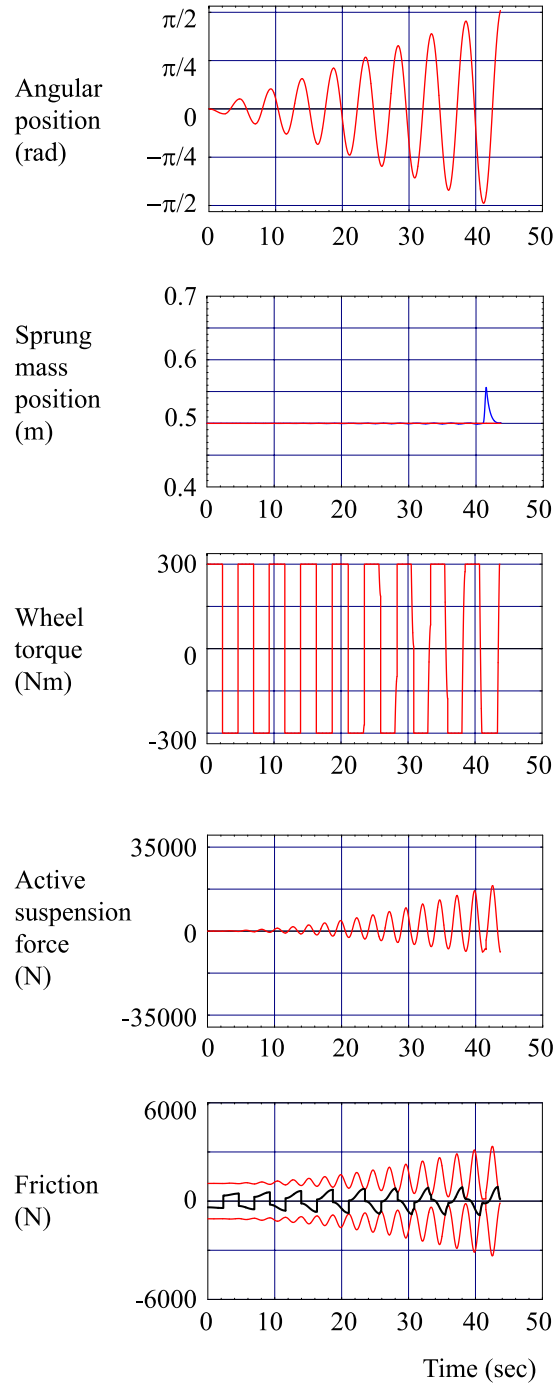


Figure 4.7: Algorithm 1 applied to the Normal Car Body model (ditch radius 6 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.

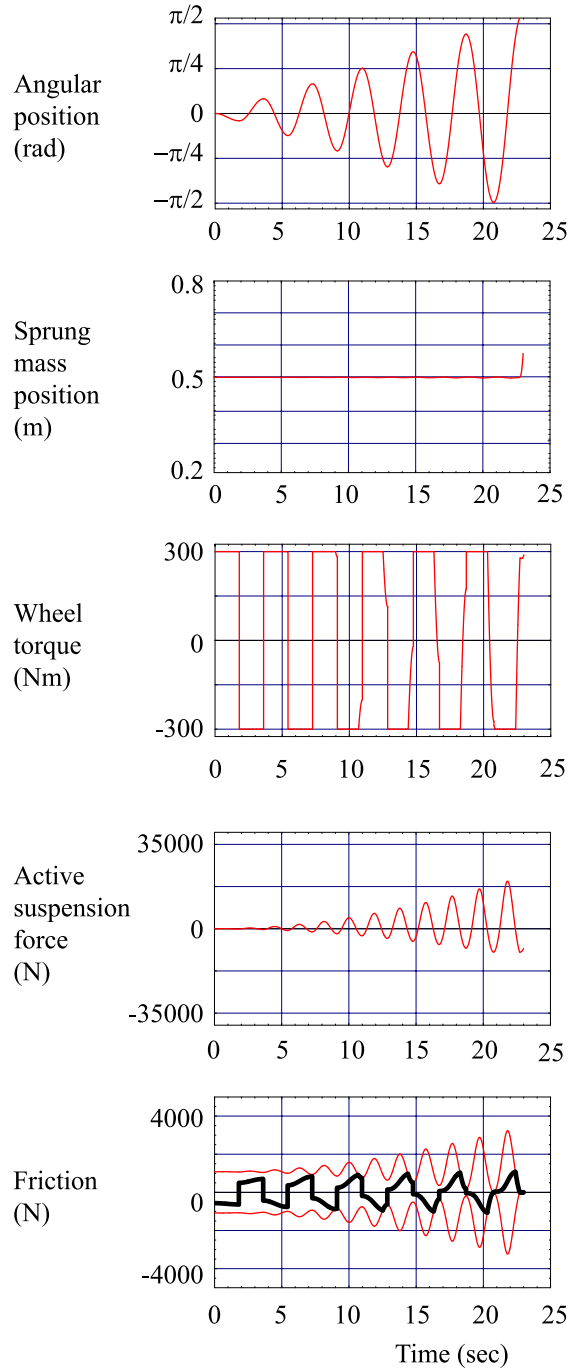


Figure 4.8: Algorithm 1 applied to the Normal Car Body model (ditch radius 4 m and maximum wheel torque magnitude 300 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.

