A DATA-DRIVEN HIERARCHICAL FRAMEWORK FOR PLANNING, NAVIGATION, AND CONTROL OF UNCERTAIN SYSTEMS: APPLICATIONS TO MINIATURE LEGGED ROBOTS

by

Konstantinos Karydis

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mechanical Engineering

Fall 2015

A DATA-DRIVEN HIERARCHICAL FRAMEWORK FOR PLANNING, NAVIGATION, AND CONTROL OF UNCERTAIN SYSTEMS: APPLICATIONS TO MINIATURE LEGGED ROBOTS

by

Konstantinos Karydis

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________
Herbert G. Tanner, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________
Ioannis Poulakakis, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________
Jeffrey Heinz, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ______________________________
Christopher Rasmussen, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

Guoquan Huang, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____

Sunil Agrawal, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

First and foremost, I thank my advisors Dr. Bert Tanner and Dr. Ioannis Poulakakis for supporting and guiding me during my doctoral studies. Over the course of these past five years, they provided me with numerous opportunities to acquire and develop a broad set of skills, granted me access to all the tools I needed to perform my research, and they offered me targeted advice when appropriate. I am ingenuously thankful for Dr. Tanner's efforts to instill in me a strong interdisciplinary academic profile, and Dr. Poulakakis's provocative discussions and questions. All I have learned through their guidance and support will be extremely helpful as I further my professional career.

I am grateful to the faculty who also mentored and supported me during my studies. I thank Dr. Jeffrey Heinz for serving in my committee, working closely together, and sharing his knowledge on computational linguistics, as well as my committee members Dr. Christopher Rasmussen, Dr. Guoquan Huang, and Dr. Sunil Agrawal for the time and effort invested in my research work, and their insightful comments that had a major positive impact on improving this dissertation. I express my gratitude to Dr. Daniel Freeman for introducing me to the world of entrepreneurship and technology innovation. I am confident that this experience will complement my academic training and serve me well in my future endeavors, academic or not. I am also pleased that this work has led to several collaborations outside the University of Delaware, and specifically with Dr. Vijay Kumar and his group at University of Pennsylvania, Dr. Ron Fearing and his group at University of California at Berkeley, as well as Dr. David Zarrouk at Ben Gurion University.

This work would not have been possible without the generous financial support by our funding agencies, and the University. Specifically, this work was supported by the National Science Foundation under grants CNS-1035577, CMMI-1130372, IIS-1350721, and the Army Research Laboratory under grant MAST CTA # W911NF-08-2-0004. I also wish to thank the National Science Foundation for awarding us the travel grant IIS-1546663, which supported participation for me and other researchers and students in the workshop on Miniature Legged Robots which I co-organized. In addition, I am grateful to the Department of Mechanical Engineering for supporting my first year of graduate studies through the Helwig Fellowship in Mechanical Engineering, and a semester of my studies through a teaching assistantship.

On a personal note, I am honored to be part of a loving and caring family that has been selflessly supporting and encouraging me in good and bad times. I deeply appreciate that—although it is very difficult for them to see me only once per year—they kept on supporting my choices and encouraging me from distance throughout this whole endeavor. I can only hope to instill in my own children the devotion and love my family has shown to me.

For their constant support I thank my close friends back in Greece (currently scattered all over Europe). Although we do not get to see each other often, our friendship remains stronger than ever. During my doctoral studies in the Unites States I have also had the pleasure to meet many people from different backgrounds and make new friendships. I wish these friendships will hold in time.

Finally, I wish to thank wholeheartedly my partner Elena. Thank you for being in my life. This dissertation is dedicated to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Performing navigation with state-of-the-art mobile robots in real-world settings is challenging because of, among other reasons, the presence of uncertainty. Dealing with uncertainty in robot navigation is a difficult problem because it invalidates the performance guarantees achieved in deterministic settings, while its precise effect on motion cannot be predicted. Typically, we find uncertainty embedded within the system ("process uncertainty"), in perception, and in the environment. This dissertation develops tools for dealing with process uncertainty. The developed tools lay the basis for a general framework that can be used to quantify the effect of process uncertainty on robot motion, and recover some performance guarantees for achieving motion tasks. If we could capture the variability in motion caused by process uncertainty, quantify risk, and establish performance trade-offs in its presence, we could then create consistent links between high-level objectives and low-level implementation. Such links would allow for robot navigation in real-world settings with performance certificates, a need that becomes pressing as robotics is rapidly gaining momentum in consumer applications. Dealing with uncertainty is not only important in robotics but also in general cyber-physical and biological systems; elements of this work may find applications in these domains as well.

We follow a data-driven hierarchical control framework to address the need for consistent links between high-level objectives and low-level implementation. The framework has three levels: low-level control, high-level task planning, and mid-level motion generation. The benefit of the hierarchical approach is that it breaks the problem into smaller sub-parts that can be tackled more easily, and allows for available techniques at the two ends to be bridged. However, we still need to ensure compatibility and reconcile the two ends in face of uncertainty. To tackle these challenges, we focus on the mid level and propose two new important components: first, simple abstract models ("templates"), which are data-driven, ensure compatibility; second, a data-driven probabilistic framework recovers some guarantees that the policies prescribed in the high (cyber) level are implementable in the low (physical) level. As it turns out, templates must be reconciled with the physics of the system through experimental data, and this is the key to achieving consistency under uncertainty.

The main ideas of the approach are fixed using a particular application area: miniature legged robots. The reduction in scale magnifies the effect of uncertainty, and thus miniature legged robots provide a suitable testbed for the proposed framework; indeed, uncertainty enters naturally (e.g., inherent uncertain leg-ground interactions), while its effect on robot motion is clearly visible. By applying the framework to this area, we enable real-time robot navigation and control at the miniature scale. The latter pushes the limits of what palm-sized crawling robots can achieve, and aids in shaping their potential in applications including building/pipe inspection, search-and-rescue, and wildlife monitoring. Overall, we provide key components of a hierarchical framework that can potentially be used for approaching more general problems of planning, navigation, and control in the presence of uncertainty. The novel components are: (i) consistent data-driven templates that ensure compatibility among the different levels of the framework; and (ii) a probabilistic framework that reconciles high-level task planners and low-level motion controllers in the presence of uncertainty. Together, these components advance the state-of-the-art in planning, navigation, and control at small scales under uncertainty, and when applied to the realm of miniature legged robots, they offer tangible benefits with regards to motion capabilities for such platforms.

Chapter 1
INTRODUCTION

Contemporary mobile robots are tasked to operate and perform reliably in real-world settings in the context of applications ranging from advanced consumer electronics to robotic first responders. As robots begin to venture outside the controlled lab settings, uncertainty becomes an important determinant of behavior, affecting robot motion and compromising successful operation. It is thus important to develop tools to deal with the uncertainty in robot motion planning, navigation, and control timely and efficiently.

Dealing with uncertainty is crucial since we find it in all real-world applications. There are many cases where we cannot simply neglect or filter it out. Yet, we can still harness uncertainty and use it constructively for enhancing robot performance and supporting robot operation in real-world settings. Unfortunately, this turns out to be a highly non-trivial task, mainly because the uncertainty affects robot operation in multiple ways.

Robot operation involves the interaction of multiple components, primarily action, perception, and the environment—see Figure 1.1—each entailing different aspects of uncertainty. Uncertainty in action, often called *process uncertainty*,[1] is found in the actual motion of the physical platform, and is captured in the models used to describe that motion. Process uncertainty stems from intrinsic platform uncertainties (e.g., manufacturing variabilities, or uncertain material properties), unmodeled dynamics (e.g., depleting battery effects, or mechanical wear), as well as uncertain physical interactions with the environment the system operates in—extrinsic uncertainties—(e.g.,

[1] Hereafter we shall use these two terms interchangeably.

walking on pebbles and sand, or flying under the effect of gusts). On the other hand, uncertainty in perception relates to the sensing capabilities endowed to the robotic platform, while uncertainty in the environment is linked to conscious decision making in dynamically-changing environments. Dealing simultaneously with all three types of uncertainty is currently daunting; instead, it may be more appropriate to first improve our understanding and deal with each aspect separately, before we combine them together into a general framework.

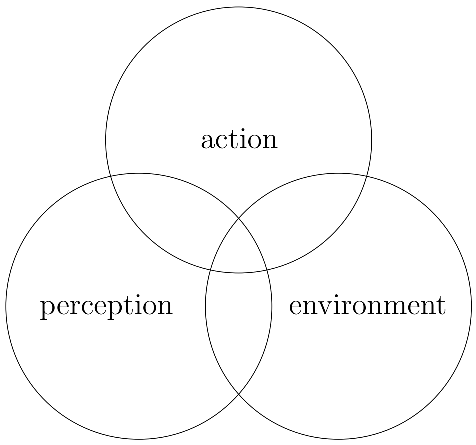


Figure 1.1: Interaction of action, perception, and the environment determine robot performance. Uncertainty is present in all three components.

The work here focuses on providing tools to capture process uncertainty in mobile robot planning, navigation and control. The rationale for this choice is that dealing with process uncertainty first provides the basis for determining "natural" robot behaviors. Then, dealing with perception and environment uncertainties can build on top of the derived tool sets for dealing with process uncertainty to further support successful autonomous robot operation in real-life (that is, dynamic) environments. Specifically, in this work we show how data from uncertain physical platforms can be used to develop tools capable of linking high-level objectives to low-level implementation in the presence of uncertainty. In turn, the developed tools can be used to capture and formally describe process uncertainty in mobile robot planning, navigation and control.

1.1 The Challenge of Dealing with Uncertainty in Robot Navigation

Dealing with process uncertainty in mobile robot planning, navigation and control is a difficult problem. First, process uncertainty arises from multiple different sources—i.e., intrinsic, extrinsic, and modeling uncertainties—and one source's effect on robot motion often cannot be studied independently of the others. In addition, the uncertainty invalidates the performance guarantees that would be available for deterministic settings, while its precise effect on motion cannot be predicted a-priori.

It is beneficial to develop a framework to quantify the effect of process uncertainty on robot motion because it may lead us to ways of recovering some performance guarantees for achieving motion tasks. Doing so would aid us in taking the execution of complex motion tasks out of the controlled lab settings and into the real world which is inherently uncertain. To be useful, the framework needs to be general enough to accommodate the multiple sources of process uncertainty, and be applicable across platforms.

1.1.1 Objective and Significance

The objective of this dissertation is thus to develop a general framework for formally capturing and reproducing process uncertainty. The framework provides tools for (i) quantifying and infusing motion uncertainty in models, (ii) restoring performance guarantees in the presence of uncertainty, and (iii) establishing tradeoffs between risk and task satisfaction. The derived tools can then be used to create consistent links between high-level task planners and low-level motion controllers. In turn, such consistent links are key for supporting robot navigation in real-life settings with performance certificates. Certifying robot navigation in real-life settings becomes pressing as robotics in consumer applications are rapidly gaining momentum. To achieve our objective we will consider a *data-driven hierarchical control framework.*

1.2 A Data-Driven Hierarchical Control Framework for Robot Navigation under Uncertainty

A promising way to perform control and navigation under uncertainty is through the application of symbolic control techniques that utilize tools from formal languages, hybrid systems, and control theory [14]. Taken together, these techniques give rise to a unified, hierarchical control framework which comprises three distinct levels: (i) the high level that accepts task objectives and generates symbolic control laws, (ii) the mid level that ensures compatibility between the generated symbolic control laws and the system dynamics, and (iii) the low level that focuses on local closed-loop control of the system.

1.2.1 Justification for a Hierarchical Approach: *Divide et Impera*

A hierarchical control approach like the one considered here has two benefits. First, it breaks the big problem into smaller subparts that can be encountered more efficiently. Secondly, it allows us to bridge available tools in the literature, and facilitates application across different platforms. This way the focus now shifts toward combining these tools to capture and subsequently harness uncertainty. The last point is in fact the main challenge that needs to be tackled when considering a hierarchical framework: the framework must be designed so that its various levels of hierarchy are compatible with each other.

1.2.2 Components of the Framework

The approach followed here is designed to ensure compatibility among levels. Each level treats different aspects of the problem, and produces outputs which subsequently serve as input to another level. Specifically, the high level uses tools from theoretical computer science for discrete task planning (and learning) to generate symbolic motion planning tasks in the form of a temporal sequence of behaviors. The mid level comprises control-oriented languages that assign to each behavior symbol in the sequence a control action realizable by the physical system. The succession of controller activations generates desired trajectories. Then, low-level navigation controllers are employed to make the physical system track to these desired trajectories.

Although this decomposition allows for tackling the uncertainty individually at each level, available approaches in the literature (e.g., [86, 29, 65]) assume specific noise statistics in models. This assumption is placed in order to facilitate analysis, yet it may be too restrictive when validating through experimental implementation. We claim that we need to resort to experimental data for deriving analytical formulations that reinforce the links between analysis and physical platforms. In turn, this calls for developing more tools to ensure consistency between levels in the presence of uncertainty—a key contribution of this dissertation.

1.2.3 Consistency Conditions and the Role of Models and Data

To ensure consistency in the presence of uncertainty we require two key conditions. Foremost, appropriate low-dimensional abstract models, often called "templates" [47], can ensure compatibility if they

- capture and reproduce salient robot motion behaviors;
- apply to different systems when operating under similar conditions; and
- facilitate both low-level control and mid-level analysis.

The second consistency condition calls for probabilistic tools that can reconcile high-level objectives with low-level realizability through

- quantifying and infusing uncertainty back into templates;
- recovering performance guarantees in face of the uncertainty; and
- establishing tradeoffs between risk and task satisfaction.

In both conditions, *data are essential.* Indeed, data are used as the basis for tuning model parameters and developing a systematic method to reinforce the connections among the three levels. Taken together, the above conditions reveal the key to achieving consistency:

> "Templates must be reconciled with the physics of the system through experimental data."

We shall use the term *probabilistically-valid templates* to refer to such templates.

1.3 Focus on Miniature Legged Robots

We fix the main ideas of the general approach using an application from the area of miniature legged robots. These robots demonstrate potential in real-world applications, and serve as testbeds for studying the effect of process uncertainty. By applying the framework to miniature legged robots, we push the limits of what they can achieve and shape their potential in applications.

1.3.1 Justification

Miniature legged robots fit within the framework proposed in this work, and in essence they can be thought of as "laboratories" to study the effect of process uncertainty. The small scale magnifies the effect of uncertainty. Compared to other small ground robots—such as wheeled ones—uncertainty in miniature legged robots enters naturally, and has clearly visible effects. Indeed, the uncertain effect of manufacturing variabilities, in conjunction with the inherently uncertain leg-ground contact interaction and exacerbated by the robot's small scale, makes miniature legged robots an interesting example of uncertain physical systems, where the same control inputs may lead to a rich set of behaviors as Figure 1.2 showcases: Depicted data correspond to the miniature legged robot OctoRoach [120] commanded to follow various curvature-parameterized trajectories in open loop for 5 sec. The path variability within each mode of operation is clearly visible.

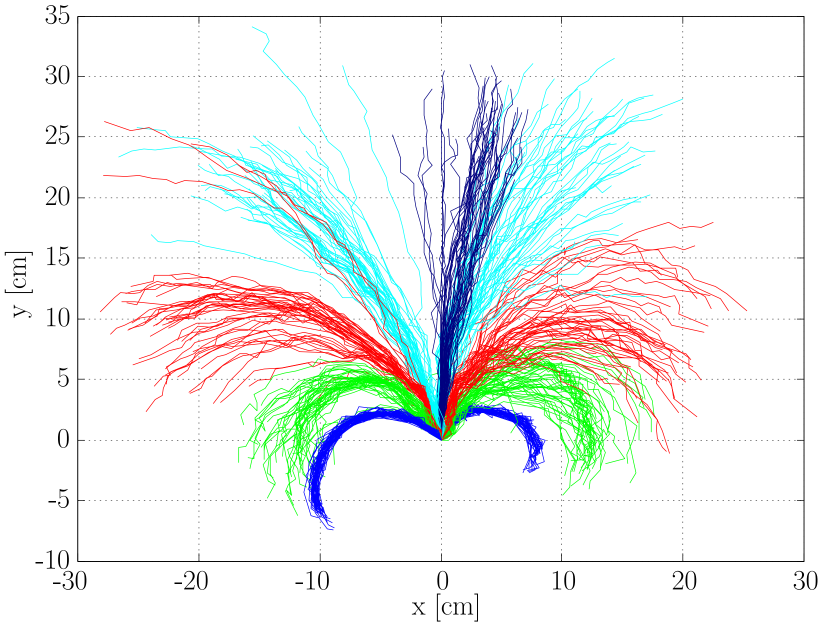


Figure 1.2: Uncertain behavior of a particular OctoRoACH [120] platform. The robot crawls at low speed to follow eleven curvature-parameterized trajectories in open loop for 5 sec. The path variability is clearly visible and consistently high although the control commands remain the same for each mode of operation, and discrepancies in initial placement between sample trials are negligible. Each mode of operation is color-coded and contains 50 trials. Data are collected via motion capture at a rate of 100 Hz.

1.3.2 Potential of Miniature Legged Robots

There is value in applying the framework to the domain of miniature legged robots as they offer multiple benefits, relative to scale. Legs support all-terrain mobility and allow these robots to overcome obstacles and move in confined spaces in ways impossible for their wheeled counterparts (e.g., [105, 92]), while low production cost and rapid manufacturing enable deployment in large numbers. Essentially, miniature crawlers can be used to explore, manipulate, and act as remote sensors in a range of real-world applications such as search-and-rescue, building and pipe inspection, unobtrusive wildlife monitoring, and Intelligence, Surveillance, and Reconnaissance (ISR), eventually involving multiple robots; Figure 1.3 depicts two representative cases.

(a)

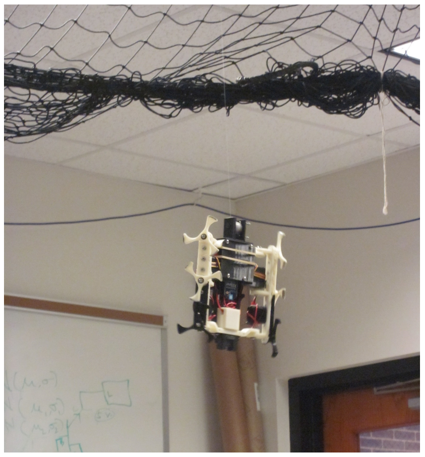
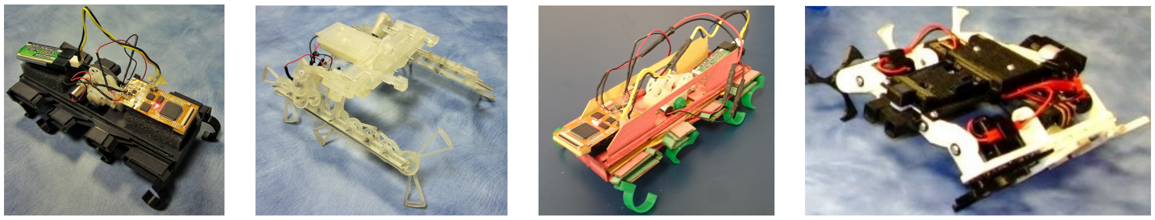

(b)

Figure 1.3: Examples of the domain of application of miniature legged robots. (a) Pipe inspection. (b) Building surveillance and intelligence.

The potential of miniature legged robots has been acknowledged recently. Fueled by advances in novel manufacturing processes, it is rapidly gaining momentum as indicated by the growing number of biologically-inspired robot designs. Some examples include a cockroach-inspired hexapod [158] which uses two piezoelectric ceramic actuators to drive its legs, the Mini-Whegs robot series [106, 87] utilizing a three-spoke rimless wheel ("wheg") design, and a miniature direct-drive legged robot [21] with similar leg design. Moving a step forward, the 3D-printed STAR robot [161] incorporates a mechanism that adjusts the sprawl angle of the robot, while PSR [144]—similar in nature to STAR—employs a passive sprawl-adjusting mechanism. On a different vein we find the Sprawlita [25] and i-Sprawl [77] robots, manufactured via the Shape Deposition Manufacturing (SDM) process [103, 15], and origami-inspired printable robots [109] System integration at very small scales has been made possible through the Smart Composite Microstructure (SCM) fabrication technique [156]; typical examples include HAMR [13, 12], RoACH [60], and Medic [82]. The SCM process has been also used to fabricate minimally actuated palm-sized walking robots, including the hexapod crawlers DASH [16], DynaRoACH [59], VelociRoACH [52], and OpenRoACH [51] as well as the eight legged robot OctoRoACH [120]. Figure 1.4 presents the robots we consider here.



(a) (b) (c) (d)

Figure 1.4: A family of miniature legged robots studied in this work. (a) The eight-legged OctoRoACH [120], and (b) the 3D-printed STAR [161]. Both robots are designed and manufactured at University of California, Berkeley. Inspired by these designs, we redesigned and manufactured in-house (c) a revamped OctoRoACH able to carry more payload, and (d) SPIDAR, a heavy-duty, highly-versatile miniature legged robot.