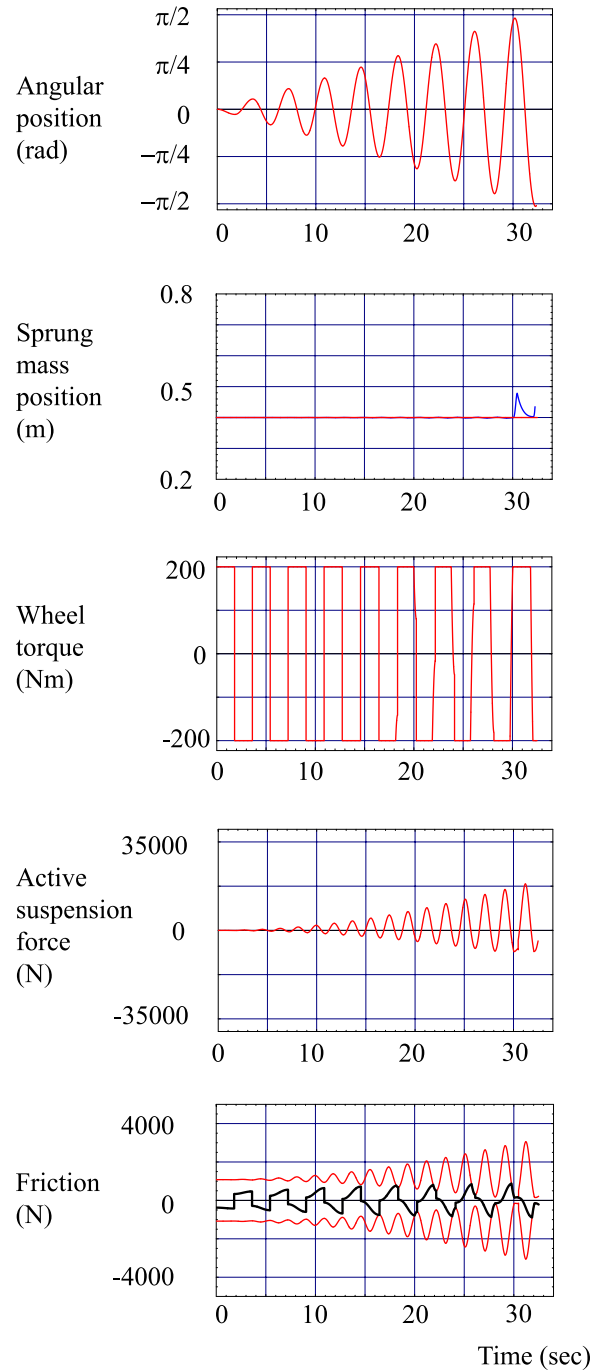


Figure 4.9: Algorithm 1 applied to the Normal Car Body model (ditch radius 4 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and car body position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.

4.3.2 Gait Search using Dynamic Programming

Even though algorithm 1 allows the robot to escape while keeping the car body deviations small, we do not know what metric exactly (for example, work done or time consumed) it optimizes. Numerical approaches such as dynamic programming can derive solutions that optimize specific objective functions. We implemented a dynamic programming approach based on Lavalle and Konkimalla's work [46], where they use a discretized state and input space to find an optimal control policy, based on a chosen metric, from valid states to the goal. We then use the feedback policy to derive the optimal path from any state to the goal by chaining the sequence of optimal inputs to the goal.

However, numerical approaches have potential pitfalls such as the exponential increase in computational expense with fine resolution. In particular, the stuck-car problem's combined state-input space has dimension six, leading to an exponential growth in computational expense with fine resolution. So it is important to find a state-input space resolution that provides a good approximation of the problem dynamics and is not computationally prohibitive. A numerical approach also requires a discretized version of the metric; so in the stuck-car problem, we could replace the integral in (4.9) with a sum to compute

$$\text{Cost} = \sum_{\text{Start}}^{\text{Goal}} (w_1 \|\tau\| + w_2 \|p\| + w_3 \|s - s_d\|) \Delta t. \quad (4.10)$$

Using a large value for w_3 (35000) has two advantages: the sprung mass is not automatically moved to the degenerate configuration $s = h - R$, and the car body oscillations are automatically minimized.

Fig. 4.10 shows a comparison of the car's motion using algorithm 1 and a dynamic programming implementation, for the case where ditch radius is 6 m and the maximum available torque is 200 Nm. Table 4.3 shows the parameters we use for the dynamic programming implementation.

4.3.3 Comparing Algorithm 1 and the Dynamic Programming Gait

We emphasize that policies derived from numerical approaches are approximate, since the state and input spaces are discretized. The cost of the dynamic programming solution, computed using (4.10) was much larger (total cost = 55256, torque use cost = 1071, active suspension use = 144, and car-body perturbations cost = 54041) than the cost of the solution prescribed by algorithm 1 (total cost = 2402, see Table 4.2). This may seem odd at start because dynamic programming gives the