As robot capabilities increase, the introduction and implementation of navigation and control techniques at this scale will expand the capacity of these miniature robots in real-world applications.

1.3.3 Challenges for Effective Navigation and Control at Small Scales

To develop and exploit the capabilities of miniature legged robots, we need to bring together the areas of Biological Inspiration, unconventional fabrication methods, as well as navigation and control at small scales. Unfortunately, the advances in our understanding of legged locomotion at small scales [143, 162, 93, 31, 63], as well as the advent of novel manufacturing processes [103, 15, 156, 21, 109, 51], and the introduction of a number of platforms have not been matched by a solid understanding on how navigation, planning, and control can be realized on these miniature crawlers.

The challenge in performing navigation and control at small scales can be attributed to two main factors. Foremost, stringent size and weight specifications constrain the number and power density of the actuators. Consequently, this leads to heavily underactuated designs, prohibits extensive reliance on feedback control to mitigate the effect of process uncertainty [59], and restricts power autonomy by imposing strict bounds on control effort. The second factor is that developing detailed analytical models for miniature legged robots that can be used for planning, navigation, and control is challenging. Indeed, the novel manufacturing processes employed to fabricate such platforms lead to complex highly-articulated designs, with involved transmission and actuation mechanisms.[2] Detailed modeling is also hindered by the uncertain mechanical properties of the structural material and the various manufacturing variabilities introduced by the relatively low precision manufacturing and assembling processes. The latter also leads to poor mobility performance [48].

Because of the aforementioned challenges, there is only a limited number of attempts to perform navigation and control at the miniature scale [101, 136]. One way to mitigate constraints of computational nature is by performing some of the demanding processing off-line [101]. While essentially circumventing the problem of limited on-board computational capacity, this approach shifts some of the challenges to communication, thus hindering real-time execution. Workarounds exist [136]; after substantial off-line pre-computations, the actual control action can be computed on-board in real time. Unlike aforementioned approaches [101], the latter also offers probabilistic performance guarantees assuming unbounded control effort, without temporal limits on when the task should be completed. However, the amount of pre-computations needed prior to deployment may limit the practical utility of this approach. Notwithstanding these recent efforts, effective real-time navigation and control at small scales under the perceived challenges is still elusive. The work here contributes to narrowing this gap.

1.4 Contributions

Overall, this dissertation provides key components of a hierarchical framework, which when completed, form the foundation for approaching more general problems of planning and control in the presence of uncertainty. The focus here is mostly on developing tools in the mid level that can be used to reinforce the connections among the high and low levels.

[2] See for example the transmission of OctoRoach [120, Fig. 2].

First, it is demonstrated that consistent data-driven abstract models (templates) are essential for ensuring compatibility among levels. As it turns outs, consistency is achieved when abstract models are validated against experimental data. To be useful, such models must also be able to capture and reproduce salient robot behaviors *and* facilitate analysis in support of navigation and control for uncertain systems. To show the practical utility of these notions a new kinematic template for legged locomotion at low crawling speeds is derived, and used to perform real-time motion planning, navigation, and control on miniature legged robots.

Subsequently, a novel probabilistic framework to reconcile high-level objectives with low-level realizability under uncertainty is constructed. Essentially, the derived tools generate a probabilistic matching between physical platforms and templates and report on its the validity. What is important is that the reported outcome comes with probabilistic guarantees of validity, which can be used to establish tradeoffs between risk and task satisfaction. These guarantees also render the aforementioned consistent templates *probabilistically-valid*, which is key to ensure consistency and recover some performance guarantees under uncertainty. All these notions are grounded in the application domain of miniature legged robots.

Last but not least, application of these tools to the area of miniature legged robots advances the state-of-the-art in planning, navigation, and control of miniature legged robots. The tools are general and can be applied to multiple platforms; we show this point by applying them to several miniature legged robots.

1.5 Intellectual Merits and Broader Impact

Dealing with uncertainty is not only important in robotics but also in more general cyber-physical and biological systems; elements of this work may find application in these domains as well. Specifically, the tools provided here are directly applicable to capturing process uncertainty in other robotic systems, such as small-scale aerial vehicles or larger legged robots. This information can then be used to harness uncertainty in support of motion planning, navigation, and control toward resilient multi-robot

Cyber-Physical Systems (CPSs). Furthermore, developing a better understanding of how control can be effectively applied at the miniature scale—given the constraints and limitations on payload, energy utilization, and computational power—is very likely to drive design decisions [144], and possibly feed back to biology by shedding light on the tradeoffs in small animal locomotion [31, 63].

1.6 Dissertation Layout

The remainder of this dissertation is organized as follows. Chapter 2 provides a technical description of the proposed hierarchical control framework, an overview of the existing tools that can be used at each level, and justifies the need to develop more tools to ensure consistency among levels in support of motion planning, navigation, and control of mobile robots under uncertainty. To ensure consistency, low-dimensional behavioral models (templates) are combined with novel probabilistic tools that harness uncertainty to create probabilistically-valid templates. To this end, Chapter 3 emphasizes the importance of templates, reports on a newly-developed template for low-speed crawling locomotion, and shows how this template facilitates navigation and control at small scales. Then, Chapter 4 reports on the novel probabilistic tools, and their application to extending deterministic models into stochastic ones with probabilistic guarantees of validity. Finally, Chapter 5 summarizes the contributions of this work, and presents several future research directions that the research efforts herein have enabled.

Chapter 2
ELEMENTS OF THE HIERARCHICAL CONTROL FRAMEWORK

This dissertation employs a data-driven hierarchical control framework for robot navigation under uncertainty. The hierarchy has three levels. The high level is responsible for discrete task planning; the low level focuses on deriving system-specific controllers; the mid level provides *tools for ensuring compatibility and consistency* between the high and low levels. This decomposition breaks the problem into smaller subproblems for which we have a relatively good idea about how to address with available tools in the literature. In what follows, we provide an overview of some of the most representative of these tools, and identify missing elements in the framework that this work provides.

2.1 High-Level Discrete Task Planning

The high level features task planners that derive symbolic motion control policies for complex navigation tasks. Consider for example a *surveillance* task: "Visit the waypoints A, B, and C infinitely often while avoiding the obstacles O_1 and O_2;" or a *reconnaissance* task: "Explore area E and return to base while avoiding any obstacles encountered." These tasks are described as high-level natural language commands, and need to be translated mathematically into some computationally tractable form, utilizing symbols for expressing actions ("go to point A"), events ("if an object is encountered"), and conditions ("while moving, take laser scans").

One way to express such tasks and behaviors in a computationally tractable form, inspired by developments in theoretical computer science, is through the use of temporal logic formulae [42]. Temporal logic, a subset of modal logic, was established to "provide a formal system for qualitatively describing and reasoning about how the truth values of assertions change over time" [42]. Intuitively, these formulae consist of Boolean operators, atomic propositions, and temporal operators that can be connected according to some syntax rules, and were first introduced for assessing the correctness of computer programs [113]. Using temporal logics, automata theory, and formal languages [139] we can reason about the temporal properties of discrete models of physical processes (automata for instance). In the context of expressing complex robot tasks, the states of an automaton may represent atomic propositions (e.g., "point A reached"), while transitions between states may reflect the control actions required by the robot (e.g., "go to").

The above notions are also used in designing appropriate task planners. The most popular approach involves the utilization of model checking [30, 11] for high-level robot motion and task planning [79, 43, 96, 126, 155]. Some recent advancements in this area include environment-reactive control [85], multi-robot distributed control [80], and planning under sensor and actuator uncertainty by modeling the system as a Markov Decision Process (MDP) while employing Probabilistic Computation Tree Logic (PCTL) [86]. Other approaches based on formal languages include the use of control-oriented languages and Dynamic Programming (DP) [8], and the Motion Grammar (MG) framework [34], which uses Context-Free Grammars (CFG) to represent planning and control of robotic systems with uncertainty in the outcomes of control actions. Alternative design tools for planning in adversarial environments [26, 46] utilize machine learning techniques that draw from game theory and grammatical inference [36].

What is common in all methodologies of this spirit is that they produce symbolic, discrete control plans. The realizability and applicability of these plans on the physical platform is implicitly assumed. Notwithstanding, they can be applied to special types of systems with rather simple dynamics (such as affine control and unicycle systems). The reason is that systems with such dynamics can be "abstracted" intuitively at a discrete level—that is, the low-level dynamics can be expressed in the form of a transition system.[1] However, in order for the derived high-level policies to find real-world applicability, they must be grounded to particular physical systems of interest through platform-compatible controllers and experimental data. This is exactly what the low level in our hierarchical architecture is tasked to do.

2.2 Low-Level Control

The low level centers on deriving control laws that are applicable to the physical system at hand. Approaches in this realm focus on stabilization [142] or path and trajectory tracking [3], and can be derived based on a variety of methods including potential fields [84] and navigation functions [81], principles of stochastic control design [7], robust control [45], optimal control [94], model predictive control (MPC) [124], or approaches that integrate the above into a single framework [137, 135, 136]. The resulting *navigation* controllers are able to steer the system around obstacles, and through waypoints.

However, these navigation controllers are typically platform– and task-specific, while methods based on robust control—that is often employed in the study of systems with uncertainty—may be overly conservative. In addition, the low-level controllers may be too restrictive relative to the full spectrum of tasks mandated by the high-level task planners. To tackle these limitations, low-level controllers should be put together in a way that ensures compatibility and consistency between the low and high levels; this is what happens at the mid level of the hierarchy.

2.3 Mid-level Integration

The mid level provides a series of tools that can bridge the gap between high-level planning and low-level control. In detail, symbolic high-level controllers—that respect the mechanical, control, sensing and communication constraints of the system—can be constructed through a control-oriented language capable of handling symbolic control

[1] Such systems are formally characterized by the existence of the bisimulation and approximate bisimulation properties [30, 104, 6, 114].

plans that are expressed in the form of a string. Such languages include Computation and Control Language (CCLs) [78] that accommodate for multi-robot cases, and Maneuver Automata (MA) [44] that offer an example of using regular languages to solve complex motion planning tasks and can be viewed as a subset of the extended Motion Description Language (MDLe) [62]. The latter forms a very general framework for encoding symbolic plans. Powerful motion planning techniques [28, 88] such as the sampling-based Rapidly-exploring Random Trees (RRT) [89] and Probabilistic Roadmaps (PRM) [75] can be considered as special cases of the MDL framework.

2.3.1 Control and Planning through Motion Primitives

The Motion Description Languages framework gives rise to a multimodal control strategy that defines different modes of operation with respect to a particular task. The underlying idea is to break a global plan down into behavioral sub-plans (*motion primitives*) that can be attained by the low-level elementary controllers. Generating the associated motion primitives depends heavily on the system at hand, and can be done in several ways. Although a general formulation that facilitates portability between different platforms is still elusive, there exist some promising methodologies. For example, motion primitives for acrobatic helicopters can be generated by on-board controllers [44], while experimental data from ants produce motion primitives that are then being applied to drive unicycle systems [38]. If a model for the platform exists, an alternative way lies on optimal control design [100]. Irrespectively of their generation method, motion primitives considerably simplify the overall planning and control problem since, instead of having to account for constraints in the high-level, each motion primitive is guaranteed to satisfy the kinematic and dynamic constraints of a system.

Once the motion primitives are available, they are then combined together according to some discrete switching logic (interrupt functions) in order to achieve the global objective. Consider for example primitives of the form (π_i, g_i), where π_i correspond to low-level elementary controllers, and $g_i \in \{0, 1\}$ are binary interrupt functions.

Intuitively, the system is being controlled by π_i until the interrupt function g_i changes from 0 to 1 (e.g., a target was reached, or the system failed to reach a target after 10 seconds); subsequently, the next control law π_{i+1} initiates. The timing of the interrupts may depend on various design parameters, such as time instances of exiting a region, reaching a bound of a cost function, and others. Concatenation of primitives yields control plans or "programs" that can be executed by the system. These plans are strings, or words in the MDL.

Existing work that falls within the general context of the above framework tends to revolve around two fundamentally complementary views: bottom-up, and top-down design approaches. Top-down approaches (e.g., [86, 29, 65]) begin with high-level specifications, and synthesize controllers in the presence of uncertainty with some probabilities of successful task execution on actual hardware. On the other hand, bottom-up techniques [24, 17, 151] begin with the system dynamics, and focus on deriving dynamically-feasible trajectories that satisfy high-level reachability objectives under the presence of uncertainty, with user-defined probabilistic guarantees.

2.3.2 What is Missing?

Irrespectively of the different methodological approach they employ, existing techniques uniformly assume specific noise statistics in the models employed. This assumption facilitates analysis, yet it may be too restrictive in experimental testing and validation. The argument put forward in this dissertation is that analytical formulations should be driven and shaped by experimental data in order to reinforce the links between theoretical analysis and physical behaviors. In turn, this motivates the development of novel tools for quantifying the uncertainty within models as it is observed in real-world datasets.

2.3.3 Need for Appropriate Models and Data to Achieve Consistency

The results reported herein equip the mid level with tools for reconciling the two ends of the spectrum in this hierarchical framework, despite the presence of uncertainty.

In particular, the thesis in this dissertation is that suitable simple models (*templates*) can ensure consistency between abstract symbolic plans and actual physical dynamics, provided that data have been employed to ground these models to physical platforms of interest.

Simple Models Ensure Realizability without Oversimplification

Carefully-crafted yet simple models can ensure realizability of high-level policies on low-level hardware without oversimplifying the system's dynamics. This is achieved if models are (i) general and suited to morphologically distinct platforms; (ii) compatible with a Control-Orientated Languages framework, facilitating the design of motion primitives; and (iii) capable of capturing the variability within individual system behaviors such as crawling or running. Note that the last point essentially means that such models must be validated against experimental data; the work here provides a systematic framework to accomplish this.

A Data-Driven Probabilistic Framework Removes Conservatism

A data-driven probabilistic framework is developed to remove conservatism by jointly validating models and extending them to stochastic regimes. In other words, the observed process uncertainty is infused into the model so that the latter can capture and reproduce the variability that is observed in practice within a certain behavior of the system. The reported method provides probabilistic guarantees of implementing high-level policies to the physical system based on the *actual* uncertainty exhibited by the system.

2.4 Discussion

Overall, the three levels considered here naturally complement each other (see Figure 2.1). Each level treats different aspects of the problem, and produces outputs which subsequently serve as input to another level. This decomposition allows for tackling the uncertainty individually at each level, and bridges different available techniques. The dissertation contributes to narrowing the gap in ensuring consistency among levels by combining templates with novel probabilistic tools that harness uncertainty to create probabilistically-valid templates. In turn, probabilistically-valid templates can be used to recover performance guarantees in face of the uncertainty, and establish tradeoffs between risk and task satisfaction, based on the uncertainty exhibited by the real system.

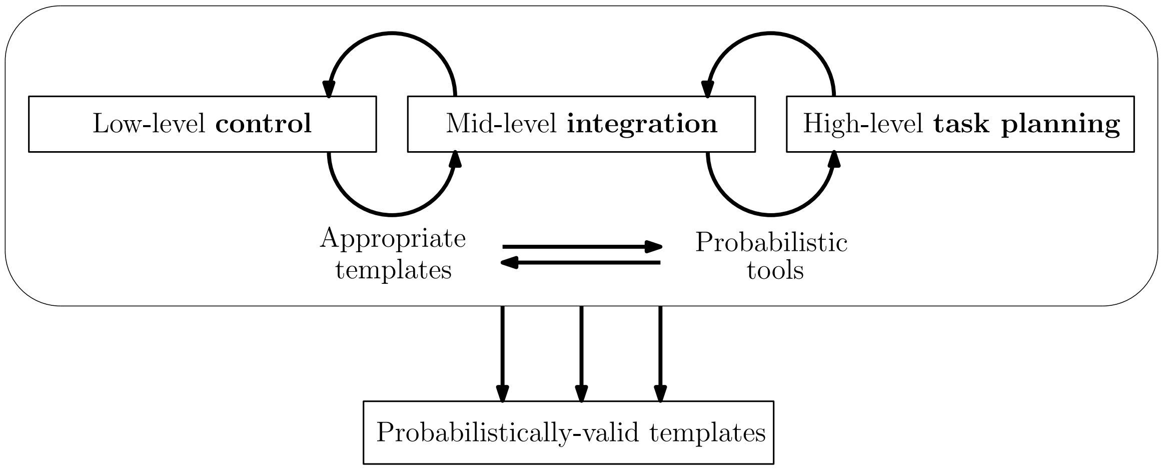

Low-level **control**
Mid-level **integration**
High-level **task planning**
Appropriate templates
Probabilistic tools
Probabilistically-valid templates

Figure 2.1: Schematic representation of the hierarchical control framework followed in this work. The levels of hierarchy complement each other. This dissertation provides results that ensure consistency among levels by introducing and employing *probabilistically-valid templates*.

The following chapters provide details on the technical approach for building the middle layer by exemplifying the steps of the method using *miniature legged robots*. Chapter 3 treats templates and demonstrates their use in navigation and control in small scales, while Chapter 4 focuses on the developed probabilistic tools for quantifying uncertainty.

Chapter 3

TEMPLATES IN ROBOT MOTION PLANNING, NAVIGATION, AND CONTROL

Loosely speaking, the term *template* [47] is used to describe the simplest possible model that captures and reproduces behaviors of interest, and applies to different robotic systems when operating under similar conditions. Compared to detailed models grounded in the morphology of the platform at hand, templates do not necessarily model exactly the mechanisms that energy is transmitted from the motors to the components that interact with the environment to generate motion (e.g., legs, wheels, arms). Instead, the purpose of a template is to abstract some of these mechanisms to an extent that facilitates analysis and control, and permits direct experimental validation.

Using templates in robotic legged locomotion is essential. Given that legged locomotion is the outcome of complex, nonlinear, and inherently uncertain interactions between the physical system and its environment, exact modeling of these interactions is a very challenging task [47]. It is also challenging to derive first-principles models that describe the mechanisms that generate motion since legged robots often rely on complex highly-articulated system designs and involved transmission and actuation mechanisms to create motion. We thus resort to extracting only some of the key features of those mechanisms to derive simpler models that exhibit salient robot behaviors.

In what follows, we present some models that have been proposed in the literature as templates for legged locomotion, and propose a new template for low-speed crawling locomotion. We then show how this new template is useful in terms of facilitating motion planning, navigation, and control for miniature legged robots.

3.1 Templates for Robotic Legged Locomotion in the Miniature Scale

3.1.1 Bio-Inspired Templates

Most bio-inspired modeling attempts concentrate on simple spring-mass systems. Their aim is to capture the underlying similarities of the center-of-mass (COM) motion in animals and robots despite their apparently diverse structural and morphological characteristics [47]. In this realm we find the Spring Loaded Inverted Pendulum (SLIP) template [18, 132] which captures the COM motion of robots (and animals) in the sagittal plane. The model was motivated by studies in diverse species of animals which suggested that, during a step, the center of mass reaches its lowest position by compressing a virtual leg spring at mid-stance, and then extends by recovering stored elastic energy [47]. Despite its apparent simplicity, SLIP and its extensions (for example the asymmetric [116] and bipedal [50, 9] SLIP) have been successfully used in a range of studies to model COM motion, and derive controllers for robots (e.g., [122, 116]). However, since most contemporary miniature legged robots such as HAMR [13, 12], DASH [16], and OctoRoACH [120] have a more sprawled than upright posture, we also need bio-inspired templates in the horizontal plane.

Research in sprawled arthropods [20, 23] has motivated the introduction of bio-inspired templates in the horizontal plane. Perhaps the most well-known template for legged locomotion in the horizontal plane is the Lateral Leg Spring (LLS) model [130, 131, 58] which has been successful in explaining lateral stabilization [134], and in deriving turning strategies [64, 119] for hexapedal runners. In its most common configuration, the LLS is a conservative mechanical system composed by a rigid torso and two prismatic legs that are modeled as massless springs. Each leg represents the collective effect of a support tripod formed during the stance phase by the front and rear ipsilateral,[1] and the contralateral middle legs that are in contact with

[1] Ipsilateral means on the same side and contralateral means on the other side. Appendix A summarizes some key terms in legged locomotion that are used here.

the ground.[2] The Sliding Spring Leg (SSL) model [160] extends LLS by incorporating the sliding effects of the leg-ground interaction that is often observed in multi-legged miniature robots. Modeling eight-legged crawlers like the OctoRoACH has received less attention in the relevant literature than hexapedal runners; besides eight-legged models for crabs [19], which typically employ a metachronal tetrapod gait to move laterally in the horizontal plane, no further analyses appear to be available.