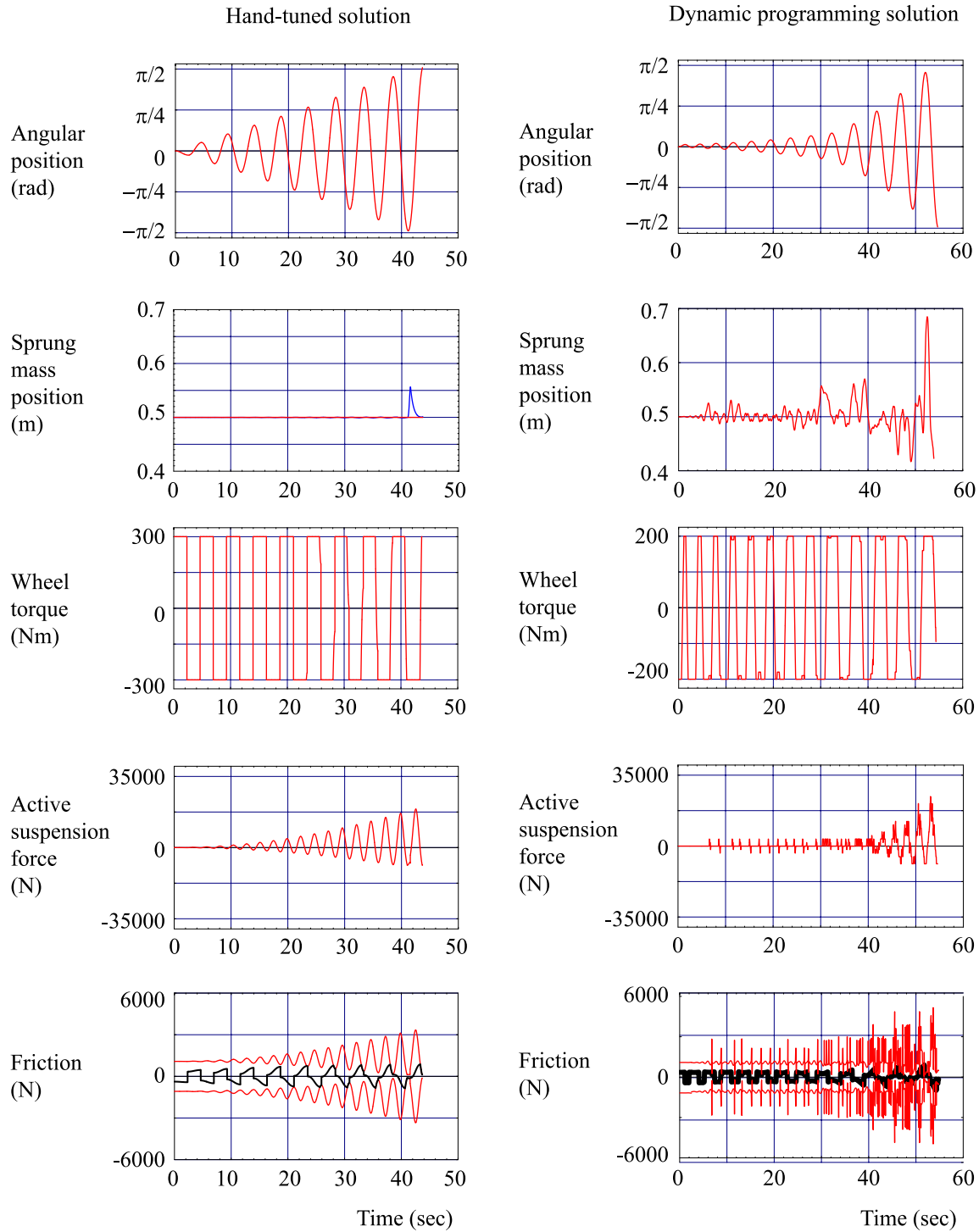


Figure 4.10: A comparison of a solution based on algorithm 1 and a solution derived from dynamic programming applied to the Normal Sprung Mass model (ditch radius 6 m and maximum wheel torque magnitude 200 Nm). Plots show the phase relationship between wheel angular position and sprung mass position, and the time history of wheel torque, the active-suspension force, and the traction force (bold) with the friction cone constraints.

Table 4.3: Dynamic Programming Parameters

Total number of states = 25857		
Model variable	Range of values	Resolution
Angular position θ (rad)	$[-1.67, +1.67]$	0.2
Angular velocity $\dot{\theta}$ (rad/s)	$[-6, 6]$	1.0
Sprung mass position s (m)	$[-1.5, 1.5]$	0.25
Sprung mass velocity \dot{s} (m/s)	$[-4, 4]$	1.0

Total number of inputs = 81		
Model variable	Range of values	Resolution
Torque τ (Nm)	$[-200, 200]$ and $[-300, 300]$	$\tau_{max}/4$
Internal actuation p (N)	$[-35000, 35000]$	$p_{max}/4$
Time step (sec)	—	0.05

Number of states \times inputs > 2 million.

solution with least cost, but the larger cost is because the dynamic programming implementation does not have sufficient input and state-space resolution. Hence, the dynamic programming solution has a limited choice at any state. The problem is that we are searching for a control strategy for a highly dynamic system like the sprung-mass oscillations using a discretized state-input space, and the dynamic programming solution can only give us an approximate solution with an upper bound on the cost. The discretization also induces a noisy solution, because of the state-input space discretization. For example, since the time-scale of car body motions on the active suspensions are much smaller than the time-scale of car motions in the ditch, a numerical solution may involve high-frequency oscillations of the car body on the active suspension.

We note that the car ramps up slowly using the dynamic programming solution when compared with Algorithm 1. This may be related to resolution issues again, since the dynamic programming policy “waits” for the right state to use bang-bang wheel torques and minimizes work done until that state is reached. Finally, the dynamic programming policy satisfies the constraints only at the states in the discretized state space and is not guaranteed to satisfy the constraints at all states along the path. In general, numerical techniques have disadvantages when exploring complex dynamical systems, but their application becomes wider as computational power increases and we find more efficient ways to explore the state space. While we have made a small attempt to apply dynamic

programming to analyze the control space of error recovery modes, it will be interesting to explore error recovery modes again using numerical techniques in the future.

In contrast to the Normal Mass model which has a singularity when the car body moves to the ditch center, the Vertical Mass model has a degenerate configuration at $\theta = \pi/2$ where the robot's motion is decoupled from the car-body's motion. Despite this difference, we believe our analysis applies to the Vertical Mass model also.

Note that this locomotion mode of using coordinated wheel torques and active-suspension forces is necessary only when ground traction is insufficient. The key contribution of this work is neither the simplified model nor the specific parameter values we choose to represent a robot stuck in a ditch, but the structure we provide to an unconventional locomotion mode to solve a mobile-robot error-recovery problem. Instead of randomly rocking the car back and forth using wheel torques and brakes to induce escape, we show that a solution exists using a combination of wheel torques and an active suspension. By understanding the problem dynamics, we present a simple feedback-based algorithm for inducing escape.

We now summarize this chapter's results.

4.4 Summary

We explored the problem of a car stuck in a ditch using simulation and presented an analysis of the key problem dynamics. Future work includes experimental work and an understanding the impact of various parameters such as the robot mass, the wheel and ditch radius, and the traction constraints on the system dynamics on the solution. Also, we note that a numerical analysis of a highly dynamic system has many pitfalls due to discretization; it can best give us a approximate solution. Furthermore, while applying these ideas in real systems, robust state estimation is also required.

The key idea behind this chapter, like chapter 3, is to show that the unconventional use of a robot's freedoms can play a significant role in mobile-robot error recovery. Since the robot's conventional locomotion mode does not apply in a locomotion error, a coordinated use of direct and indirect actuation is necessary to induce motion. We now summarize this thesis's contributions and present future work in Chapter 5.