3.1.2 Use of Car-Like Templates in Legged Locomotion

Beyond bio-inspired templates, it has been hypothesized that models developed for wheeled vehicles may also be applicable in the context of legged locomotion. In detail, [98, 35] suggest that a unicycle model may suffice for the gross description of the kinematics of multi-legged robots such as RHex [127]. Moreover, it has been shown analytically that under some simplifying assumptions, the dynamics of RHex can be reduced to that of a unicycle with a dynamic extension [110]. The validity of the aforementioned hypothesis has been also tested in smaller scales [74] for STAR [161] when the robot operates at low-sprawled postures.[3]

More generally, however, uncertainty becomes an important determinant of behavior as robots scale down in size; in this case, bio-inspired legged templates may provide better intuition than car-like templates regarding the mechanisms through which the uncertainty affects the behavior of the system. The former, feature physically-relevant parameters that can be tuned to capture or model (aspects of) uncertainty and may offer some explanation about its source. For instance, varying the touchdown and liftoff angles of a legged model may capture terrain profile variations[4]—information

[2] See [58] for a detailed description of LLS and its various configurations.

[3] STAR features a rimless wheel leg design; consequently, at sufficiently low sprawl postures, the robot's legs are in contact with the ground for longer periods of time, resulting to smooth, car-like behaviors [161]. Of course, this result may not generalize well to other miniature legged robots with different leg design, like the OctoRoACH [120].

[4] Chapter 4 reports on a new methodology for tuning model parameters to capture uncertainty.

which is important in order to design better controllers. Car-like models lack the descriptive capacity to capture such phenomena, yet they have been successfully used for deriving a vast number of approaches to perform motion planning [88, 28], navigation, and control.[5] We will take advantage of such methods by applying them to small-scale legged robots using appropriate bio-inspired templates.

3.1.3 Need for more Templates at the Miniature Scale

We focus on the particular operation regime of *crawling locomotion at low speeds*. The term essentially describes quasi-static operation regimes which are characterized by relatively slow maneuvers with no extended slippage or flight phases. Such operation regimes are important in applications; for example, if we mount a small camera on a miniature legged robot, the robot needs to crawl slowly and smoothly, and offer the system sufficient time to register and process new scenes. Besides, enabling navigation and control at these regimes can then be used as a building block for high-speed dynamic locomotion in other applications.

In the quasi-static operation regime, the motion of miniature robots is mainly dominated by surface forces instead of dynamic and inertia effects [143, 121], as it is often the case in meso– and large-scale robots. However, we currently lack detailed descriptions of ground interaction that could be incorporated into a dynamical model [91]. What is more, it remains unclear how to directly map the parameters of such models to robot design parameters [59], and control parameters such as motor gains. On the other hand, crawling at low speeds in quasi-static operation regimes is typically captured well by a kinematic model; see for instance, the kinematic model for a centipede microrobot [56]. Although this model is capable of describing the physical platform accurately, it tends to be tailored to the particular mechanism employed by this robot. Thus, there appears to be a need for more general *kinematic templates*, that can (i) capture the salient features of locomotion behaviors of multiple robots, (ii) be amenable

[5] Due to the extensive body of literature that falls under this category, we mention later in this section only those results that have been considered in this work.

to analysis and control, and (iii) provide intuitive and physically-relevant mechanisms to infuse uncertainty. To meet these challenges, we introduce a new kinematic template called Switching Four-Bar Mechanism (SFM) [68].

3.2 The Switching Four-Bar Mechanism (SFM) Template

The Switching Four-Bar Mechanism has been proposed [68] as a kinematic template for low-speed, quasi-static locomotion in the horizontal plane. The model does not rely explicitly on the detailed mechanisms by which the physical platform generates motion; it is thus capable of capturing salient behaviors of *multiple* miniature legged robots. It is also amenable to analysis as it allows *analytic*, closed-form state propagation and facilitates control by enabling real-time computations. Additionally, the SFM encompasses a small set of intuitive and *physically-relevant* parameters that can be tuned to achieve various motion patterns, and may offer explanation regarding the mechanisms through which uncertainty affects the motion of the system (see Chapter 4 for details on the last benefit of the model).

3.2.1 Description of the Model

The SFM—see Figure 3.1(a)—is a horizontal-plane model consisting of a rigid torso and four rigid legs organized in two pairs [68]. With the notation of Figure 3.1(a), d is the distance between the two hip-point joints A and B, l denotes the leg length, and G is the model's geometric center. The two leg pairs $\{AO_1, BO_2\}$, and $\{AO_3, BO_4\}$ are defined as the *right* and *left pair*, respectively.[6] The two pairs become active in turns as shown in Figure 3.1(b). We require the following design specifications.

Design Condition 1 *Once a pair of legs touches the ground (effective pair), the tips of the legs remain fixed until the other pair touches down (no slipping).*

Design Condition 2 *At any given time other than the switching instant between pairs, only one pair is active (touches the ground).*

[6] The notation follows from the leading leg at each pair.

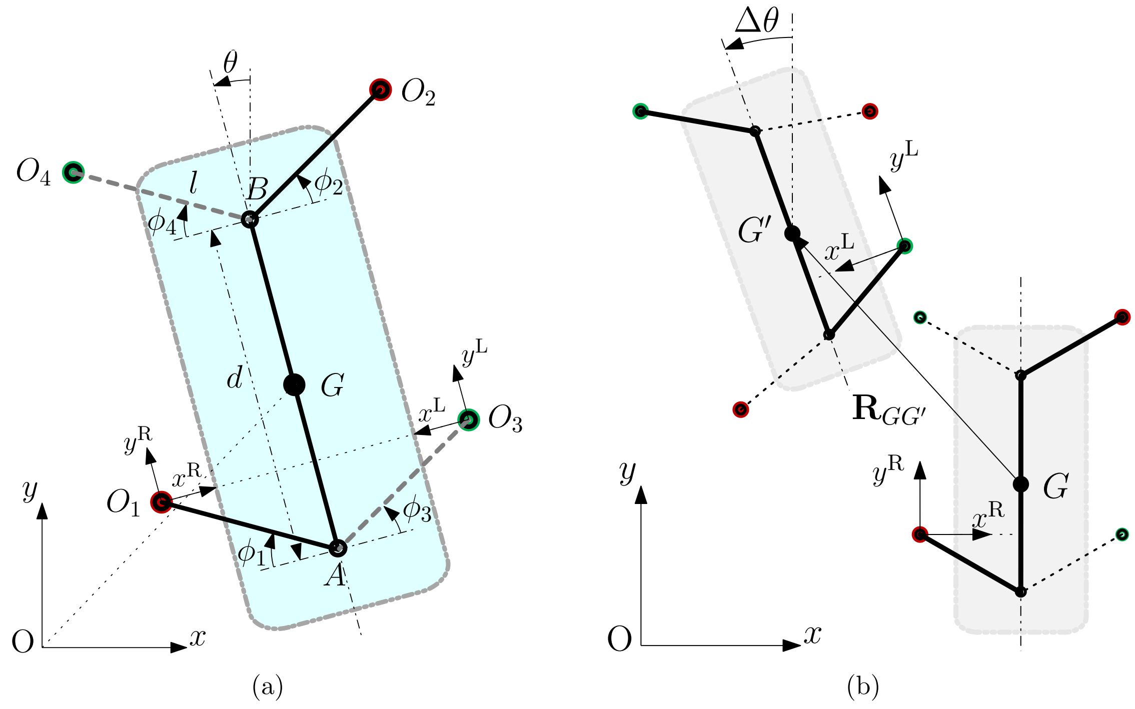


Figure 3.1: (a) The SFM template. Two pairs of rigid legs become active in turns, forming two fourbar linkages, $\{O_1A, AB, BO_2\}$ and $\{O_3A, AB, BO_4\}$. d is the distance between the two hip-point joints A and B, l denotes the leg length, and G is its geometric center. (b) Pictorial representation of state propagation—the active pair is marked with thick solid lines.

Design Condition 3 *Within a pair, a 50% duty factor*[7] *for its legs is used. Intuitively, both legs of each pair touch, and lift off the ground at the same time instant.*

The practical significance of the above design specifications is that the template can be essentially viewed as a switching four-bar mechanism (hence the name). The kinematics of each effective pair is determined by the four-bar linkage these legs represent. Due to the kinematic equivalence to a four-bar linkage, the motion at each step is fully determined by one degree of freedom: the angle ϕ_1 when the right pair is active, and the angle ϕ_3 when the left pair is active (see Figure 3.1(a)). The motion of the model is parameterized by the leg *touchdown* and *liftoff* angles ϕ_i^{td} and ϕ_i^{lo}, respectively

[7] *The duty factor refers to the percentage of the total cycle which a leg touches the ground; see Appendix A.*

(where $i = 1, \ldots, 4$), requiring $\frac{\pi}{2} \geq \phi_i^{\text{td}} \geq \phi_i^{\text{lo}} \geq -\frac{\pi}{2}$. We follow the convention that a leg angle $\phi_i \in [\phi_i^{\text{td}}, \phi_i^{\text{lo}}]$ is positive when the tip of the leg (point O_i) is in the "upper semi-plane" defined by the longitudinal axis of the model and its normal that crosses the leg's hip-point (A or B)—all angles ϕ_i are sketched positive in Figure 3.1(a). To facilitate analysis, we also define the *sweep* angle ψ_i as the absolute difference between the touchdown and liftoff angles, that is $\psi_i = |\phi_i^{\text{td}} - \phi_i^{\text{lo}}|,\ i = 1, \ldots, 4$. Due to Design Specification 3, the sweep angle takes the same value for both legs within a pair so that $\psi_1 = \psi_2 = \psi_{\text{R}}$ and $\psi_3 = \psi_4 = \psi_{\text{L}}$. Additionally, we define the leg angular velocity $\dot{\phi}_i,\ i = 1, \ldots, 4$ as a means to introduce time in the parameterization of the paths realized by the SFM.

3.2.2 Analysis and Closed-Form Expressions

The state of the model is a tuple $q = (x_G, y_G, \theta) \in \mathbb{R}^2 \times \mathbb{S}$. With reference to Figure 3.1(a), (x_G, y_G) denotes the position of the geometric center of the model, G, with respect to some global coordinate frame $\{\text{O}\}$, and θ is the orientation of the model, defined as the angle formed between the longitudinal axis of the model and the y axis of the global frame. We adopt the convention that positive changes in the orientation of the model correspond to counter-clockwise rotations.[8]

3.2.2.1 State propagation during a step in the local frame

In each active pair, the progression of the position vector $\mathbf{R}_{GG'} = (\Delta x, \Delta y)$ and orientation $\Delta\theta$ of the model during a step in the local coordinate frame—see Figure 3.1(b)—is given by

$$(\Delta x, \Delta y, \Delta\theta) = f(\phi; \xi) \ , \tag{3.1}$$

where $f : \left[\phi^{\text{td}}, \phi^{\text{lo}}\right] \times \Xi \to \mathbb{R}^2 \times \mathbb{S}$ and $\xi \in \Xi$ denotes the model's set of parameters. The parameters ξ in (3.1) contain the touchdown angles of each active pair, and are

[8] Note that this differs from the way we sign the leg angles ϕ_i.

used to ground the mechanism. Then, the state propagation in the local frame is a function of one variable (ϕ), determined by the active pair.

When the right pair is active, the local coordinate frame is attached at O_1 (see Figure 3.1), $\phi = \phi_1$, $\xi = \left\{\phi_1^{\text{td}}, \phi_2^{\text{td}}\right\}$, and $(\Delta x, \Delta y, \Delta\theta) = f_{\text{R}}(\phi_1; \phi_1^{\text{td}}, \phi_2^{\text{td}})$ with

$$f_{\text{R}}(\phi_1; \phi_1^{\text{td}}, \phi_2^{\text{td}}) = \begin{bmatrix} r(\phi_1)\sin\Big(\omega(\phi_1) - |\phi_1 - \phi_1^{\text{td}} + \chi(\phi_1)|\Big) - r(\phi_1^{\text{td}})\sin\omega(\phi_1^{\text{td}}) \\ r(\phi_1)\cos\Big(\omega(\phi_1) - |\phi_1 - \phi_1^{\text{td}} + \chi(\phi_1)|\Big) - r(\phi_1^{\text{td}})\cos\omega(\phi_1^{\text{td}}) \\ \phi_1 - \phi_1^{\text{td}} + \chi(\phi_1) \end{bmatrix}. \quad (3.2)$$

In Appendix B we provide the exact expressions for the length $r(\cdot)$, and the angles $\omega(\cdot)$ and $\chi(\cdot)$, and describe the steps to derive the above state propagation equations.

Similarly, when the left pair is active, the local coordinate frame is attached at O_3, $\phi = \phi_3$, $\xi = \left\{\phi_3^{\text{td}}, \phi_4^{\text{td}}\right\}$, and $(\Delta x, \Delta y, \Delta\theta) = f_{\text{L}}(\phi_3; \phi_3^{\text{td}}, \phi_4^{\text{td}})$ with

$$f_{\text{L}}(\phi_3; \phi_3^{\text{td}}, \phi_4^{\text{td}}) = \begin{bmatrix} -\Big(r(\phi_3)\sin\Big(\omega(\phi_3) - |\phi_3 - \phi_3^{\text{td}} + \chi(\phi_3)|\Big) - r(\phi_3^{\text{td}})\sin\omega(\phi_3^{\text{td}})\Big) \\ r(\phi_3)\cos\Big(\omega(\phi_3) - |\phi_3 - \phi_3^{\text{td}} + \chi(\phi_3)|\Big) - r(\phi_3^{\text{td}})\cos\omega(\phi_3^{\text{td}}) \\ -\left(\phi_3 - \phi_3^{\text{td}} + \chi(\phi_3)\right) \end{bmatrix}. \quad (3.3)$$

Remark 1 *Note that (3.2) and (3.3) have the exact same structure, and differ only in the signs of Δx and $\Delta\theta$. This is a direct effect of the symmetry of the* SFM *about the longitudinal axis of the model. Thus, the motion of one pair mirrors that of the other, a feature that is exploited to expedite computations.*

3.2.2.2 State propagation between steps in the global frame

Figure 3.2 depicts an example of the state propagation in the global frame. Let q^{R^-} and q^{R^+} denote the state of the model expressed in the global coordinate frame before and after completing a right step, respectively. For the left pair, we define similarly q^{L^-} and q^{L^+}.

Letting the right pair go first, the model is initiated at $(x_G, y_G, \theta) = q^{\text{R}^-}$. The touchdown configuration $(\phi_1^{\text{td}}, \phi_2^{\text{td}})$ defines the position of the points O_1 and O_2, which remain fixed for the duration of the step. We propagate the model state in the local frame according to (3.2) until ϕ_1 reaches its liftoff configuration, ϕ_1^{lo}. The model state at the liftoff configuration—expressed in the local frame—is $(\Delta x, \Delta y, \Delta\theta) = f_{\mathrm{R}}(\phi_1^{\mathrm{lo}}; \phi_1^{\mathrm{td}}, \phi_2^{\mathrm{td}})$, and is then mapped back to the global frame through the 3×3 homogeneous transformation matrix $\mathrm{T}(q^{\mathrm{R}^-})$, yielding q^{R^+}. The state of the model at the end of the right step initiates the left step, with $q^{\mathrm{R}^+} = q^{\mathrm{L}^-}$. From $(\phi_3^{\mathrm{td}}, \phi_4^{\mathrm{td}})$ we find O_3 and O_4, solve (3.3) with $\phi_3 = \phi_3^{\mathrm{lo}}$ as a terminal condition, and use the homogeneous transformation $\mathrm{T}(q^{\mathrm{L}^-})$ to map the (local) model state to q^{L^+}. Then, q^{L^+} is used to initiate the next (right) pair and the cycle continues; see Figure 3.2.

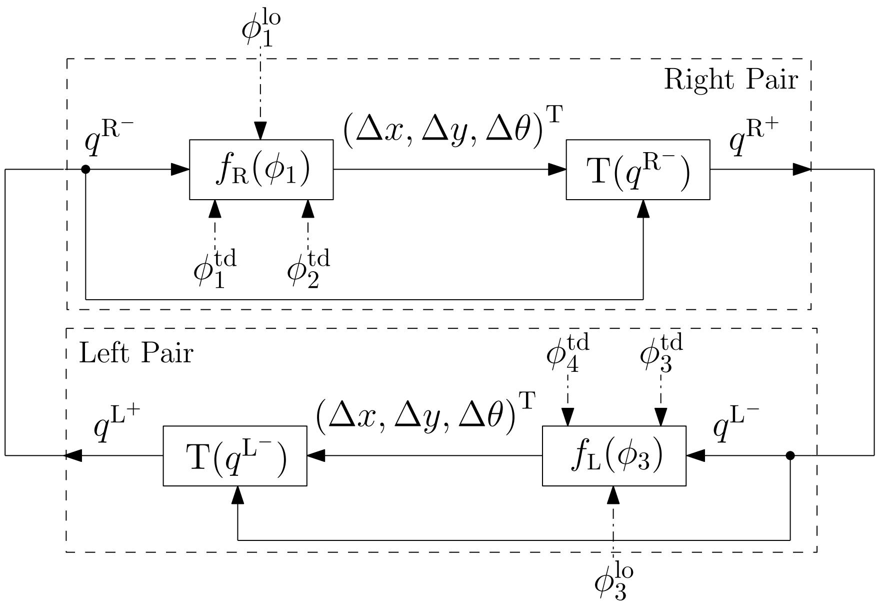


Figure 3.2: Model state propagation. State propagation in the local frame is mapped to the global frame through the homogeneous transformations $\mathrm{T}(q^{\mathrm{R}^-})$ and $\mathrm{T}(q^{\mathrm{L}^-})$ on $\mathsf{SE}(2)$. Note that the end state of a pair is used directly to initiate the next one.

The closed-form state propagation equations (3.2), (3.3) facilitate step-by-step state feedback control for miniature legged robots. One control strategy that we explore later in Section 3.3 is to select liftoff configurations to form a trajectory tracking controller based on the structure of the SFM. In the remainder of this section we present strategies for tuning the SFM parameters so that the model generates various straight-line and curved motion patterns. Subsequently, we apply these strategies to capture the behavior of a range of miniature legged robots.

3.2.3 Generating Motion Patterns with the SFM

Here we provide strategies [71] to generate straight-line and curved model trajectories with the model, and use tools from differential geometry to link curvature specifications to model parameters. This is useful since—when performing navigation tasks—it is often the case that some high-level planner prescribes curvature-specific plans, such as "go straight for 10 sec and then make a 90^o right turn." Deriving such strategies essentially links the SFM to curvature-parameterized high-level objectives.

3.2.3.1 Generating straight-line paths

In order to generate straight-line paths we select

$$\begin{aligned} \phi^{\text{td}}_{\mathsf{SLP}} &= \phi^{\text{td}}_1 = \phi^{\text{td}}_2 = \phi^{\text{td}}_3 = \phi^{\text{td}}_4 \ , \\ \phi^{\text{lo}}_{\mathsf{SLP}} &= \phi^{\text{lo}}_1 = \phi^{\text{lo}}_2 = \phi^{\text{lo}}_3 = \phi^{\text{lo}}_4 = -\phi^{\text{td}}_{\mathsf{SLP}} \ , \end{aligned} \tag{3.4}$$

where the subscript SLP is used to denote the straight-line path configuration. The configuration in (3.4) practically suggests that in order to achieve straight-line paths we need to enforce *symmetry among the sweep angles*. From (3.4) the sweep angle is

$$\psi_{\mathsf{SLP}} = \psi_1 = \psi_2 = \psi_3 = \psi_4 = |\phi^{\text{td}}_{\mathsf{SLP}} - \phi^{\text{lo}}_{\mathsf{SLP}}| = 2\phi^{\text{td}}_{\mathsf{SLP}} \ . \tag{3.5}$$

The initial, and straight-line configurations are illustrated in Figure 3.3(a) and Figure 3.3(b) respectively.

Figure 3.4 depicts the evolution of the state of the model for the case of straight-line path generation, and for parameter values in accordance with the configuration given in (3.4) and (3.5). All plots correspond to a duration of one cycle, having the same initial geometric center position and body orientation. Without loss of generality we set $d = 13$ cm, and $l = 3$ cm.[9] As we increase the value of the sweep angles (equivalently, $\phi^{\text{td}}_{\mathsf{SLP}}$), the model covers more ground in a single cycle, while the waving pattern of motion—associated with the kinematics of the four-bar mechanism—becomes more pronounced.

[9] These values in fact correspond to the length and half-width of the OctoRoACH.