Chapter 5

Conclusion

In this chapter, we first summarize the thesis’s contributions, followed by a listing in section 5.2 of possible extensions of our research. We then present some closing thoughts in section 5.3.

5.1 Contributions

This doctoral thesis makes three contributions:

1. Exploring the structure in mobile-robot error recovery.
2. Discovering and understanding legless locomotion.
3. Designing gaits for locomotion error recovery.

We show using two case studies, namely the high-centered robot problem and the stuck-car problem, that mobile-robot error recovery has structure that can be exploited to devise novel escape strategies. A key result we show is that carefully designed escape gaits lead to controllable solutions. In the high-centered robot problem, we exploit the available freedoms, namely unconstrained leg motions, to excite body motions which interact with the environmental contact to induce incremental translation. While such translation is slow, it may be the only available locomotion mode for escaping from a high-centered state. In the stuck-car problem, we show that an active suspension helps overcome poor traction to induce a wheeled robot to escape from a slippery ditch. We obtain significant insight into both systems using first principles, such as Lagrangian dynamics. However, our approach does not model the uncertainty in robot-environment interaction, which is a interesting future research problem.

A key contribution of this thesis is an analysis of legless locomotion, using prototype robots, theory, simulations, experiments, and models. We show that the high-centered Rocking and Rolling Robot, a rocking hemisphere with two actuated legs, has complete planar accessibility by choosing different leg trajectories. Since RRRobot's dynamics is complex, we define a set of simplified models that allow us to quantify the contributions of the RRRobot's dynamics and kinematics to its locomotion.

Finally, we take advantage of a robot's remaining freedoms and the robot's environmental interaction during a locomotion error to find locomotion recovery gaits. In the high-centered robot problem, we use simplified models to analyze legless locomotion's elements, allowing us to carefully choose leg motions to induce translation. In the stuck-car problem, in addition to exploring gaits using mechanical systems analysis, we use dynamic programming to explore the solution space for different problem parameters. Note that we choose different methods in the two problems, since there is a difference in problem complexity: RRRobot's state space is \mathbb{R}^{14} , while the stuck-car problem's state space is \mathbb{R}^4 .

We now present possible future directions to build on our work.

5.2 Future Directions

While we have explored two mobile-robot error recovery problems and control and planning for mechanical systems, many open problems remain. We now suggest interesting future directions.

5.2.1 Automatic Control Strategies for Robots

A robot's performance is enhanced if it can automatically choose from a repertoire of control techniques. For example, when the Mars rover gets stuck in sand or when a car spins out of control, conventional operation is infeasible because environmental interaction is unreliable. Since teleoperation is not always effective, can an autonomous robot perform better by choosing the right control strategy from its behavior suite? The ultimate goal is to design robots that are aware of their capabilities under varying conditions and can choose the best available action depending on goals. The key then is to develop many control techniques for mobile robots and find the conditions that necessitate transitions from one mode to another.

5.2.2 Robot Mobility and Design

What is the best way to quantify a robot's mobility? Most measures of robotic mobility, such as speed or the height of a step the robot can climb, are based on a local analysis of a robot's conventional locomotion capabilities. A proper quantification of a robot's mobility should include even extreme behaviors, possibly derived using global analysis. For example, a local analysis shows that a wheel starting from rest cannot climb a step with weak motor torques. But a global analysis will show that the wheel can climb the step with sufficient momentum and carefully timed wheel torques during impact. Thus, a robot with weak actuators can achieve greater mobility with dynamic behaviors. Of course, robots do not always require extreme dynamic behaviors, and a balance between robot design and mobility is required. A systematic global and local kinematics and dynamics analysis will help quantify a robot's true mobility, possibly resulting in a mapping between robot complexity and mobility. Such an analysis ultimately leads to minimalist and better designs,

5.2.3 Uncertainty in Robot-Environment Interaction

The uncertainty in robot state and robot-environment interaction significantly affects mobility. For example, a person walking on icy ground does not know and cannot predict the instantaneous foot-ground forces. One strategy to plan a path across the ice is to assume worst-case scenarios such as poor traction forces, but worst-case scenarios may preclude a solution. This problem becomes particularly acute in locomotion errors where information about robot state and the environment-interaction is rarely available. Can we find control strategies that factor in such uncertainty, say, using aggressive maneuvers when information is available and conservative strategies at other times? Furthermore, sometimes the robot-environment interaction modes alternate rapidly. An interesting research problem is to find automatic control strategies in the presence of uncertainty in robot-environment interaction.

5.2.4 Alternate Rocking and Rolling Robot Designs

RRRobot's design is good to demonstrate legless locomotion, but alternate designs exist. This subsection looks at three alternate designs: 1) an RRRobot with orthogonal hip joints, 2) an RRRobot with many legs, and 3) an RRRobot with one multi-degree-of-freedom leg.

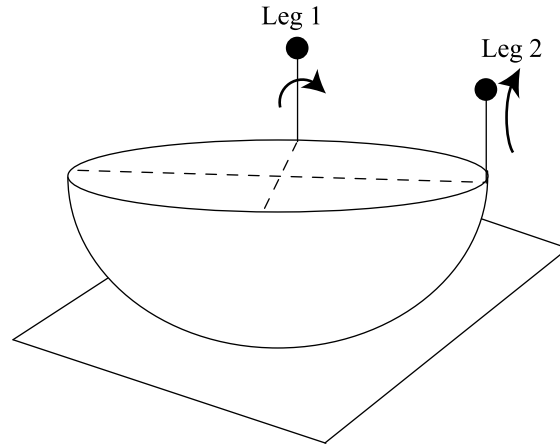


Figure 5.1: Schematic diagram of an RRRobot with orthogonal leg rotational axes.

The “Ortho-RRRobot”

One alternate RRRobot design is to have orthogonal hip joints (see Fig. 5.1), instead of aligned leg axes. Note that the robot retains the sphere-plane contact kinematics, but the leg motions that induce translation may differ. Also, both roll and pitch oscillation amplitudes will be comparable. One exercise is to find new gaits for this RRRobot; that is, find the leg motions that excite out-of-phase body rotational oscillations.