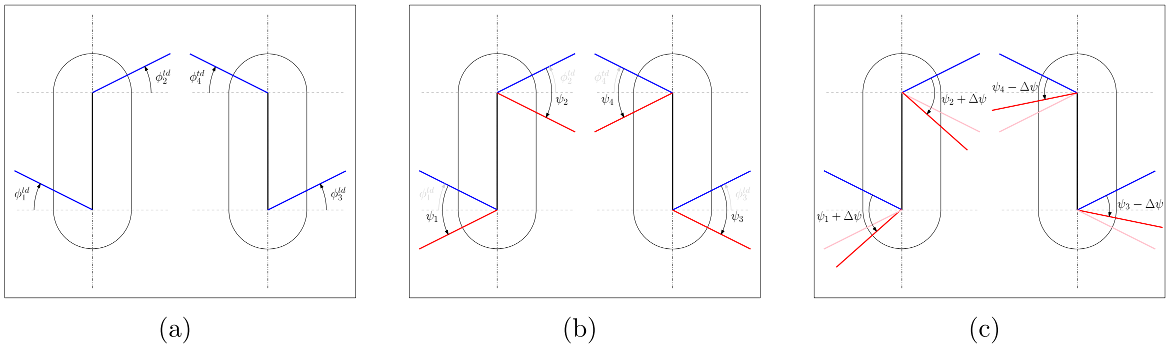


Figure 3.3: Model configurations to achieve straight-line and curved paths. (a) The initial configuration is determined by setting the touchdown angles, $\phi_i^{\text{td}}, i = 1, \ldots, 4$. (b) Straight-line paths are achieved when the leg touchdown and liftoff angles are symmetric with respect to the horizontal body axis (equivalently, all sweep angles are equal). (c) Creating asymmetry, $\Delta\psi$, in the sweep angles enables turning of the model.

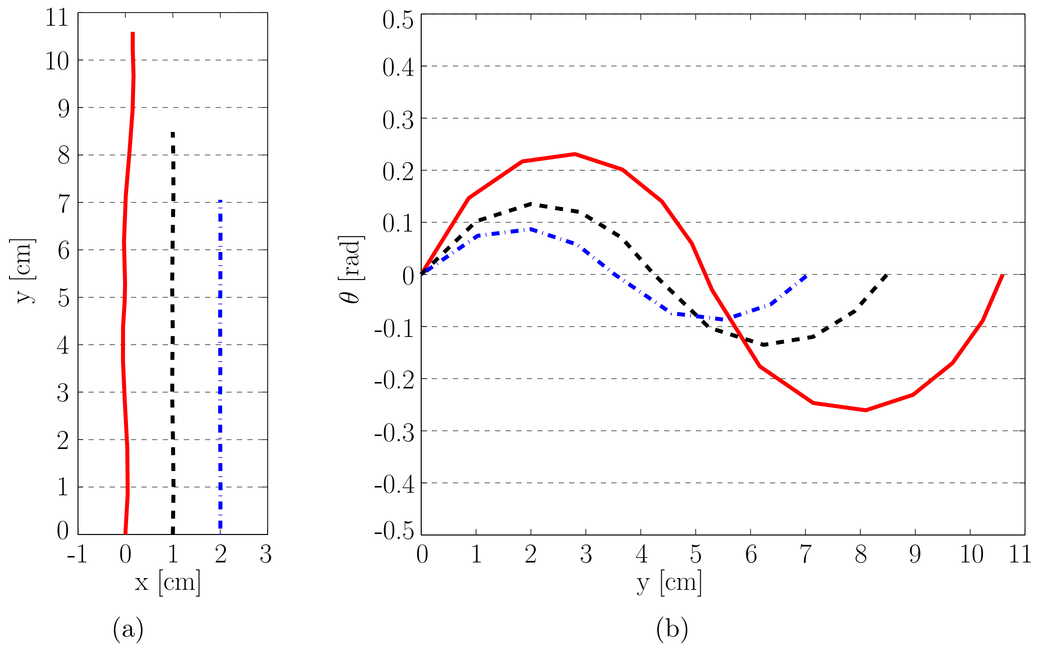


Figure 3.4: Evolution of the model's state when executing straight-line paths. Solid, dotted, and dash-dotted curves correspond to $\phi_{\mathsf{SLP}}^{td} = \frac{\pi}{3}$ rad (red), $\phi_{\mathsf{SLP}}^{td} = \frac{\pi}{4}$ rad (black), and $\phi_{\mathsf{SLP}}^{td} = \frac{\pi}{6}$ rad (blue), respectively. (a) Evolution of the position of the geometric center, G, of the model. Increasing ϕ_{SLP}^{td} accentuates the waving motion pattern, and enables the model to cover longer distances along the y-direction. (b) Evolution of the orientation of the model, θ, as it moves forward.

Figure 3.5 highlights how the model moves along straight-line paths. The pair of legs that is active—that is, in contact with the ground—is marked with solid lines, while the respective leg tips (marked with solid disks) remain fixed, forming a pin joint with the ground. The legs of the active pair rotate around these joints according to the kinematics of a four-bar linkage until the liftoff configuration is reached. At this instant (Figure 3.5(b)), the opposite pair of legs becomes active while the formerly active pair resets to its touchdown configuration. The process is then repeated (Figure 3.5(c)).

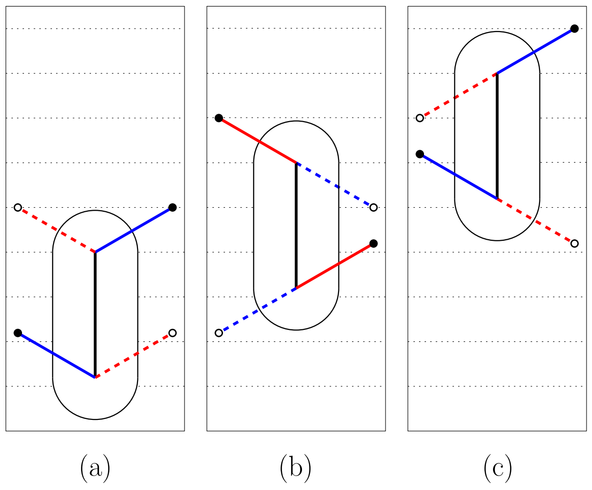


Figure 3.5: The SFM propelling itself in straight-line paths. Solid lines indicate the active pair, the tips of which are marked with solid disks. (a) The right pair begins first. Its legs rotate around their tips, which are assumed to remain fixed to the ground. (b) When the right pair reaches its liftoff configuration, the left pair turns active and the right pair is instantaneously reseted to its touchdown configuration. (c) Both pairs have completed their stride, and the model is ready to enter the next cycle.

3.2.3.2 Generating curved paths

We saw that the symmetry of sweep angles results in straight-line paths. Curved paths can be generated by asymmetric sweep angles. Indeed, as Figure 3.4(b) suggests, creating asymmetry in the sweep angles between the two pairs (Figure 3.3(c)) leads to non-zero values for the model's orientation θ at the end of each step. If the asymmetry is retained, this deviation will keep propagating, thus moving the template along a curved path. The asymmetry, $\Delta\psi$, is defined by $\Delta\psi \triangleq |\psi_R - \psi_L|$.

Figure 3.6 presents the evolution of the geometric center of the model when generating counter-clockwise curved paths for various combinations of touchdown, liftoff and sweep angles. As before, $d = 13$ cm, and $l = 3$ cm, while all paths are ten-cycle long (i.e. comprise 20 model steps), having the same initial position of the geometric center and body orientation. Notice that as the asymmetry $\Delta\psi$ increases, the produced paths have smaller radii of curvature.

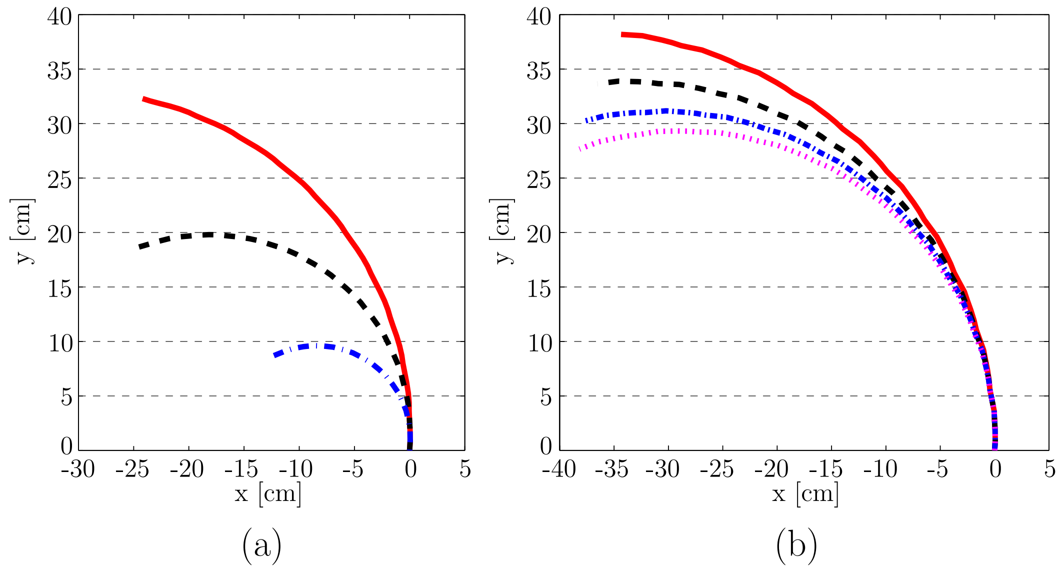


Figure 3.6: Evolution of the position of the model's geometric center, G when executing counter-clockwise curved paths. (a) Only the right pair turns active. In all three cases $\phi_1^{\mathrm{td}} = \phi_2^{\mathrm{td}} = \frac{\pi}{3}$ rad. Solid, dashed, and dash-dotted curved correspond to $\phi_1^{\mathrm{lo}} = \phi_2^{\mathrm{lo}} = -\frac{\pi}{6}$ rad (red), $\phi_1^{\mathrm{lo}} = \phi_2^{\mathrm{lo}} = 0$ rad (black), and $\phi_1^{\mathrm{lo}} = \phi_2^{\mathrm{lo}} = \frac{\pi}{6}$ rad (blue), respectively. (b) Both pairs are active. We set $\phi_1^{\mathrm{td}} = \phi_2^{\mathrm{td}} = \frac{\pi}{3}$ rad, $\phi_1^{\mathrm{lo}} = \phi_2^{\mathrm{lo}} = \phi_3^{\mathrm{lo}} = \phi_4^{\mathrm{lo}} = 0$ rad, and plot the model's response for $\phi_3^{\mathrm{td}} = \phi_4^{\mathrm{td}} = \frac{\pi}{6}$ rad (solid, red), $\phi_3^{\mathrm{td}} = \phi_4^{\mathrm{td}} = \frac{\pi}{7}$ rad (dashed, black), $\phi_3^{\mathrm{td}} = \phi_4^{\mathrm{td}} = \frac{\pi}{8}$ rad (dash-dotted, blue), and $\phi_3^{\mathrm{td}} = \phi_4^{\mathrm{td}} = \frac{\pi}{9}$ rad (dotted, magenta).

The model allows for asymmetry to be generated in multiple ways, and as a result, paths of similar curvature can be achieved through different combinations of touchdown, liftoff, and sweep angles. For example, consider the limiting case of right pair actuation only, depicted in Figure 3.6(a). In this special case, the generated paths correspond to sharper turns and have shorter length relative to their counterparts shown in Figure 3.6(b), in which right and left pair actuation alternate. To resolve redundancy we need a systematic way for matching prescribed path curvatures to model parameter values.

3.2.3.3 From model parameters to path curvatures

To support path and motion planning, we first need to characterize the geometric characteristics of paths produced from the model. Specifically, we are interested in relating path curvatures to model parameters such as the touchdown and liftoff angles. This is done by drawing tools from differential geometry of curves and surfaces [40, 146].

From a general point of view, the curvature of curves on a surface can be linked to surface (Gaussian) curvatures by applying the *Gauss-Bonnet theorem* [40, Section 4-5]. For our planar problem, the Gaussian curvature is zero, and the Gauss-Bonnet theorem reduces to

$$\sum_{j=0}^{N_s} \int_{s_j}^{s_{j+1}} k(s)\mathrm{d}s + \sum_{j=0}^{N_s} \alpha_j = 2\pi \ , \tag{3.6}$$

where N_s is the total number of model steps, s_j is the arc length of the j-th step, and $k(s)$ is the curvature of the curve associated with each step, given by

$$k(s) = \frac{x'y'' - x''y'}{((x')^2 + (y')^2)^{\frac{3}{2}}} \ .$$

The quantity α_j denotes the instantaneous change in orientation when switching from step j to step $j+1$ (Figure 3.7). The integral term in (3.6) provides the contribution of the curvature of the path followed within a step, while the remaining term accounts for the instantaneous change in orientation between steps. Then, selecting an appropriate number of steps allows the SFM to transcribe a closed circular curve, for which

$$2\pi R = \int_c \mathrm{d}s \ . \tag{3.7}$$

Combining (3.6) and (3.7) yields

$$R = \frac{\int_c \mathrm{d}s}{\sum_{j=0}^{N_s} \int_{s_j}^{s_{j+1}} k(s)\mathrm{d}s + \sum_{j=0}^{N_s} \alpha_j} \ . \tag{3.8}$$

The path curvature produced by specific values for the touchdown and liftoff angles is given by taking the inverse of (3.8), that is, $k_{\mathsf{path}} = 1/R$. In principle, (3.8) allows one to translate macroscopic requirements regarding desired path curvatures into model parameters realizing them.

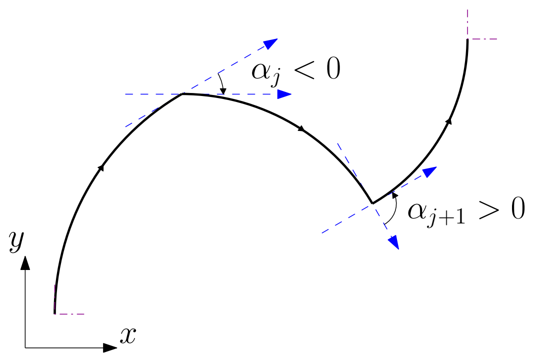


Figure 3.7: Instantaneous change in orientation when switching between steps. By convention, the sign of the angle is given by the right hand rule.

3.2.3.4 Reachability properties

Figure 3.8 graphically presents the set of reachable states from an initial state $q_0 \in \mathcal{C} \subset \mathbb{R}^2 \times \mathbb{S}$ for a total execution of 10 model steps. Without loss of generality, we pick $q_0 = (0, 0, 0)$ in units of [cm, cm, deg], and set the parameters d and l to 13 cm and 3 cm respectively. All four legs are initiated with the same touchdown angle (i.e. $\phi_1^{\text{td}} = \phi_2^{\text{td}} = \phi_3^{\text{td}} = \phi_4^{\text{td}}$). The reachable set gives us insight into the controllability properties of the model. From the graph produced in Figure 3.8, it follows that this system is *accessible* [28]. In fact, we can achieve *small-time local accessibility* if we restrict the problem to $\mathbb{R}^2$, and treat the orientation θ as a parameter.

3.2.4 Application to Miniature Legged Robots

The aforementioned strategies are helpful in analysis, but in order to ensure compatibility with the physical hardware, they must be linked to experimental data. The purpose of this section is to provide the mechanisms through which the SFM can be

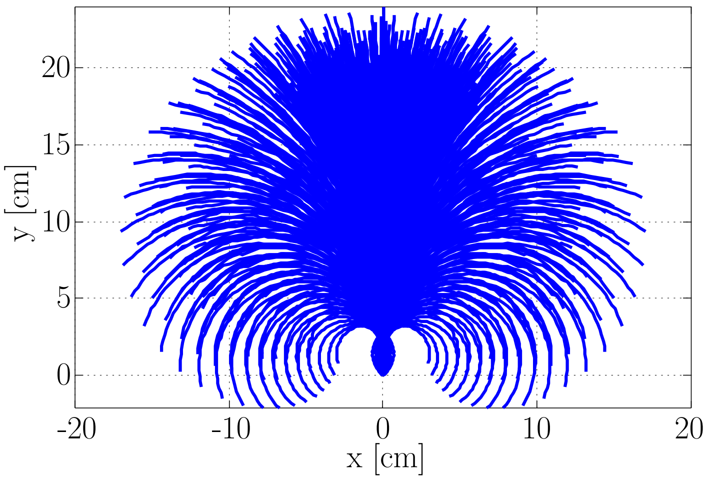


Figure 3.8: The set of reachable states produced by the SFM, starting at the initial state $x_0 = (0, 0)$ cm, $\theta_0 = 0^{\circ}$, spanning 10 model steps. Not all possible combinations of touchdown and liftoff angles are shown; this leads to some small areas not being covered by the paths.

used to capture salient motion behaviors of a variety of miniature legged robots. Application to multiple robots supports the claim that the Switching Four-bar Mechanism can indeed serve as a *template* for miniature legged robots operating quasi-statically.

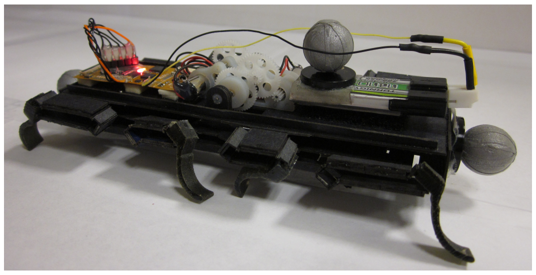
3.2.4.1 Brief description of experimental platforms

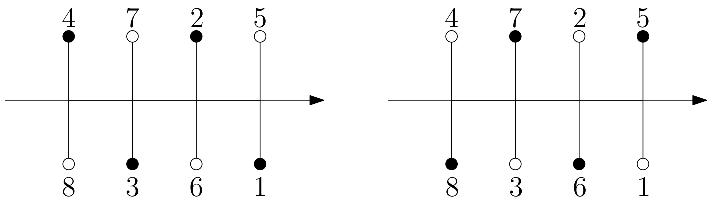
We focus on three robots: OctoRoACH [120], a revamped OctoRoACH version designed in-house, and SPIDAR. The OctoRoACH robots are manufactured through the Smart Composite Microstructure (SCM) fabrication technique [156], while SPIDAR is a hybrid design that incorporates 3D-printed and laser-cut parts. Despite their distinct morphological characteristics, all three robots employ a differential-drive steering method. Essentially, a single DC motor drives all legs at one side, without any explicit coupling between the two sides. The control inputs are the two motor gains, K_L and K_R, that control the leg velocities of the left and the right side, respectively.

The OctoRoACH

The OctoRoACH (Figure 3.9(b)) consists of a rectangular body, two actuators, and eight legs organized so that all four legs at one side are driven by a single actuator.

The robot features an ImageProc 2.2 board [120, Figure 3] for communication and motor control[10] (Atmel AT86RF231 Zigbee transceiver, ADXL345 3-axis accelerometer, ITG-3200 3-axis gyro, and two motor drivers) while a 3.7 V, 300 mAh lithium polymer (LiPo) battery powers the assembly. The leg drive kinematics [120, Figure 2] combines a slider-crank linkage responsible for leg abduction and adduction, and a parallel four-bar mechanism responsible for leg protraction and retraction.[11] Due to its mechanical coupling, the robot follows a gait whose foot-fall pattern consists of two alternating tetrapods (Figure 3.9(b)). Such metachronal gaits have been studied [19] in the context of an eight-legged arthropod (the Ghost Crab), and are in direct analogy with the tripod gaits commonly employed by a variety of six-legged animals and robots [58].


(a)



(b)

Figure 3.9: (a) The OctoRoACH [120] designed and manufactured at the University of California, Berkeley. Its body size is 130x60x30 mm, it weights 35 g, and it can reach a maximum speed of 0.5 m/s.(b) The foot-fall pattern of the robot, which results to an alternating tetrapod gait.

This alternating tetrapod gait is overlaid to the top view of the robot in Figure 3.10(a). Legs $\{1, 2, 3, 4\}$ form the "right" tetrapod, and legs $\{5, 6, 7, 8\}$ form the

[10] This particular electronics suite does not allow the robot to move backwards.

[11] For definitions of these motions, see Appendix A.

"left" tetrapod. For each tetrapod, ipsilateral legs touch the ground at the same instant and rotate in phase with the same angular velocity, depicted in the eight-legged simplified kinematic model of Figure 3.10(b) [71]. Based on this coupling, we can then combine ipsilateral legs of each tetrapod into a single "virtual" leg. This reduction maps the model in Figure 3.10(b) to the SFM template shown in Figure 3.10(c), where contralateral virtual legs (i.e., $\{O_1, O_2\}$) represent the collective effect of the tetrapod they replace (i.e., the right tetrapod $\{1, 2, 3, 4\}$) [71].

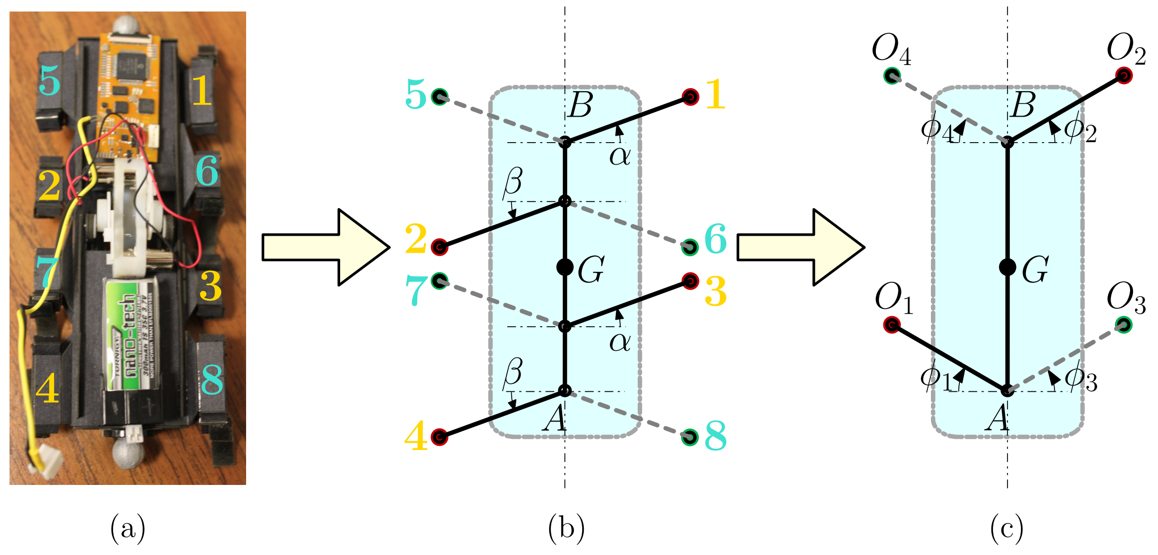


Figure 3.10: Relating the SFM template to the OctoRoACH. (a) The alternating tetrapod gait superimposed on the robot. (b) An eight-legged kinematic simplification of the gait mechanism used by the robot [71]. (c) The SFM template is formed by grouping coupled legs (indicated by the angles α and β and shown here for the right tetrapod) within a tetrapod into a single virtual leg that induces the same displacement. That is, legs $\{1, 2, 3, 4\}$ reduce to the pair $\{O_1, O_2\}$, while legs $\{5, 6, 7, 8\}$ reduce to the pair $\{O_3, O_4\}$.

Revamped OctoRoACH

Following extensive hardware experimentation, we noticed that certain parts of the OctoRoACH were failing in a consistent manner. Specifically, the housing of the robot tends to sag fast, while the two middle bars inside the housing—that enable leg abduction-adduction and protraction-retraction (see [120, Figure 2])—bend. Both effects significantly alter the kinematics of the robot. Additionally, the laminate film used for the joints ("flexures") rips easily, while the tips of the legs (made of soft molded rubber) degrade fast, altering the legs' friction coefficient and thus affecting robot performance through increased slipping.

All these motivated us to perform some mild modifications to the original design, which gave rise to the revamped OctoRoACH shown in Figure 3.11 (left). The updated design features a support nerve inside the housing (not shown) that holds together the upper and lower parts of the robot and prevents sagging. Bending of the middle bars is now prevented by using two additional support bars that form a "bridge" between the two ends of each middle bar (vertical flaps on the robot). We used Kapton® Tape to increase the strength of the joints, while all legs are now 3D-printed.[12] As a result, the revamped OctoRoACH has improved durability, longer life-cycle, and increased payload capacity without as fast performance degradation.

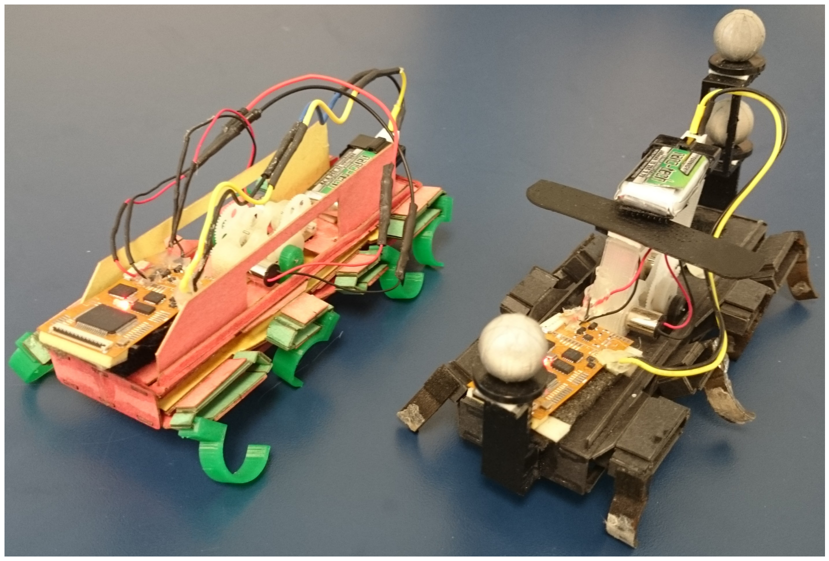


Figure 3.11: The revamped OctoRoACH compared to the original design. Under minimal modifications (described in text), the revamped OctoRoACH has improved durability, longer life-cycle, and increased payload capacity.

[12] We use polylactic acid (PLA) filament on a MakerBot Replicator 2 3D-printer.

SPIDAR

The third robot we consider is SPIDAR, shown in Figure 3.12(a). The purpose is to show that the SFM is applicable regardless of the morphological characteristics of the robot. SPIDAR is a low-cost, heavy-duty six-legged miniature robot that uses a rimless-wheel leg design similar to those employed by Mini-Whegs [106, 87] and STAR [161] robots. SPIDAR is made mostly of laser-cut body parts (Acrylic and Delrin® on a 30 W VersaLaser 350), together with 3D-printed legs and leg supports (PLA on a MakerBot Replicator 2). Contrary to the OctoRoACH family (e.g. [16, 59, 120, 52, 51]), all SPIDAR parts are bolted together; doing so effectively decreases the assembly time, facilitates repairs, and diminishes potential inconsistencies when gluing parts together (as in the OctoRoACH family). A rigid case houses the electronics; currently, these include an Arduino Fio (8 MHz ATmega328P) microcontroller with integrated XBee socket for communications, a dual TB6612FNG motor driver, a 6 V step-up voltage regulator, and two 3.7 V, 500 mAh LiPo batteries—one powers the motors, and the other the electronics. The electronics suite is protected by the case, yet access ports have been incorporated to facilitate programming of the Arduino board.

Similarly to OctoRoACH, SPIDAR uses two DC motors, each controlling all legs of one side. Extending the design of STAR [161], our design incorporates two servo motors for independent sprawl control. This way, SPIDAR can combine the benefits of wheeled and legged locomotion by appropriately adjusting its sprawl posture.[13] The legs of the robot are mechanically coupled so that the front and rear ones at each side are bound to touch and lift off the ground at the same time instant (Figure 3.12(b)). This way, we can combine the coupled legs into a single virtual leg that induces the same displacement as the pair is replaces (cf. Figure 3.10), which in turn leads to the Switching Four-bar Mechanism template.

[13] It has been found [161] that robots in this category behave like wheeled vehicles when operating in sprawled configurations; on the other hand, more upright postures can be used to overcome obstacles at the expense of a less smooth behavior.

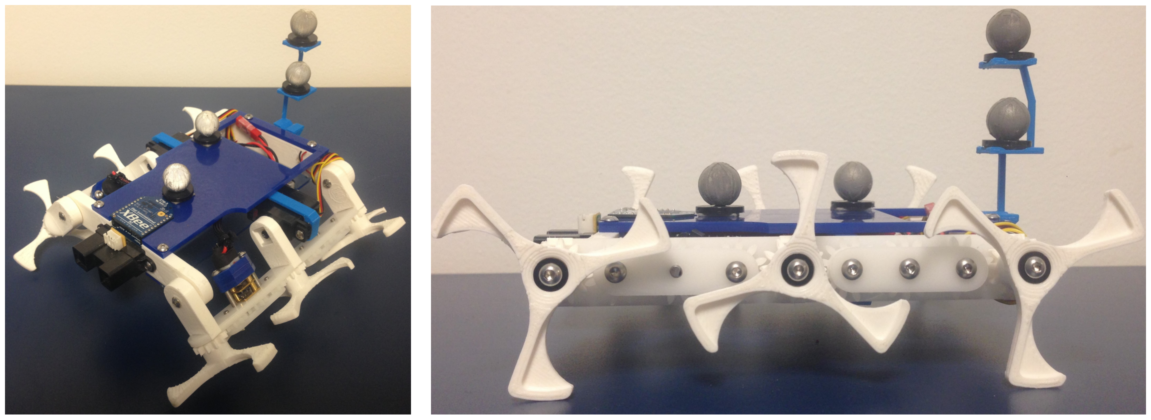

(a) (b)

Figure 3.12: (a) The SPIDAR designed and manufactured in-house, at the University of Delaware. It measures 140x150x60 mm (including the offset due to the legs when fully-extended), and weights 350 g. (b) Side view of the robot where the mechanical coupling of the legs can be easily observed. The same holds for the other side (not shown), and as a result, the robot is designed to follow an alternating tripod gait. This gait is mapped to the SFM by grouping coupled legs into virtual ones, as in the case of OctoRoACH; see Figure 3.10.

3.2.4.2 Technical approach

This section evaluates the capacity of the SFM for capturing the behavior of miniature legged robots in the horizontal plane, as observed through planar position and orientation measurements.[14] As noted in Chapter 2, the approach focuses on developing *motion primitives* since the latter considerably facilitate motion planning and control.

Experimental implementation of motion primitives

All three robots are controlled via the two motor gains K_L and K_R. Due to the differential-drive steering method that the robots employ, changing the motor gains

[14] Selecting these quantities is justified by the focus on robot navigation in low-speed, quasi-static operation regimes. Yet, one may elect to capture in a model other quantities (such as leg velocities or forces) that may be useful in high-speed locomotion, or may offer explanation regarding animal locomotion [63]; in such cases, other dynamic models may be more appropriate than the SFM.

results in either straight-line or curved paths. Specifically, three families of motion primitives are considered: (i) *straight-line paths* (SL) when $K_L = K_R$, (ii) *clockwise turns* (CW) when $K_L > K_R$, and (iii) *counter-clockwise turns* (CCW) when $K_L < K_R$. Open-loop state measurements are collected through a VICON motion capture system (8 cameras for an approximately $5 \times 5 \times 2$ m^3 working volume) at a rate of 30 Hz. The measured states express the position of the geometric center of a robot $(x_G, y_G) \in \mathbb{R}^2$, and its orientation $\theta \in \mathbb{S}$ (see Figure 3.1(a)). At the beginning of each individual experimental trial, the robots are placed into a designated start area with an initial state set at $(x_G,\ y_G,\ \theta) = (0,\ 0,\ 0)$ [cm, cm, deg]. All trials are conducted on a rubber floor mat surface and have a duration of 3 sec.

Remark 2 *The selected time duration of* 3 *sec offers a tradeoff between robot path dispersion, and path length which in turn affects the computationally complexity of the problem. Figure 3.13 provides some insight on this choice. The histograms show how individual experimental paths of the OctoRoACH disperse at different time instances. After the* 3 *sec period the path dispersion is high, and as a result, constructing motion primitives that last longer may not be meaningful in such cases since the variability in the experimental data becomes unacceptably high. On the other hand, shorter execution times may increase significantly the computational complexity when concatenating primitives, which becomes a challenge in the context of motion planning.*

Following this procedure, $I = 100$ experimental trials are observed for each motion primitive. Figure 3.14 presents the collected paths for each case: top plots correspond to OctoRoACH, middle plots to the revamped OctoRoACH, and bottom plots to SPIDAR. Continuous thick curves mark experimental averages and are overlaid on the body of the collected paths shown with thin curves. Table 3.1 contains the motor gains that realize the primitives at hand, and the observed final states on average.

Comparing the behavior of the original OctoRoACH with the one designed in-house, it is observed that the latter demonstrates less variability in its motion. This result was anticipated since the robot structure is more rigid than the original design,

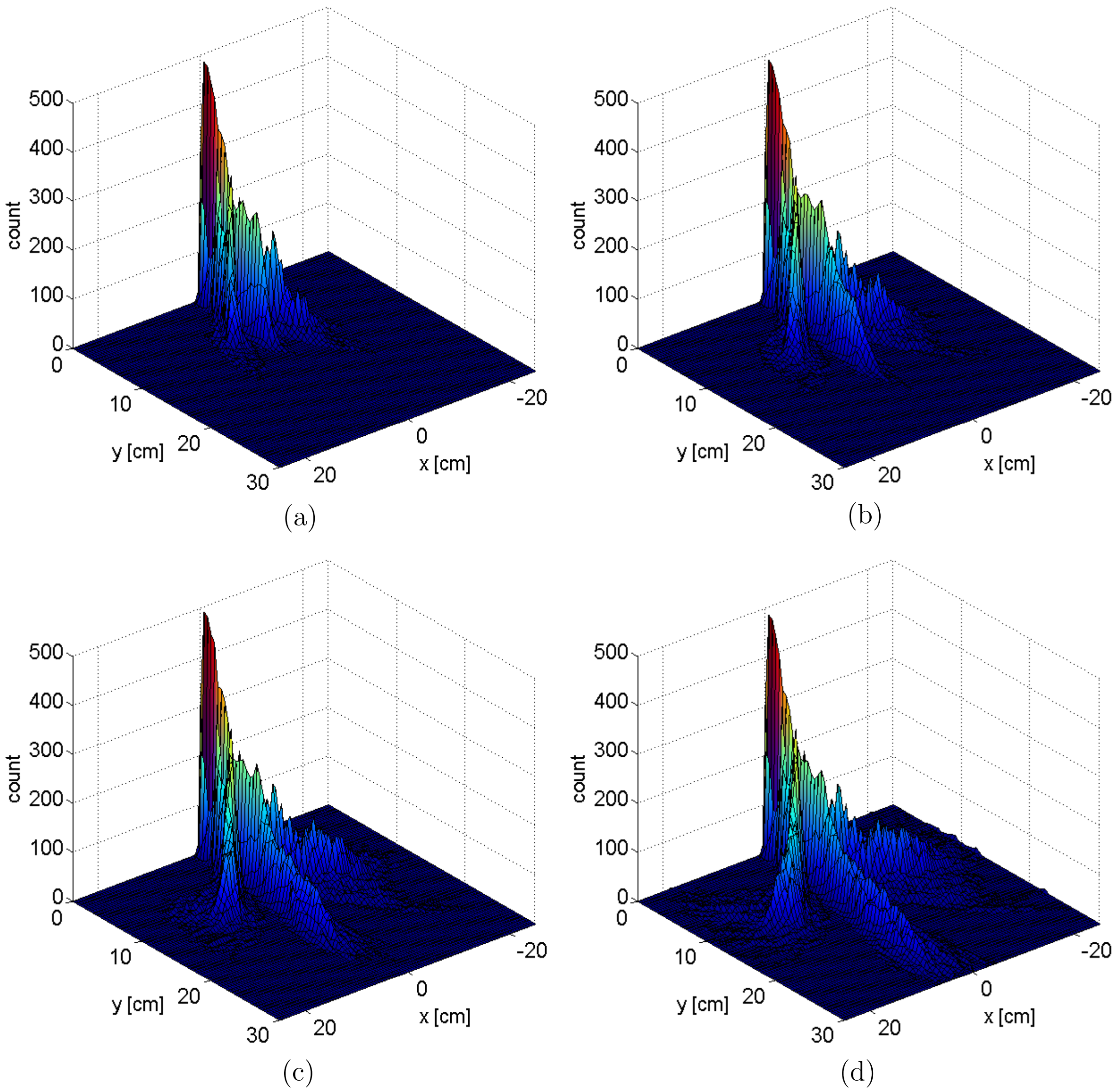


Figure 3.13: Path dispersion at (a) 2 sec, (b) 2.5 sec, (c) 3 sec, (d) 3.5 sec for the OctoRoach. All primitives start at the origin, and are largely dispersed at the end of the 3 sec trial as shown in (c). The z axis counts the number of paths that are inside a particular grid square. Due to the selected grid size, some paths may appear more than once inside a square.

Table 3.1: Motion Primitives

Platform	Primitive Type	Motor Gains (K_L, K_R)	Final State (average values) $\big(x_f$ [cm], y_f [cm], θ_f [deg]$\big)$
OctoRoACH	SL	$(40, 40)$	$(2.30, 18.19, -7.29)$
	CW	$(60, 20)$	$(7.88, 9.67, -38.40)$
	CCW	$(20, 60)$	$(-10.53, 11.38, 42.35)$
Revamped OctoRoACH	SL	$(40, 40)$	$(-0.72, 17.02, 2.27)$
	CW	$(60, 20)$	$(4.50, 9.71, -44.05)$
	CCW	$(20, 60)$	$(-4.58, 8.71, 42.96)$
SPIDAR	SL	$(100, 100)$	$(-0.24, 26.77, 4.69)$
	CW	$(150, 50)$	$(9.75, 20.33, -54.94)$
	CCW	$(50, 150)$	$(-10.64, 20.64, 61.97)$

therefore the uncertain effects of body compliance in robot motion are reduced. Moreover, the friction between the 3D-printed legs and the rubber floor mat is higher, thus the revamped OctoRoACH tends to cover less distance at the same time compared to the original design. Shorter path lengths are mainly observed when executing curved paths; this can be associated with the anisotropic surface of the feet and the mat. SPIDAR behaves very smoothly, and path variability remains low despite the fact that this robot moves significantly faster than the OctoRoACH robots (see Figure 3.14).

Linking to model parameters

The next step is to identify those parameter values that enable the SFM template to produce paths that capture the experimental averages. Each experimental trial has the form of a time series of length T; in the case considered here, $T = 90$ since a trial lasts 3 sec and is observed at a rate of 30 Hz. Then, a motion primitive is defined as the timed average $w^{\text{ave}}(t),\ t \in 1, \ldots, T$, of the individual paths it encapsulates (e.g.,

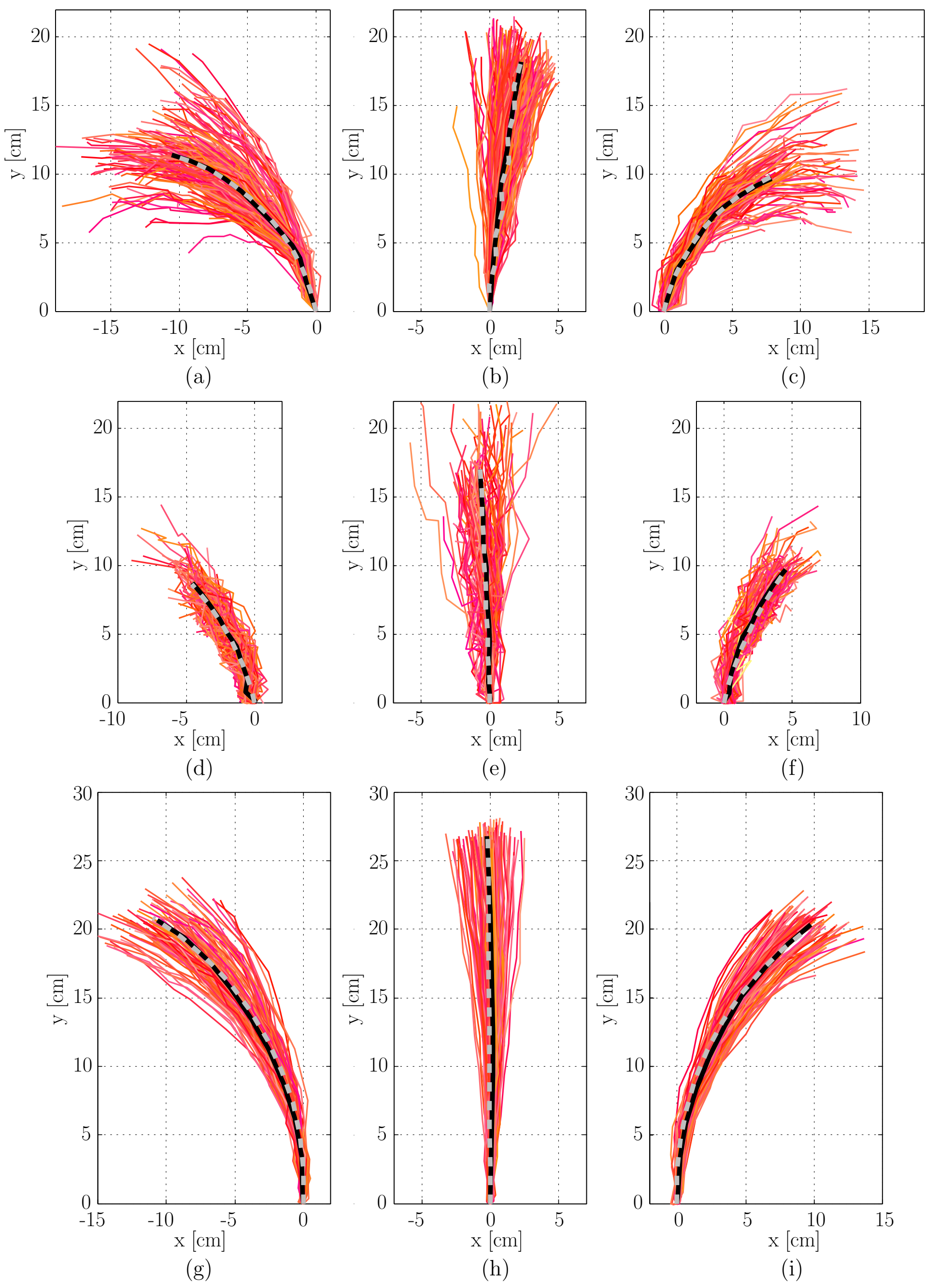


Figure 3.14: Experimental data for the motion primitives considered here. Individual paths are shown with thin curves, while experimental averages are marked with thick curves. The SFM outputs that capture best these experimental averages are shown with lightly-shaded dashed curves. (a)-(c) CCW, SL, and CW primitives for the OctoRoACH. Similarly (d)-(f) for the revamped OctoRoACH, and (g)-(i) SPIDAR.