A Multi-Legged RRRobot

An alternate RRRobot design is a multi-legged RRRobot. Consider an RRRobot with $2n, n > 1$, legs placed in a symmetrical manner around the hemisphere’s largest circumference (see Fig. 5.2, where n equals four). While the original RRRobot locomotes laterally, what is the space of planar motions available to the multi-legged RRRobot in Fig. 5.2? For example, we know that in-phase motions of opposite legs produce pure body pitch rotation. Thus, by using multiple pairs of opposing legs, we may be able to induce translation through out-of-phase body oscillations. Furthermore, there may exist leg motions that induce translation in any direction.

An interesting associated problem is to find the number of effective legs required to produce the desired body attitude oscillations. For example, in the two-legged RRRobot, we can actuate motions along the body pitch and yaw axes with two legs at different phases, although we do not have individual control over each of the body attitude freedoms. But if we restrict the leg motions to have the same phase, then we lose the ability to yaw and roll, and the body can only pitch.

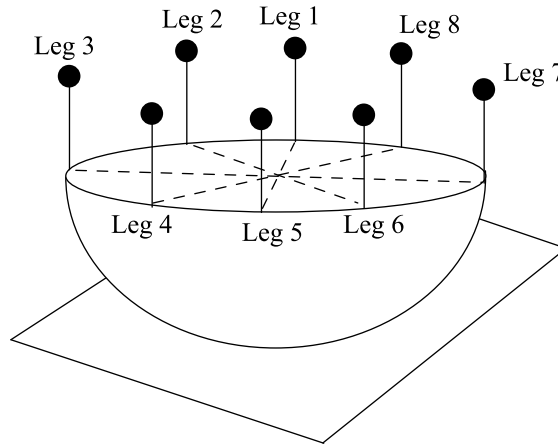


Figure 5.2: An eight-legged RRRobot

This indicates that we have lost some control over the robot. This notion can be extended to the multi-legged version of RRRobot also.

Consider two extreme sinusoidal gaits: the first gait where each leg has different phase and offsets, and the second gait where each leg has the same phase and offset. The first gait allows interesting, but complex, phase relationships between the body's attitude freedoms, reflecting the notion that there are $2n$ effective legs. In contrast, the second gait can only produce body attitude rotations with a single phase; that is, the second gait has only one effective leg. What controllability features do we lose when moving from the first gait to the second gait? An interesting problem is to explore the capabilities of a multi-legged RRRobot for varying number of effective legs.

Now, suppose we understand the relation between the number of effective legs and the achievable body attitude trajectory, then we can select gaits based on the desired body attitude trajectory. Coupling this knowledge with our experience that body roll-yaw oscillations produce predominantly forward locomotion, pitch-yaw oscillations produce predominantly sideways locomotion, and roll-pitch-yaw oscillations with yaw drift produce curved paths, we can choose gaits for desired trajectories based on the number of effective legs required.

For example, if the required body trajectory is only pitching motion, then we can choose a gait where only two opposite legs have motions are in phase, and the rest of the legs are stationary. For example, if we require yaw and pitch oscillations, we may use out of phase motions for legs 1 and 5 for producing yaw motions and in phase motions for legs 2, 4, 6, and 8 for producing pitch oscillations. Legs 3 and 7 can be stationary. This will result in a losing degrees of freedom (from eight legs to two effective legs), but may be sufficient to achieve the required body attitude trajectory.

A Legless Locomotion Template

Finally, is there a simple RRRobot design that captures the essence of legless locomotion's dynamics and kinematics? This is similar to the idea of finding templates for legged locomotion. Full and Koditschek [63] define a template as a model created by trimming away all the incidental complexity of joints and that which serves as a guide for locomotion control. In short, templates are useful to understand the broad principles underlying a physical system. For example, Raibert [60] first showed that quadruped locomotion may be organized with reference to a single virtual leg model called the Simple Inverted Pendulum model. Recently Saranli et al. [67] have shown that the motion of a ankle-actuated, knee-actuated, and hip-actuated monopod can also be tied to the SLIP model.

A valid template for legless locomotion must capture the broad principles of legless locomotion, namely:

- Underactuation: RRRobot has seven degrees of freedom with just two actuators.
- Out-of-phase body attitude oscillations: Out-of-phase body pitch-yaw rotations induce lateral translation while out-of-phase body roll-pitch-yaw rotations with body yaw drift induce circular translation.
- Exploiting inertia differences: Out-of-phase leg motions induce net body yaw by exploiting the varying system inertia.

We propose a legless locomotion template where RRRobot has only one leg, pivoted at the geometric center of the sphere at a spherical joint (see Fig. 5.3). The massless leg has a point mass at its distal end, and is actuated in each of its three rotational freedoms.

We believe that this template captures the important elements of legless locomotion. Clearly, the generic model is underactuated (eight degrees of freedom with just three actuators); out-of-phase body rotations can be produced using, say, circular leg motions; inertia differences can be created using suitable leg roll, yaw, and pitch configurations. Producing pitch and roll body rotations is straight-forward, since the body's rotations and leg motions are coupled along these axes. Producing yaw rotations requires a combined roll-yaw or pitch-yaw leg rotation. In fact, using a single leg pivoted at a spherical joint permits some capabilities beyond the two-legged RRRobot; for example, the model can tilt itself while locomoting by moving the leg to either side. There are some restrictions on body motions though; for example, body yaw is impossible when the leg is vertical, since the leg has no inertia; but body yaw is possible by rolling or pitching the leg slightly, and then

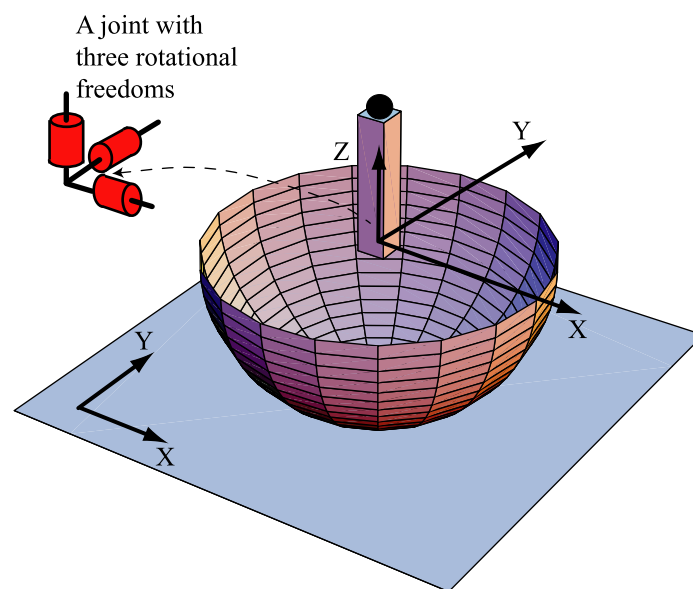


Figure 5.3: Legless locomotion generic model.

yawing the leg. It will be interesting to see if this generic model can achieve all the body rotations required to produce locomotion.

To compare the template's motion with the two-legged RRRobot, we require a metric for equivalence. There are two ways to compare the motion of the template with the two legged RRRobot:

- Compare the body rotational trajectories's time-histories, using metrics such as frequency, amplitude, and phase.
- Compare average planar paths, since planar translation is the main goal of RRRobot's locomotion.

One view of the leg motions that achieve the same body translation in the template RRRobot and the original RRRobot is that the two legs in RRRobot effectively reduce to a single leg with multiple rotational freedoms. In summary, a legless locomotion template helps understand the principles behind underactuated undulatory locomotion.

5.2.5 Kinematic Reductions for Mechanical Systems with Gravity

The key benefit of a kinematic reduction is that it eliminates the robot motion's dependence on time, and the robot motion can be expressed as a function of its configuration only. But this is difficult for legless locomotion, because of the gravity-induced pitch and roll oscillations which maintain a clock. So the key idea is to work inside this clock to create body yaw rotations, since RRRobot's body yaw rotation is dependent on configuration only.

A kinematic reduction for RRRobot could involve finding the leg motions to achieve body rotations like in Fig. 3.3. While the body-pitch oscillations always include gravitational drift, the key idea is to create quick body-yaw rotations at the extremes of the body pitch oscillation cycles. One way to achieve this is to excite pitch oscillations, and while the robot has non-zero pitch configuration, we induce body yaw through a quick out-of-phase leg motion.

Furthermore, Lewis and Murray [49] suggest a method to develop kinematic reductions for systems with gravitational drift. This when combined with the height-functions work of Shammass et al. [50], albeit for systems without gravitational drift, may lead to kinematic reductions for complex mechanical systems like RRRobot. This thesis only provides a kinematic reduction for the yaw model, a simplified version of RRRobot with no gravitational drift. Finding a kinematic reduction for a complex mechanical system like Rocking and Rolling Robot is an open problem.

5.2.6 Stuck Car Problem: Experimental Validation

While we provide models and analysis for the stuck-car problem in chapter 4, experimental validation is required. One experimental set-up to validate the models is the following: a remote-control car that moves up and down a planar semi-circular ditch (say, created with a retainer wall). A solenoid attached to the car moves a reaction mass up and down to create variations in ground traction force. The car and solenoid is controlled by a PIC processor such as the Cerebellum [6], connected possibly using a tether. The key challenge is to monitor wheel slip. We suggest a motion-capture system like Vicon [7] to track the car's motion and wheel rotation. Since the Vicon system has a high frame-rate (120 frames/sec), we can design wheel torques and solenoid actuation depending on system state.

We now present some closing thoughts.

5.3 Closing Thoughts

Mobile robotics is challenging because of the numerous variables, including robot design, the operational environment, and robot operation mode. A significant ability missing in mobile robots is error recovery. We have explored only two specific error recovery problems using simple models and prototype robots; but we show that we can exploit the structure in locomotion errors to find novel locomotion modes for escape. There is significant scope for more research to improve the robustness of mobile robots. The goal is to analyze the structure in different locomotion errors and provide a robot with multiple control strategies for recovery. The robot can then automatically choose the right control strategy for the particular situation.

Bibliography

- [1] <http://www.rec.ri.cmu.edu/projects/autonomous/index.htm>.
- [2] Mathematica 4.2. <http://www.wolfram.com>.
- [3] Red Team Carnegie Mellon University. <http://redteamracing.org/>.
- [4] RHex experiment movies. <http://www.cs.cmu.edu/~bravi/research.html>.
- [5] The sony aibo. <http://en.wikipedia.org/wiki/Aibo>.
- [6] The Cerebellum PIC board. <http://www.botrics.com/products/cereb>.
- [7] Vicon motion capture system. <http://www.vicon.com>.
- [8] R. Abraham and J.E.Marsden. *Foundations of Mechanics*. Reading, MA: Addison-Wesley, 1978.
- [9] Andrew Alleyne. Improved vehicle performance using combined suspension and braking forces. *Vehicle System Dynamics*, 27:235–265, 1997.
- [10] Richard Altendorfer, Uluc Saranli, H. Komsuoglu, Daniel Koditschek, Jr. H. B. Brown, Martin Buehler, Ned Moore, D. McMordie, and Robert Full. *Evidence for a Spring Loaded Inverted Pendulum Running in a Hexapod Robot*. Springer Verlag, 2001.
- [11] James Bagnell, Andrew Y. Ng, and Jeff Schneider. Solving uncertain markov decision problems. Technical Report CMU-RI-TR-01-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2001.
- [12] R. Balasubramanian and Alfred A. Rizzi. Kinematic reduction and planning using symmetry for a variable inertia mechanical system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3829–3834, 2004.