straight-line paths). The shorthand notation $\mathcal{M}(\xi)$ is used to describe a model that is parameterized by the parameter vector $\xi \in \Xi$, while the term $\mathsf{out}(\mathcal{M}(\xi))_t$ denotes a model-generated path.[15] Note that a model-generated path is essentially a time series that contains the evolution of the state of the model, $q(t) = (x_G(t), y_G(t), \theta(t)) \in \mathbb{R}^2 \times \mathbb{S}$; the subscript t highlights this fact. Then, the (nominal) parameter values $\bar{\xi} \in \Xi$ that result in model paths that capture the experimental averages in a least-squares sense are identified by solving the optimization problem:

$$\bar{\xi} = \arg\min_{\xi \in \Xi} \sum_{t=1}^{T} \|\mathsf{out}(\mathcal{M}(\xi))_t - w^{\text{ave}}(t)\|^2 \ . \tag{3.9}$$

Certain conventions are made before solving (3.9). First, the quantities d and l are chosen so to match the robots' actual length and half-width, respectively. Thus, $d = 13$ cm, $l = 3$ cm for both OctoRoACH robots, while for SPIDAR $d = 14$ cm and $l = 7.5$ cm. An additional case ($d = 13$ cm, $l = 3$ cm) is also considered for SPIDAR. The latter is used as a first means to test the robustness of the model when (some) parameter values may vary. The number of model steps for each primitive is set to $N_\pi = 10$. This number of model steps has been chosen empirically over the course of data collection and model analysis to provide adequate resolution for the touchdown and liftoff configurations to capture 3 sec-long experimental data. A different approach would be to include the leg angular velocities $\dot{\phi}_i$ as additional parameters in the optimization problem (3.9).[16] It is reminded here that straight-line motion is generated by activating both left and right pairs of legs, with the same touchdown and liftoff configurations, that is $\phi_1^{\text{td}} = \phi_2^{\text{td}} = \phi_3^{\text{td}} = \phi_4^{\text{td}} = \bar{\phi}^{\text{td}}$ and $\phi_1^{\text{lo}} = \phi_2^{\text{lo}} = \phi_3^{\text{lo}} = \phi_4^{\text{lo}} = \bar{\phi}^{\text{lo}}$. Clockwise turns are generated as a variation of the above configuration, where only the left pair is active i.e., $\phi_1^{\text{td}} = \phi_2^{\text{td}} = \phi_1^{\text{lo}} = \phi_2^{\text{lo}} = 0$ throughout the stride. Similarly, counter-clockwise turns are produced by activating only the right pair: $\phi_3^{\text{td}} = \phi_4^{\text{td}} = \phi_3^{\text{lo}} = \phi_4^{\text{lo}} = 0$. Experimental data from the OctoRoACH robots

[15] Although not crucial here, this notation will be used thoroughly later in Chapter 4.

[16] The effect of this different parameterization is in fact evaluated later in Section 4.3.

demonstrate a very rapid change of orientation at the beginning of their paths (see Figure 3.14(a)-(f)). This is captured by including the initial orientation of the model, θ^{init}, as an additional parameter that needs to be identified based on data.

With these conventions in place, the model parameters to be identified form the vector

$$\xi = \left[\bar{\phi}^{\text{td}} \;\; \bar{\phi}^{\text{lo}} \;\; \theta^{\text{init}}\right] \in \Xi \ , \tag{3.10}$$

and the selection is made by solving (3.9) for each of the cases shown in Figure 3.14. Table 3.2 contains the nominal model parameters, as well as the model-predicted final state for each robot at the end of each primitive. Null entries in the third column of the table indicate that the initial orientation was not included in the parameter identification problem (3.9). Nominal SFM paths are shown in Figure 3.14 as lightly-shaded thick dashed curves. Both cases for SPIDAR are very close, so the output of the first case ($d = 14$ cm and $l = 7.5$ cm) is shown only.

Table 3.3 provides an error measure for the quality of fit (ϵ_x [cm], ϵ_y [cm], ϵ_θ [deg]) in the form

$$\frac{1}{T}\sum_{t=1}^{T} \left|\mathsf{out}(\mathcal{M}(\xi))_t - w^{\text{ave}}(t)\right| \ . \tag{3.11}$$

Position errors are small in all cases, independently of the platform. The orientation error, however, seems to depend on the platform. The OctoRoACH robots demonstrate more variability in their motion, which is observed in experiments through rapid changes in orientation during a primitive. These discrepancies are caught by the error measure in (3.11). As a result, the errors in Table 3.2 are in agreement with the visual observations made based on Figure 3.14 suggesting that the original OctoRoACH design exhibits more variability than the revamped OctoRoACH, while SPIDAR has more smooth and less uncertain behavior.

Overall, the model is found to be capable of capturing experimental data well in all cases. This is achieved by utilizing primarily two physically-relevant parameters: the touchdown and the liftoff angles. The model's initial orientation is added as a

Table 3.2: Identified Nominal SFM Parameter Values

Platform	Primitive Type	Model Parameters $\{\bar{\phi}^{\text{td}}, \bar{\phi}^{\text{lo}}, \theta^{\text{init}}\}$ [deg]	Model-Predicted Final State $\big(x_G\,[\text{cm}], y_G\,[\text{cm}], \theta\,[\text{deg}]\big)$
OctoRoACH	SL	$\{65.57, 27.31, 0.00\}$	$(2.31, 18.21, 0.00)$
	CW	$\{38.78, 15.70, -15.00\}$	$(7.94, 9.85, -68.58)$
	CCW	$\{40.40, 11.65, 15.00\}$	$(-10.44, 11.37, 76.60)$
Revamped OctoRoACH	SL	$\{0.06, -39.81, 0.00\}$	$(-0.70, 17.04, 0.00)$
	CW	$\{23.96, 4.93, -15.00\}$	$(4.53, 9.57, -37.48)$
	CCW	$\{28.60, 11.30, 15.00\}$	$(-4.62, 8.64, 44.33)$
SPIDAR	SL	$\{8.39, -12.97, 0.00\}$	$(-0.17, 26.83, 0.00)$
$d = 14\,\text{cm}$	CW	$\{38.95, -3.87, 0.00\}$	$(9.61, 20.42, -55.71)$
$l = 7.5\,\text{cm}$	CCW	$\{26.91, 12.95, 0.00\}$	$(-10.96, 20.83, 61.92)$
SPIDAR	SL	$\{25.06, -28.65, 0.00\}$	$(-0.14, 26.80, 0.00)$
$d = 13\,\text{cm}$	CW	$\{25.98, 12.93, 0.00\}$	$(9.53, 19.99, -56.89)$
$l = 3\,\text{cm}$	CCW	$\{39.88, -4.28, 0.00\}$	$(-10.23, 20.76, 58.01)$

parameter only in the case of the turning primitives for the OctoRoACH robots. Moreover, the SFM is able to capture the behavior of SPIDAR equally well in both cases; this provides some preliminary evidence that the SFM is robust as a model when some parameters are perturbed. Taken together with the fact that the reported data come from *morphologically distinct robots* that operate quasi-statically at *different speeds*, the aforementioned results experimentally affirm that the Switching Four-bar Mechanism can serve as a *template* for miniature legged robots in quasi-static operation regimes. The remainder of this chapter focuses on motion planning, navigation, and control for miniature legged robots using the SFM template.

Table 3.3: Errors in Fit

Platform	Primitive Type	Error in Fit $(\epsilon_x \text{ [cm]}, \epsilon_y \text{ [cm]}, \epsilon_\theta \text{ [deg]})$
OctoRoACH	SL	$(0.14, 0.57, 7.38)$
	CW	$(0.11, 0.21, 17.20)$
	CCW	$(0.35, 0.37, 19.55)$
Revamped OctoRoACH	SL	$(0.08, 0.64, 2.42)$
	CW	$(0.26, 0.47, 7.26)$
	CCW	$(0.21, 0.51, 7.91)$
SPIDAR	SL	$(0.15, 0.92, 1.87)$
$d = 14$ cm	CW	$(0.15, 0.66, 1.92)$
$l = 7.5$ cm	CCW	$(0.15, 0.75, 1.26)$
SPIDAR	SL	$(0.15, 0.91, 1.89)$
$d = 13$ cm	CW	$(0.14, 0.69, 1.23)$
$l = 3$ cm	CCW	$(0.12, 0.59, 1.96)$

3.3 Template-Based Motion Planning, Navigation, Control for Miniature Legged Robots Using the SFM

The purpose of this section is to show that the SFM is also useful in the context of motion planning, navigation, and control for miniature legged robots. The system identification procedure described before plays an important role since it essentially links the control inputs (that is, the motor gains) of the physical robot to SFM template parameter values. This information can then be used to link high-level planning policies to control strategies that can be directly implemented to the physical hardware.

The approach is applied to OctoRoACH and SPIDAR, employs the motion primitives constructed in the previous section for each robot, and consists of three stages. These stages are (i) trajectory planning (mid level), (ii) trajectory tracking control (low level), and (ii) trajectory re-planning and control. The third stage essentially consolidates the mid and low levels, and is needed when the uncertainty that affects the motion of a robot is too large to be handled solely by a low-level controller. As we will show shortly, this situation has been observed in the case study of OctoRoach navigation. The task is to reach a desired state (position and orientation) at a given static environment while avoiding obstacles. The desired state is assumed to be given a-priori by some high-level planner. This assumption is only made for illustration purposes, and does not limit the generality of the approach; later in Section 3.4 we extend the results to include visibility-based constraints as an example of a more complex high-level specification.

3.3.1 Trajectory Planning

In the first stage, the time-parameterized motion primitives are used in a temporal sequence to generate a collision-free reference trajectory between an initial and the desired state selected by some high-level planner. The construction of suitable sequences of primitives is achieved by employing a *Rapidly-exploring Random Tree* (RRT) solver [88, Sections 5.5 and 14] and [28, Section 7.2.2].[17] Extensions leading to optimal plans [67] are in principle applicable, but we choose the original RRT solver mainly due to its popularity, proven efficacy in experiments, and availability of direct software implementations.

The RRT solver requires a map of the environment (i.e. the configuration space), the initial and desired states q_0 and q_d, respectively, as well as the available motion primitives SL, CW, and CCW for each robot. Since obstacles are present, the configuration space is expressed as $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$, where $\mathcal{C}_{free}$ is used to denote the obstacle-free portion of the configuration space, while $\mathcal{C}_{obs}$ denotes the obstacle region. In the core of the approach, the solver uses the primitives to generate new vertices in

[17] The RRT is deemed sufficient as we consider a single initial-goal pair configuration. Cases with multiple initial-goal pairs are tackled by employing *probabilistic roadmaps* [75] (PRMs).

a graph. The end of each vertex is used as a node from which new vertices branch out—at first there is only the initial node. Vertices that cross obstacle boundaries are automatically discarded. Using this approach, the *reachability tree* can be constructed by a breadth-first implementation. To reduce, however, the computational cost of a breadth-first approach, the RRT solver implements a *sampling-based* approach to select a node for spanning new vertices. Specifically, a point in the obstacle-free configuration space is sampled,[18] and then the node that is closer—according to some distance metric e.g., the Euclidean metric—to the sampled point, is used to span new vertices. The process continues until q_d has been reached. Note that due to motion constraints and the discrete nature of robot motion primitives, finding a sequence of primitives ending exactly at q_d is very unlikely; therefore, we consider the target reached when the trajectory ends within a radius of 15 cm around the target position, with final orientation in the range of $[-45^o, 45^o]$. Table 3.4 summarizes the key steps of the RRT solver we implement.

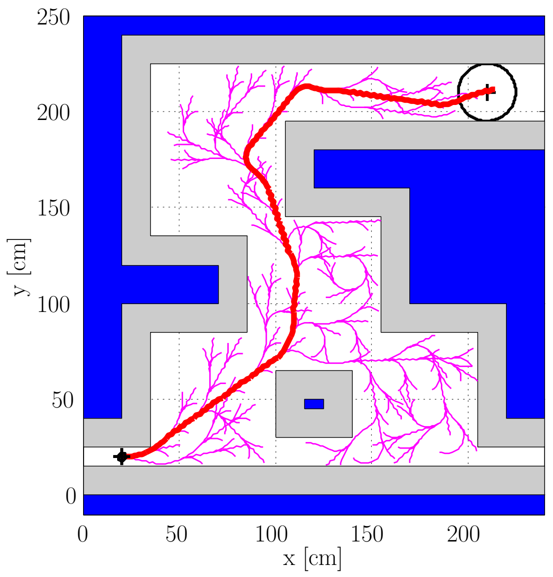
Table 3.4: Primitives-Based RRT Planner Steps

```
1.  Read Workspace, $q_0$, $q_d$;
2.  Read the SL, CW, and CCW primitives;
3.  for $i = 1$ to $k$ do
4.      Sample random point $\alpha(i)$ in free workspace;
5.      Find vertex $q_n$ closest to $\alpha(i)$;
6.      Create SL, CW, and CCW edges from $q_n$;
7.      Add collision-free edges;
8.      Update vertex list;
9.      Exit if a neighborhood of $q_d$ is reached;
```

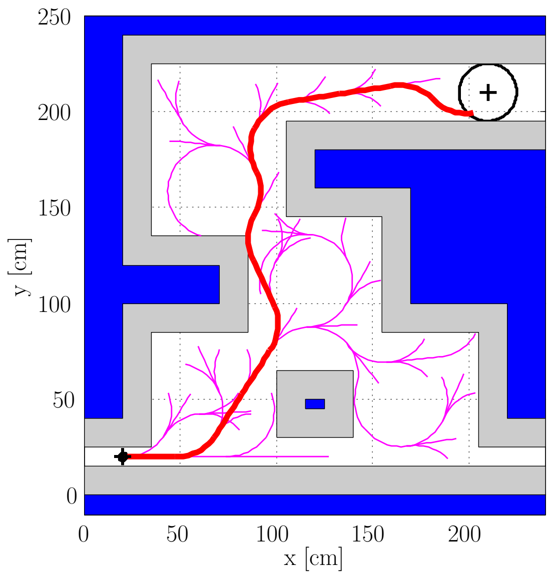
The solver is implemented in two case studies; first for OctoRoACH and then for SPIDAR. Figure 3.15 shows sample reference trajectories that steer each robot from the initial state $q_0 = (20, 20, -90)$ [cm, cm, deg], to the goal centered on $q_d =$

[18] Here, we sample points from $\mathcal{C}_{free}$ according to a uniform distribution; however, one may choose to introduce bias in the sampling to speed up the process.

(210, 210, −90) [cm, cm, deg]. The initial and desired states remain the same for both OctoRoach (Figure 3.15(a)) and spidar (Figure 3.15(b)). Obstacles are artificially augmented (lightly shaded regions) to account for platform volume. Any vertices that cross the boundary of augmented obstacle regions are also discarded. The thin curves in Figure 3.15 mark the branches of the constructed trees, while the thick curves highlight the constructed reference trajectories for each robot.[19] The reference trajectories shown in Figure 3.15 are used below in both simulations and experiments.



(a)



(b)

Figure 3.15: The RRT is combined with the motion primitives generated by the SFM template for the (a) OctoRoach and (b) spidar (see Table 3.2). The solver generates desired trajectories (thick curves) in environments populated with obstacles. We discard any edges and vertices that cross the boundaries of the augmented obstacle regions (lightly shaded areas) surrounding the actual obstacles (in blue).

Figure 3.16 shows the physical environment that is considered in both the OctoRoach and spidar case studies. Experiments are ran on a rubber floor mat and environment boundaries are created using foam material. The robot is placed manually

[19] In Figure 3.15(b) the reference trajectory appears to touch the augmented obstacle region, but this is an artifact of marking that trajectory as a bold curve.

in a designated START position, and the goal is for the platform to reach a rectangular region marked GOAL in Figure 3.16. Ground truth information on the geometric center and the orientation of the platform is recorded using a VICON motion capture system. Control computations are performed in Python running on a host Linux laptop, while reference trajectory generation is done using MATLAB implementations of RRT on the same machine. These software modules interface with the physical platform through ROS. Control commands (desired motor gains) are sent to the robot at a rate of 3.33 Hz—recall that a primitive lasts 3 seconds and is made up of 10 steps.

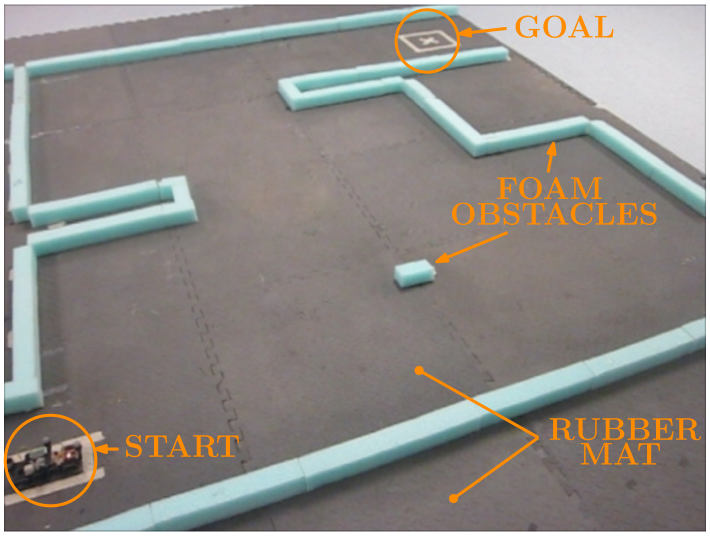


Figure 3.16: The physical environment realizing the case studies considered here. The robot starts at the bottom left corner and is required to navigate to the top right corner of the environment, while avoiding collisions.

3.3.2 Trajectory Tracking Control

The next stage focuses on the development of a low-level trajectory tracking controller based on the structure of the SFM template. Figures 3.17 and 3.18 justify the need for developing a trajectory tracking controller. Specifically, the figures depict simulation and experimental results of executing the sequence of the primitives chosen by the RRT directly on the system—the OctoRoACH in this case—in an open-loop fashion. In the simulation results, shown in Figure 3.17, the model is corrupted with noise on a step-by-step basis. In detail, given an initial state and desired values for the touchdown and liftoff angles,[20] the model state propagates deterministically according to (3.2) or (3.3), depending on whether the step is taken by the right or the left pair, respectively. Once a step is concluded, the state of the model is perturbed by additive zero-mean multivariate Gaussian noise with covariance $\Sigma = \mathrm{diag}\{\sigma_x^2, \sigma_y^2, \sigma_\theta^2\}$. The process of injecting uncertainty is shown in the flow diagram of Figure 3.19. Overall, we notice that even for low magnitude of infused uncertainty ($\sigma_x = \sigma_y = 0.25$, $\sigma_\theta = 2^o$ as in Figure 3.17(a)), the probability of following the desired trajectory is substantially small.[21] The above result is also observed in open-loop experiments with the OctoRoach, as shown in Figure 3.18. The significant levels of noise and motion disturbances, clearly impact performance and render successful execution unlikely, necessitating the closure of a low-level control loop to perform trajectory tracking.

The analytic expressions derived in Section 3.2.2 greatly facilitate the design of a closed-loop controller, the purpose of which is to minimize the tracking error which is expected by the simulations, and also observed in experiments. To achieve this, at the beginning of a step j—right or left—the controller computes the template-predicted state at the beginning of the next step $j+1$ and compares it with the desired one for the model parameters of Table 3.2, selecting those parameter values that minimize the predicted error at step $j+1$. The selected parameter values are implemented for the current step and the process repeats. It is emphasized that the controller is implemented on a step-by-step basis; therefore, we require resolution in the steps that compose the primitives which comprise reference trajectories.

Let N_s be the template steps in each primitive, and let N_π be the number of primitives that compose the reference trajectory. A reference trajectory thus consists

[20] We remind here that each primitive consists of 10 model steps by design, while desired parameter values are shown in Table 3.2 and are selected based on the type of primitives that compose a reference trajectory.