- [13] R. Balasubramanian, Alfred A. Rizzi, and Matthew T. Mason. Toward legless locomotion control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.
- [14] Ravi Balasubramanian. Legless locomotion: Concept and analysis. Technical report, The Robotics Institute, Carnegie Mellon University, 2004.
- [15] Ravi Balasubramanian, Brendan Meeder, Alfred A. Rizzi, and Matthew T. Mason. Legless locomotion: Freeing high-centered legged robots (movie). In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.
- [16] Ravi Balasubramanian, Alfred A. Rizzi, and Matthew T. Mason. Legless locomotion for legged robots. In *Proceedings of the International Conference on Robots and Intelligent Systems*, volume 1, pages 880–885, 2003.
- [17] Ravi Balasubramanian, Alfred A. Rizzi, and Matthew T. Mason. Legless locomotion: Models and experimental demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.
- [18] John Bares and David Wettergreen. Dante II: Technical description, results and lessons learned. *International Journal of Robotics Research*, 18(7):621–649, July 1999.
- [19] A. Bellaiche, J.P. Laumond, and P. Jacobs. Controllability of car-like robots and complexity of the motion planning problem. In *Proceedings of the International Symposium on Intelligent Robotics*, pages 322–337, 1991.
- [20] S. Bhattacharya and S.K.Agrawal. Spherical rolling robot: A design and motion planning studies. *IEEE Transactions on Robotics and Automation*, 16(6), 2000.
- [21] Antonio Bicchi, Andrea Balluchi, Domenico Prattichizzo, and Andrea Gorelli. Introducing the sphericle: an experimental testbed for research and teaching in nonholonomy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2620–2625, 1997.
- [22] A.M. Bloch, J. Baillieul, P.Crouch, and J. Marsden. *Nonholonomic Mechanics and Control*. Springer, 2003.
- [23] F. Bullo, A. D. Lewis, and K. M. Lynch. Controllable kinematic reductions for mechanical systems: concepts, computational tools, and examples. In *Mathematical Theory of Networks and Systems*, August 2002.

- [24] F. Bullo and Kevin M. Lynch. Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems. *IEEE Transactions on Robotics and Automation*, 17(4):402–412, August 2001.
- [25] Francesco Bullo and Andrew D. Lewis. *Geometric Control of Mechanical Systems Modeling, Analysis, and Design for Simple Mechanical Control Systems*. Springer-Verlag New York-Heidelberg-Berlin, 2004.
- [26] Carlo Camicia, Fabio Conticelli, and Antonio Bicchi. Nonholonomic kinematics and dynamics of the sphericle. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 805–810, 2000.
- [27] Lillian Y. Chang and Nancy S. Pollard. Constrained least-squares optimization for robust estimation of center of rotation. *Journal of Biomechanics*, 2006. In press.
- [28] S. Chantranuwathana and Huei Peng. Adaptive robust control for active suspensions. In *Proceedings of the American Control Conference*, volume 3, pages 1702–1706, 1999.
- [29] Wuwei Chen, James K. Mills, and Le Wu. Neurofuzzy and fuzzy control of automotive semi-active suspensions. *Internation Journal of Vehicle Autonomous Systems*, 1(2):222–236, 2003.
- [30] John J. Craig. *Introduction to Robotics*. Addison Wesley, 1989.
- [31] Leonard David. Opportunity mars rover stuck in sand. Space.com, 2005. http://www.space.com/missionlaunches/050428_rover_update.html.
- [32] Kevin Dowling. *Limbless Locomotion: Learning to Crawl with a Snake Robot*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1997.
- [33] E.Papadopoulos and S.Dubowsky. On the nature of control algorithms for free-floating space manipulators. *IEEE Transactions on Robotics and Automation*, 7(6):750–758, 1991.
- [34] F.Bullo and A.D.Lewis. Kinematic controllability and motion planning for the snakeboard. *IEEE Transactions on Robotics and Automation*, 19(3):494–498, June 2003.
- [35] Chris Fernandes, Leonid Gurvits, and Zexiang Li. Near optimal nonholonomic motion planning for a system of coupled rigid bodies. *IEEE Transactions on Automatic Control*, 39(3), 1994.

- [36] K. Fujiwara, Fumio Kanehiro, Shuuji Kajita, Kazuhito Yokoi, Hajime Saito, Kensuke Harada, Kenji Kaneko, and Hirohisa Hirukawa. The first human-size humanoid that can fall over safely and stand up again. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 1920–1926, 2003.
- [37] Robert J. Full and Michael S. Tu. Mechanics of a rapid running insect: Two-, four- and six-legged locomotion. *Journal of Experimental Biology*, pages 215–231, September 1991.
- [38] G.Endo, K.Togawa, and S. Hirose. A self-contained and terrain-adaptive active cord mechanism. *Advanced Robotics*, 13(3), 1999.
- [39] G.Walsh, A. Sarti, and S. Sastry. Algorithms for steering on the group of rotations. In *The proceedings of the American Control Conference*, 1993.
- [40] Eric Hale, Nathan Schar, Joel Burdick, and Paolo Fiorini. A minimally actuated hopping robot for exploration of celestial bodies. In *Proceedings of the IEEE International Conference On Robotics and Automation*, 2000.
- [41] A. Jain and G. Rodriguez. Linearization of manipulator dynamics using spatial operators. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):239–248, Jan/Feb 1993.
- [42] J.Carlson and R.R.Murphy. How uavs physically fail in the field. *IEEE Transactions on Robotics and Automation*, 21(3):pp.423–437, 2005.
- [43] F. Jean. The car with n trailers: characterization of the singular configurations. In *ESAIM: COCV*, volume 1, 1996. <http://www.emath.fr/cocv/>.
- [44] P. S. Krishnaprasad and Dimitris P. Tsakiris. Oscillations, SE(2)- snakes and motion control: A study of the roller racer. Technical report, Institute for Systems Research, University of Maryland, 1998.
- [45] J.P. Laumond. *Robot Motion Planning and Control*. Springer, 1998.
- [46] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *International Journal of Robotics Research*, 20(9):729–752, September 2001.
- [47] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