[21] The specific values for σ_x, σ_y, and σ_θ considered in Figure 3.17 are chosen at this point arbitrarily; however, in Chapter 4 we provide a systematic means for infusing into a model the uncertainty that is actually observed in experiments.

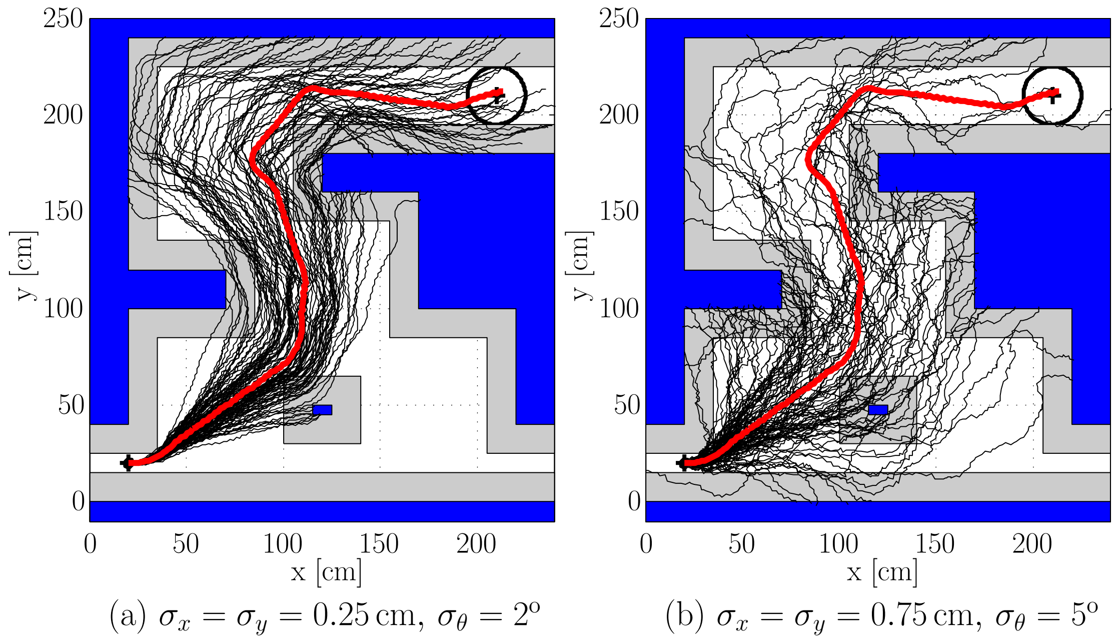


Figure 3.17: Simulated response of the system commanded to follow the desired trajectory shown earlier in Figure 3.15(a) (thick red curve), when uncertainty perturbs its state at the end of every step. In all cases we simulate 100 trials. (a) Open-loop response of the system with low magnitude of infused uncertainty scores a 10% success rate. (b) As the magnitude of the infused uncertainty grows, the system looses track of the desired trajectory, and the success rate reduces to 2%.

of $N_s \cdot N_\pi$ steps; the index $j \in [1, N_s]$ denotes a particular step of the trajectory. For $n \in \{1, \ldots, N_\pi\}$, define the mapping

$$\pi_n : [N_s{\cdot}(n{-}1){+}1, N_s{\cdot}n] \to \mathbb{R}^2 \times \mathbb{S} \ .$$

In essence, π_n is a sequence of desired states expressed in the global coordinate frame indexed by $j \in [N_s{\cdot}(n{-}1){+}1, N_s{\cdot}n]$. The reference trajectory Π is then defined as the concatenation

$$\Pi \triangleq \pi_1\pi_2\ldots\pi_{N_\pi} : [1, N_s] \to \mathbb{R}^2 \times \mathbb{S} \ .$$

Using the above notation, $\Pi[j] = (x'[j], y'[j], \theta'[j])$ denotes the desired state of the system at the beginning of step j. Let $q[j] = (x_G[j], y_G[j], \theta[j])$ be the actual state

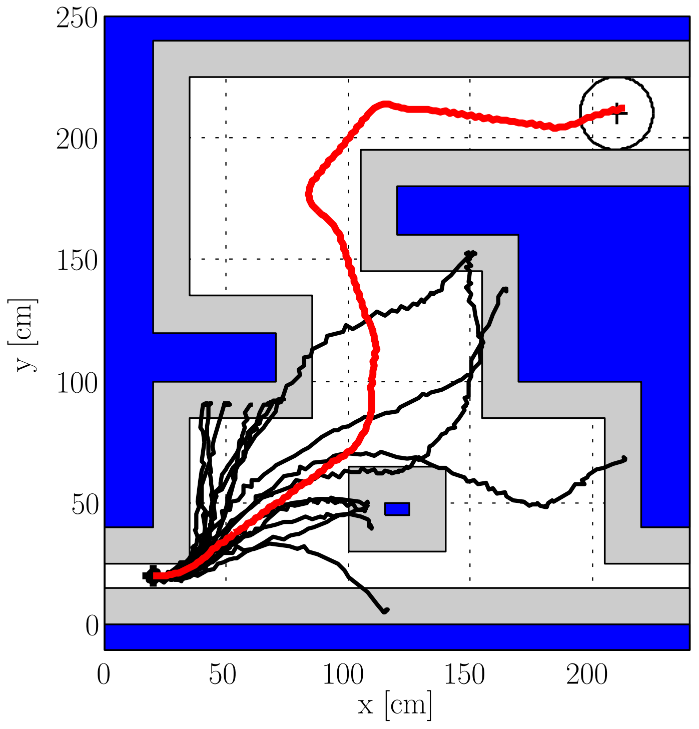


Figure 3.18: Experimental results with the OctoRoACH commanded to follow the desired trajectory generated by the RRT solver in open-loop. $I = 15$ trials are considered. As expected by the simulations, open-loop execution of the desired trajectory is unsatisfactory. The robot collides with its environment soon after it starts navigating.

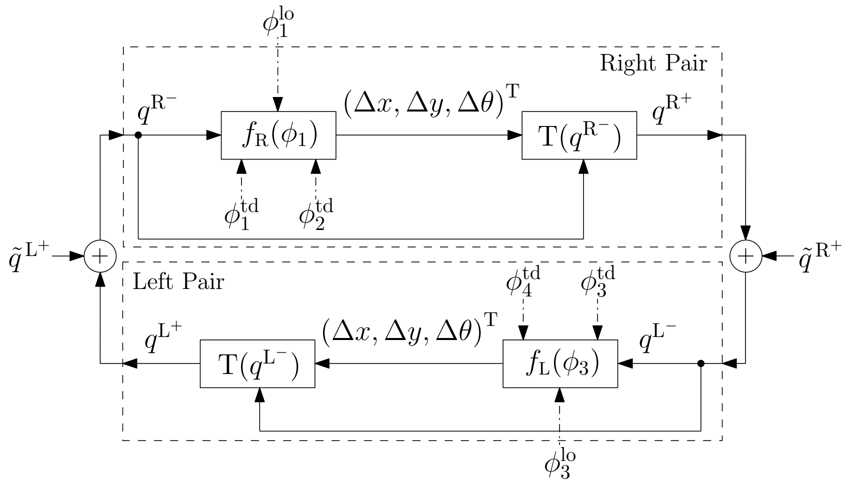


Figure 3.19: Model state propagation with injected step-by-step uncertainty. The uncertainty is added at a step's end state expressed in the global frame, and is $\tilde{q}^{\,\text{R}^+}$ for a right step or $\tilde{q}^{\,\text{L}^+}$ for a left step.

at step j. For a parameter pair $\left(\bar{\phi}^{\text{td}}, \bar{\phi}^{\text{lo}}\right)$ drawn from Table 3.2, the template-predicted state of the system at the beginning of the next step is

$$(x_G[j{+}1], y_G[j{+}1], \theta[j{+}1]) = \mathrm{T}\left(q[j]\right)(\Delta x, \Delta y, \Delta\theta) \tag{3.12}$$

where $\mathrm{T}\left(q[j]\right)$ is the homogeneous transformation on $\mathsf{SE}(2)$ that maps the progression of the model in the local frame back to the global (see Figure 3.2). $\Delta x, \Delta y$, and $\Delta\theta$ are determined by $\left(\bar{\phi}^{\text{td}}, \bar{\phi}^{\text{lo}}\right)$ and are given by the expressions in (3.2) or (3.3) depending on the active pair (i.e., right or left, respectively).

With the template-predicted state $(x_G[j{+}1], y_G[j{+}1], \theta[j{+}1])$, the predicted error at the beginning of the next step is

$$\begin{aligned}\delta x &= x_G[j+1] - x'[j+1]\ \ ,\\ \delta y &= y_G[j+1] - y'[j+1]\ \ ,\\ \delta\theta &= \ \ \theta[j+1] - \theta'[j+1]\ \ ,\end{aligned}$$

where $(x'[j{+}1], y'[j{+}1], \theta'[j{+}1])$ is the desired state at the beginning of step as given by the reference trajectory. Through the computation of $(x_G[j{+}1], y_G[j{+}1], \theta[j{+}1])$, the predicted error depends on the parameters $\left(\bar{\phi}^{\text{td}}, \bar{\phi}^{\text{lo}}\right)$. The objective of the controller is to pick $\left(\bar{\phi}^{\text{td}}, \bar{\phi}^{\text{lo}}\right)$ from Table 3.2 so that

$$\rho(q, q') = \sqrt{\left[\rho_\theta(\theta, \theta')\right]^2 + \alpha_{cl}\left[\rho_{\mathbf{x}}(\mathbf{x}, \mathbf{x}')\right]^2} \tag{3.13}$$

is minimized. In (3.13), $\rho_\theta(\theta, \theta')$ is the $\mathsf{SO}(2)$ metric using complex notation (see [88, Section 5.1.2]), and $\rho_{\mathbf{x}}(\mathbf{x}, \mathbf{x}')$ is the Euclidean metric on $\mathbb{R}^2$. Essentially, $\left[\rho_\theta(\theta, \theta')\right]^2 = (\cos(\theta) - \cos(\theta'))^2 + (\sin(\theta) - \sin(\theta'))^2$, and $\left[\rho_{\mathbf{x}}(\mathbf{x}, \mathbf{x}')\right]^2 = (\delta x^2 + \delta y^2)$. The factor α_{cl} is a relative weight between orientation and planar distance discrepancies that is chosen empirically; in this setup, $\alpha_{cl} = 0.015$ offers a good tradeoff in penalizing errors in orientation and planar distance.

To assess (i) the performance improvement compared to open-loop motion, and (ii) the sensitivity of the trajectory tracking controller to (process) noise, we first run closed-loop simulation tests with OctoRoach. Similarly to the open-loop simulations shown in Figure 3.17, we add noise generated according to a zero-mean normal distribution at the end of each step. This acts essentially as a stochastic perturbation to the right hand side of (3.12). Figure 3.20 depicts both the closed-loop response of the system in simulation for the case study of Figure 3.15(a), for varying degrees of infused uncertainty. In each case we simulate the system 100 times. Figure 3.20 shows the response of the system for significantly large motion perturbations when the trajectory tracking control loop is closed. As expected, closing a low-level trajectory tracking loop drastically improves the chances of the system reaching its target without collisions.

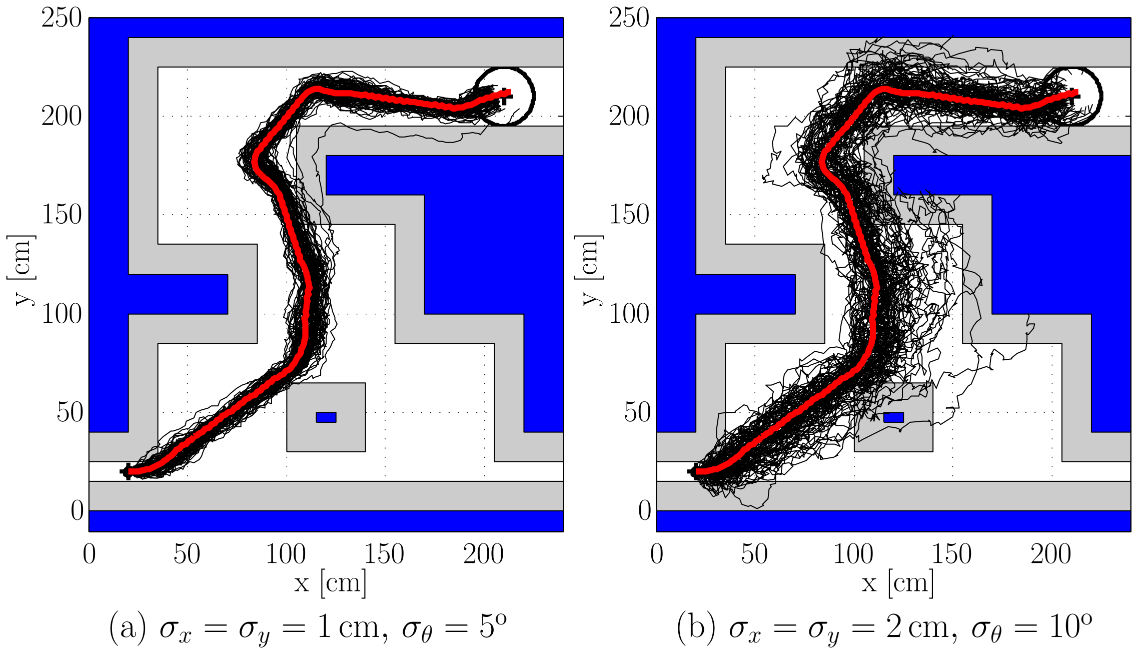


Figure 3.20: Simulated response of the system commanded to follow a desired trajectory (thick red curve), when uncertainty affects its state at the end of every step. In all cases we simulate 100 trials. (a) Closed-loop response of the system using the reported trajectory tracking controller. When the magnitude of the infused uncertainty is low, the controller enables the system to follow the desired trajectory, scoring a 100% success rate. (b) As the magnitude of the infused uncertainty grows, the system exits the region of attraction of the controller and may loose track of the desired trajectory, with the success rate reducing to 85%.

Next, we test the reported trajectory tracking controller on the physical robot. This time we use both OctoRoach, and spidar. The procedure is the same for both robots albeit the difference in the desired reference trajectories and nominal model parameters implementing the motion primitives of interest; see Figure 3.15, and Table 3.2, respectively. The controller picks a model parameter configuration from the available ones shown in Table 3.2, and this selection is mapped to motor gains through Table 3.1. We run the test 15 times for each robot, and plot the results in Figure 3.21.

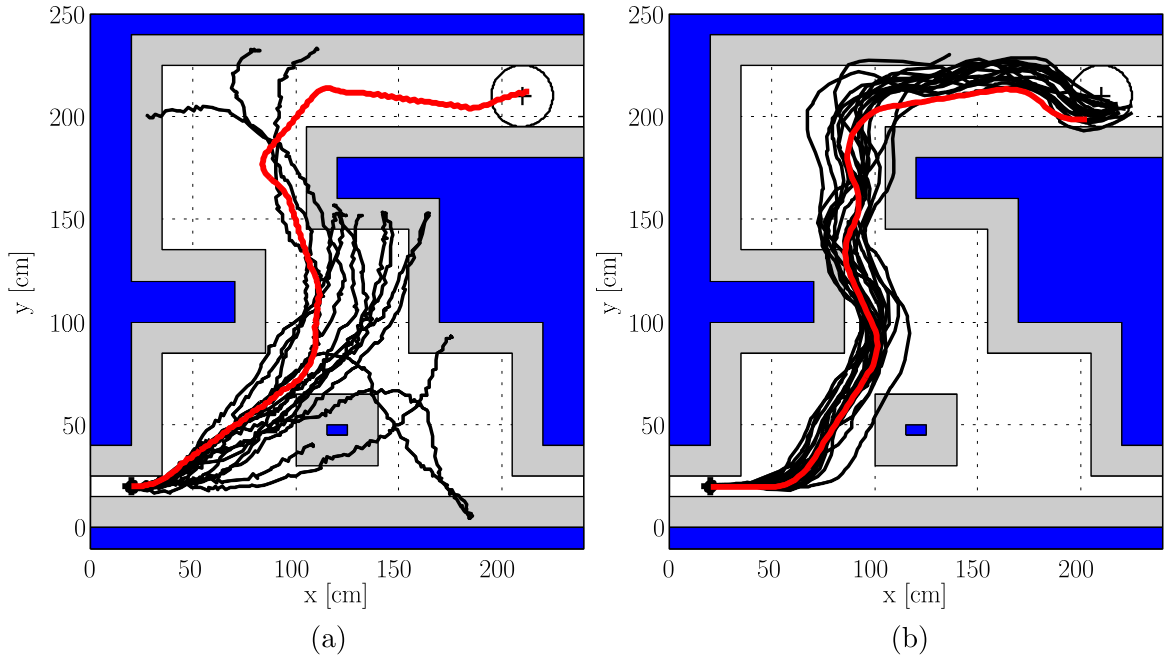


Figure 3.21: Experimental results of closed-loop navigation for the OctoRoACH and SPIDAR. The desired trajectory is marked with a thick curve (in red) in both cases. $I = 15$ paths are collected for each robot. (a) Closed-loop response of OctoRoACH using the reported trajectory tracking controller. Compared to open-loop execution (cf. Figure 3.18), the controller substantially improves the behavior of the robot and enables it to follow the desired trajectory for longer; however, the local nature of the controller inhibits successful completion of the task. (b) Closed-loop response of SPIDAR using the reported trajectory tracking controller. The response of the robot is smoother, and application of the controller yields a 93.3% probability of reaching the target.

It can be readily verified from Figure 3.21 that the reported controller is effective in the case of SPIDAR, with only one out of 15 trials failing to reach the target.[22]

[22] The performance in open-loop navigation is similar to the observation we made for OctoRoACH; the robot still collides with its surroundings rapidly.

Focusing on the performance of the OctoRoACH, compared to the open-loop case the controller substantially improves the behavior of the robot and enables it to follow the desired trajectory for longer. However, the persistence of process noise in conjunction with the length of the reference trajectory turned out to be insurmountable in all 15 experiments. This also reveals the robustness limits of the controller: beyond a certain threshold, none of the primitives of Table 3.2 can close sufficiently fast the gap between the current and desired state on the reference trajectory—which with persistent noise perturbation, can grow uncontrollably (Figure 3.21(a)). To address this limitation another loop closure is mandated. Closed at a higher level, this second control loop is the subject of the next section.

3.3.3 Trajectory Replanning and Control

The main idea in the third stage of the approach is to wrap a control loop around the reference trajectory generator that works in a receding-horizon fashion, closing at a period corresponding to $\delta_\pi < N_\pi$ primitives. Essentially, the system executes the initial trajectory for δ_π primitives, and then recomputes from the current state a new trajectory to follow for the next δ_π primitives. This cycle is repeated until either the goal is reached, or the system collides. In this implementation, we use the RRT solver to update the reference trajectory so that the approach essentially reduces to trajectory replanning. Note, however, that other formal predictive control schemes such as [73] can also in principle be applied to close this outer control loop. The outer control loop is applied only to OctoRoACH since the low-level trajectory tracking controller was successful in steering SPIDAR toward the target at a 93.3% rate.

The OctoRoACH results change drastically when the outer control loop is also closed. The trajectory is refined every 2 primitives using the RRT solver, and the process is repeated until the target is reached, or the robot collides with its environment. Using this trajectory replanning and control scheme leads to an 80% success rate, with only three out of 15 trials failing to reach the desired state. Figure 3.22 shows the results of applying this control scheme. Table 3.5 summarizes the trajectory completion rates for each case focused on the OctoRoACH; OL, CL, and RCL stand for open-loop, closed-loop and replanning-and-control loop, and correspond to Figures 3.18, 3.21(a) and 3.22, respectively.