- [48] Andrew Lewis, Jim Ostrowski, Richard Murray, and Joel Burdick. Nonholonomic mechanics and locomotion: The snakeboard example. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2391–2397, 1994.
- [49] Andrew D. Lewis and Richard M. Murray. Configuration controllability of simple mechanical control systems. *SIAM Journal on Control and Optimization*, 35(3):766–790, 1997.
- [50] Zexiang Li and John Canny. Motion of two rigid bodies with rolling constraint. *IEEE Transactions on Robotics and Automation*, 6(1):62–72, Feb. 1990.
- [51] R. Linnemann, B. Klaassen, and F. Kirchner. Walking robot scorpion—experiences with a full parametric model. In *Modeling and Simulation*, pages 1012–1018, 2001.
- [52] F. Luca and J.-J. Risler. The maximum of the degree of nonholonomy for the car with n trailers. In *Proceedings of the IFAC Symposium on Robot Control*, pages 165–170, 1994.
- [53] Matthew T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, 2001.
- [54] T. A. McMahon. Mechanics of locomotion. *The International Journal of Robotics Research*, 3(2):4–28, 1984.
- [55] David J. Montana. The kinematics of contact and grasp. *The International Journal of Robotics Research*, 7(3):17–32, June 1988.
- [56] John J. Murray and Daniel W. Johnson. The linearized dynamic robot model: Efficient computation and practical applications. In *Proceedings of the 28th Conference on Decision and Control*, 1989.
- [57] R. M. Murray, Z. X. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.
- [58] Ken’ichiro Nagasaka, Yoshihiro Kuroki, Shin’ya Suzuki, Yoshihiro Itoh, and Jin’ichi Yamaguchi. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. In *Proc. 9th RSJ/JSME Robotics Symposia of Japan*, pages 386–391, 2004.
- [59] James P. Ostrowski. *The Mechanics and Control of Undulatory Robotic Locomotion*. PhD thesis, California Institute of Technology, 1996.
- [60] Marc H. Raibert. *Legged Robots that Balance*. The MIT Press, Cambridge, Massachusetts, 1986.

- [61] J. A. Reeds and R. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2), 1990.
- [62] Dan Reznik and John Canny. A flat rigid plate is a universal planar manipulator. In *International Conference on Robotics and Animation*, 1998.
- [63] R.J.Full and Daniel E. Koditschek. Templates and anchors: Neuromechanical hypothesis of legged locomotion on land. *The Journal of Experimental Biology*, 202:3325–3332, 1999.
- [64] R.J.Full, Timothy Kubow, J. Schmitt, P. Holmes, and Daniel Koditschek. Quantifying dynamic stability and maneuverability in legged locomotion. *Integrative and Comparative Biology*, 42:149–157, 2002.
- [65] Chunlei Rui, Ilya V. Kolmanovsky, and N. Harris McClamroch. Nonlinear attitude and shape control of spacecraft with articulated appendages and reaction wheels. *IEEE Transactions on Automatic Control*, 45(8):1455–69, Aug. 2000.
- [66] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. Intelligent asimo: System overview and integration. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 2478–2483, 2002.
- [67] U. Saranli, W. J. Schwind, and D. E. Koditschek. Toward the control of multi-jointed monoped runners. In *International Conference on Robotics and Automation in Space (i-SAIRAS)*, pages 2676–2682, 1998.
- [68] Uluc Saranli, Martin Buehler, and Daniel E. Koditschek. RHex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, 20(7):616–631, July 2001.
- [69] Uluc Saranli and D.E. Koditschek. Back flips with a hexapedal robot. In *Proceedings of the IEEE International Conference On Robotics and Automation*, volume 3, pages 2209–2215, 2002.
- [70] J. Schmitt and P. Holmes. Mechanical models for insect locomotion: Dynamics and stability in the horizontal plane i. *Theory of Biology and Cybernetics*, 83:501–515, 2000.
- [71] Elie Shammas. *Generalized Motion Planning for Underactuated Mechanical Systems*. PhD thesis, Carnegie Mellon University, 2006.

- [72] Elie Shamma, Howie Choset, and Alfred Rizzi. Natural gait generation techniques for principally kinematic mechanical systems. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.
- [73] M.W. Spong. Swing up control of the acrobot. In *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pages pp2356–2361, 1994.
- [74] H.J. Sussmann and W. Tang. Shortest paths for the reeds-shepp car : a worked out example of the use of geometric techniques in nonlinear optimal control. Technical Report Report SYCON-91-10, Rutgers University, 1991.
- [75] R. S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [76] E. Tunstel. Evolution of autonomous self-righting behaviors for articulated nanorovers. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, pages 341–346, 1999.
- [77] Christopher Urmson, Joshua Anhalt, Michael Clark, Tugrul Galatali, Juan Pablo Gonzalez, Jay Gowdy, Alexander Gutierrez, Sam Harbaugh, Matthew Johnson-Roberson, Hiroki Kato, Phillip L Koon, Kevin Peterson, Bryon K Smith, Spencer Spiker, Erick Tryzelaar, and William Red L. Whittaker. High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. Technical Report CMU-RI-TR-04-37, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2004.
- [78] Manuela Veloso, William Uther, Masahiro Fujita, Minoru Asada, and Hiroaki Kitano. Playing soccer with legged robots. In *Proceedings of The Intelligent Robots and Systems Conference*, 1998.
- [79] V. Verma, G. Gordon, and R. Simmons. Efficient monitoring for planetary rovers. In *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.
- [80] Gregory C. Walsh and S. Shankar Sastry. On reorienting linked rigid bodies using internal motions. *IEEE Transactions on Robotics and Automation*, 11(1), 1995.
- [81] Mark Yim. *Locomotion with a unit-modular reconfigurable robot*. PhD thesis, Department of Mechanical Engineering, Stanford University, 1994.

- [82] Garth Zeglin. *The Bow Leg Hopping Robot*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1999.
- [83] D. Zenkov, A. Bloch, and J. Marsden. The energy-momentum method for the stability of nonholonomic systems. Technical report, California Institute of Technology, 1997.