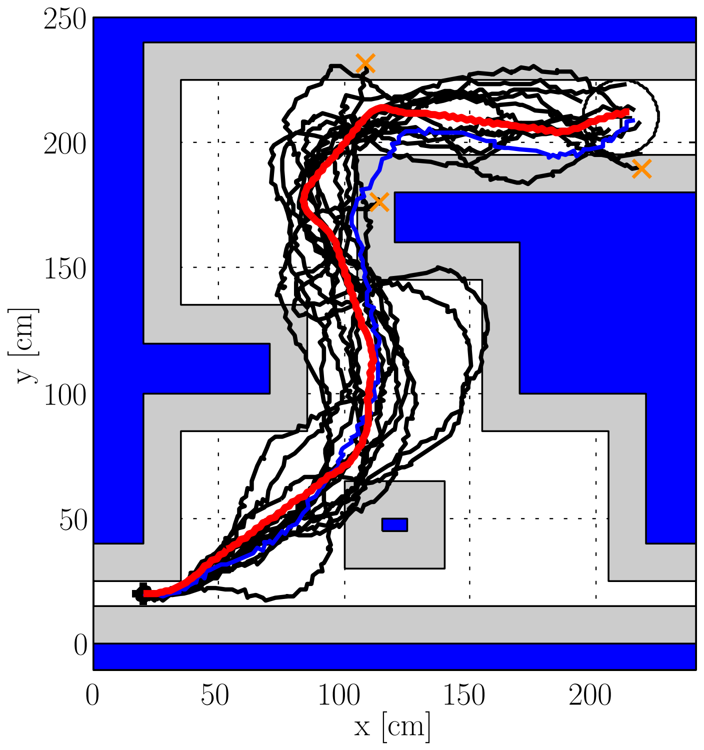


Figure 3.22: Experimental results with the OctoRoACH. As before, the desired trajectory is marked with a thick curve (in red), and we repeat the trial for 15 times. The combination of local trajectory tracking control with the prediction phase enables the robot to successfully complete its task most of the times (terminal collisions are marked with a cross).

Table 3.5: Trajectory Completion Rates for OctoRoACH

Case	OL	CL	RCL
Trajectory Length Covered (maximum)	52.4%	69.7%	100%
Trajectory Length Covered (minimum)	17.8%	22.1%	61%
Trajectory Length Covered (average)	29.9%	49.8%	95.4%

Figure 3.23 illustrates in more detail how the trajectory replanning and closed-loop control scheme worked in a specific experimental trial, highlighted in blue (color version) in Figure 3.22. The tree that is constructed from a robot state is shown in thin curves, while the thick curve marks the reference path selected by the planner. As the robot moves toward its goal, the root of the tree translates closer to the destination. Each instance in Figure 3.23 illustrates a new reference trajectory update, calculated after executing two primitives (approximately 20 robot steps) along the current reference trajectory. On its way to the goal, process noise pushes the robot in the shaded region stopping it short of collision with obstacles (Figures 3.23(g)-(h)). In these cases, the RRT solver is allowed to generate new edges inside the augmented obstacle regions provided that generated trajectories exit the shaded region after the first primitive. Note that the behavior can be modified by tuning the control horizon, or activating

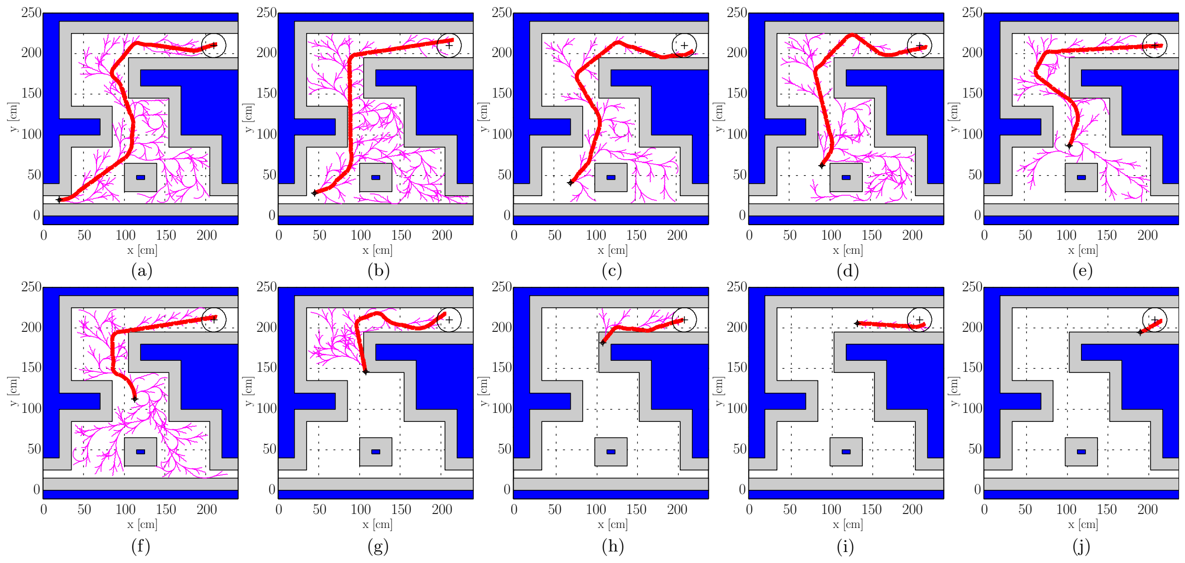


Figure 3.23: Illustrative example of how the two-stage trajectory replanning and control scheme works when applied to the physical robot. (a) The initial trajectory (thick curve) generated by the RRT solver (see Figure 3.15) is updated after the first 2 primitives of the plan have been executed, with the local trajectory tracking controller being active. (b) The updated trajectory (thick curve) is followed for the first 2 primitives, and updated again by the solver. (c)-(j) This process is repeated until the desired state has been reached.

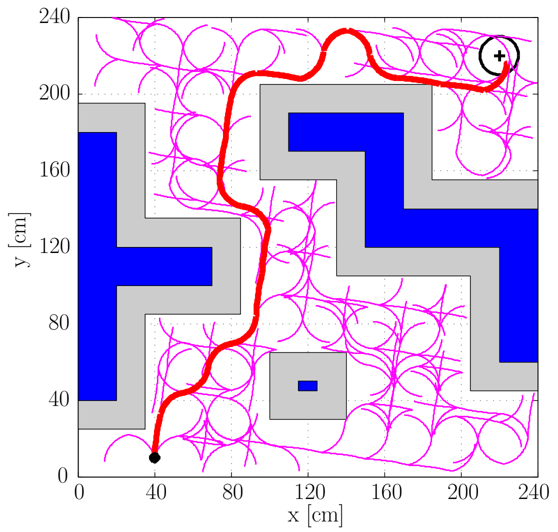
the prediction phase only when the error in the trajectory tracking stage grows above a certain threshold.

3.4 Additional Considerations

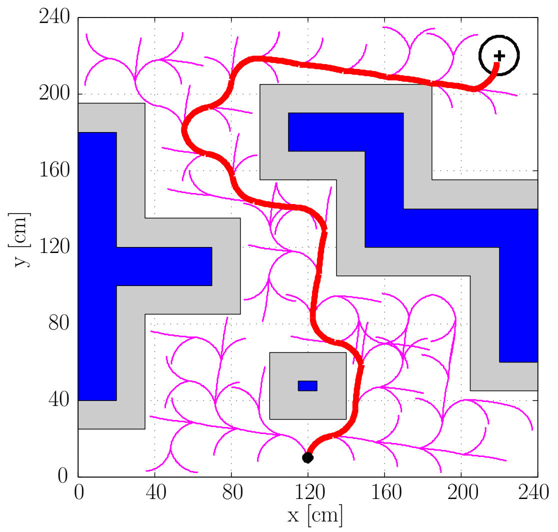
The Switching Four-Bar Mechanism (SFM) template offers promise in supporting motion planning for miniature legged robots. The reported template integrates well within an RRT solver which typically defines the benchmark in motion planning. Figure 3.15 highlighted two particular examples applied to distinct miniature legged robots. Figure 3.24 below provides further evidence by showing motion plans for the OctoRoACH when the initial state varies [69]. The solver manages to find a path using the template-based motion primitives in all cases.

The reported approach is general, and can be used to deal with navigation tasks that extend beyond reachability, to more complex, dynamic tasks as well. We show this point through a surveillance task under visibility-based and operational tempo constraints (see Figure 3.25). We envisage an OctoRoACH equipped with a 360^o field-of-view camera, that needs to monitor an environment populated with obstacles while maximizing the camera's visibility polygon. The task is served by a recursive solver as follows. From an initial node we construct the reachability tree based on the template-based primitives. One way to account for the operational tempo constraints is to require that the next node to be visited must be within some prespecified distance from the starting node. In the case we consider here, we set that distance to be between three– and six-primitives long. Then, out of all possible nodes, the solver selects the node that maximizes the visibility polygon. The visibility polygon is constructed by spanning rays from the position of the camera intersecting the vertices of the obstacles; the intersection points are marked with an asterisk in Figure 3.25 and define the vertices of the visibility polygon. The robot then moves to the selected node, and the procedure repeats. An example with six iterations is shown in Figure 3.25.

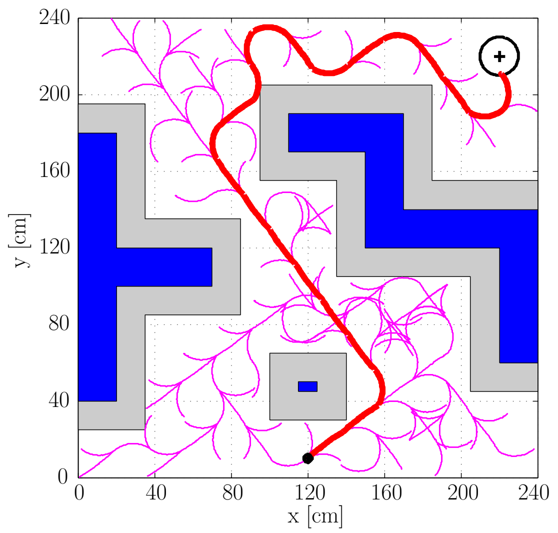
As a final remark, we address the case of a purely data-driven approach, without using any model. The experimentally-constructed motion primitives (i.e. experimental



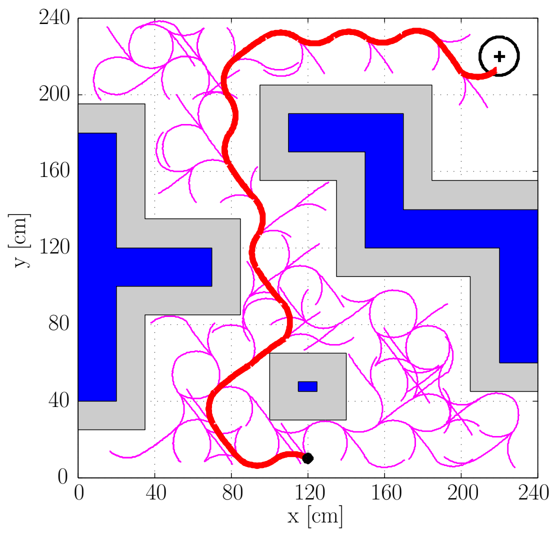
(a)



(b)



(c)



(d)

Figure 3.24: The SFM integrates within the RRT solver when the initial state varies. The first case differs in the initial position, while the last three start at the same position but with different initial orientations. Regardless the initial state, the solver was always able to find a solution in a matter of seconds. The curvy final part in cases (a), (c), and (d) is caused by setting the desired final orientation in the interval $[-30, 30]^o$; this effect can be rectified by letting the execution time vary. The tree is constructed using template-predicted OctoRoACH motion primitives.

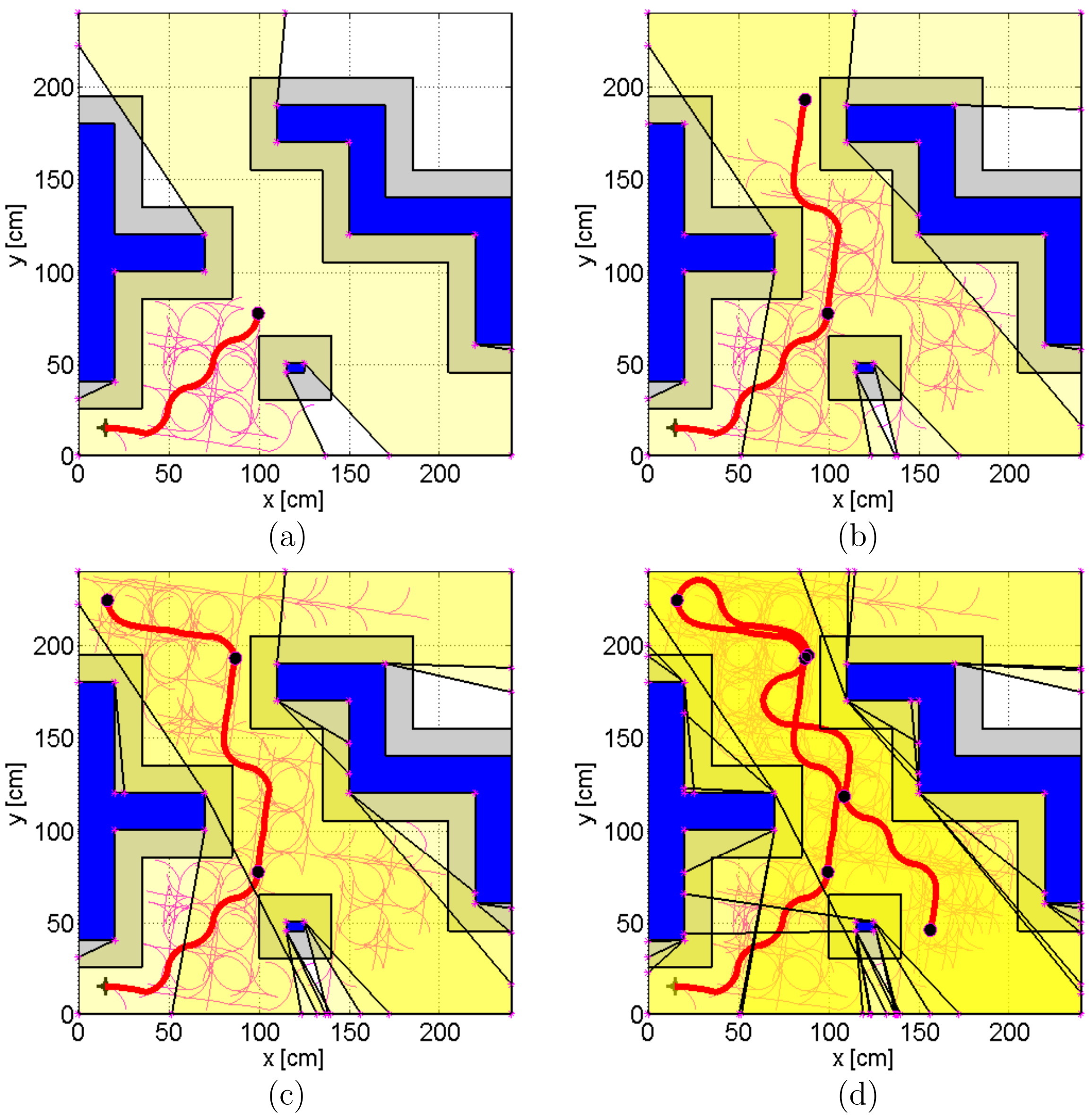
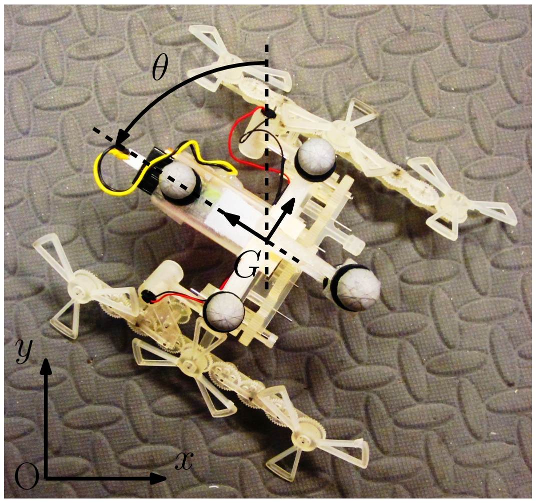


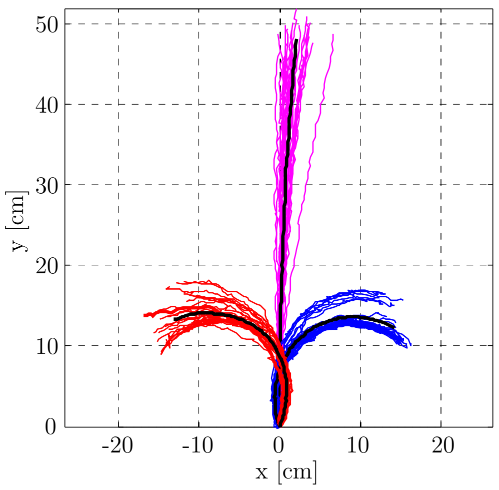
Figure 3.25: An example of a more complex task: surveillance under visibility-based and operational tempo constraints. Our approach works in a recursive fashion. (a) It first constructs a sub graph of the reachability tree (thin lines), and selects the node that maximizes the visibility polygon (highlighted region). (b)-(d) Then the system navigates to the selected node and the process repeats.

sample means) are used directly for motion planning and navigation. This emulates the case of a robot shipped with built-in motion primitives. This case study features

STAR [161], since its behavior is considerably smoother and less variable. Figure 3.26(b) contains the experimental paths gathered during the calibration phase. Thick curves correspond to the experimental average for each primitive, and are subsequently used to define motion primitives for the robot. In turn, the primitives are used by the RRT solver to generate reference trajectories in environments of increasing complexity (Figure 3.27). The generated sequences of motion primitives are then executed on the physical robot; the results are gathered in Figure 3.28.



(a)



(b)

Figure 3.26: (a) The state of the robot is $(x_G, y_G, \theta) \in \mathbb{R}^2 \times \mathbb{S}$. (b) Data collected from STAR. Thin curves depict the evolution of the geometric center of the robot, while the experimental sample means out of a total of 30 paths for each case is shown in thick curves.

With respect to Figure 3.28, 95%, 83%, and 59% of paths for cases (a), (b), and (c), respectively, were implemented in full before a terminal contact with an obstacle or the boundary occurred. Similarly, 94.3%, 93.8%, and 75.9% of the planned path length was covered for each case. Finally, 13.3% of paths succeeded in reaching the goal for case (a), 10% for case (b), and there were no successful paths for the hardest workspace case of Figure 3.28(c). Common in all cases is that uncompensated accumulated errors make actual paths deviate substantially from the planned. These errors can be reduced

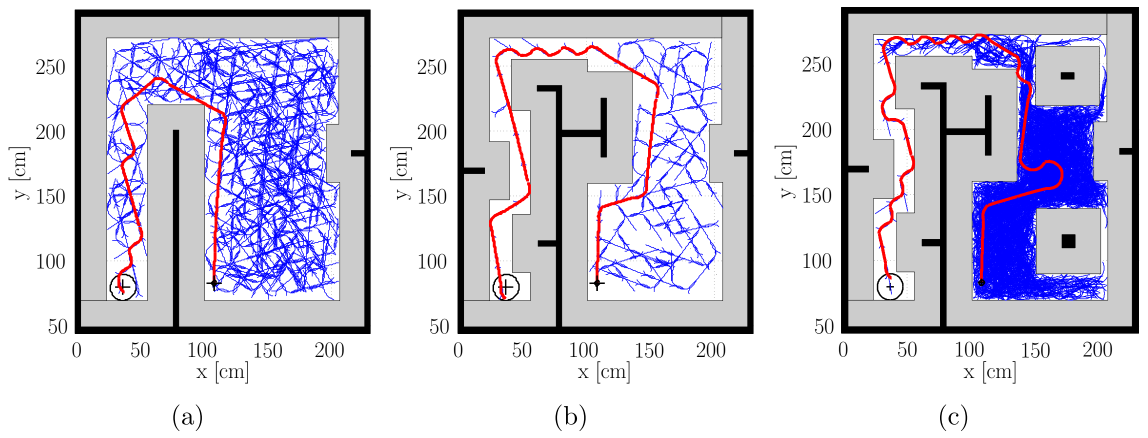
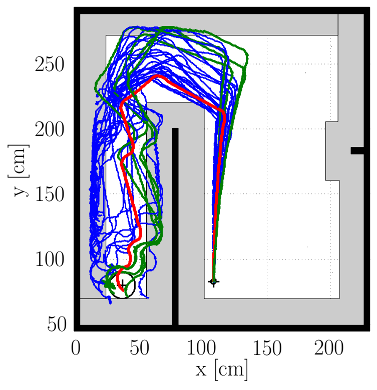


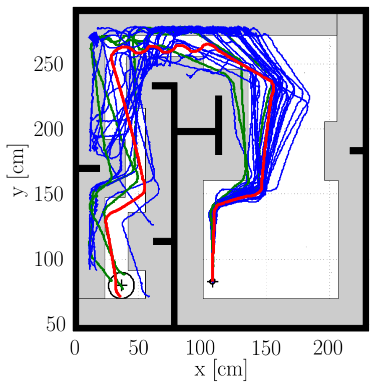
Figure 3.27: Implementation of the RRT solver for the case study of STAR. The map increases in complexity from left to right by adding more obstacles. (a) The basic map: Many solutions exist, and the resulted shortest path involves minimal switching among robot actions. (b) A set of obstacles has been added to block the initial path. The planner has to respect the motion constraints of the problem; this leads to the "wavy" motion pattern close to the top left corner. (c) The most complicated environment considered: Two areas to the right are now inaccessible.

in part by closing a control loop to ensure trajectory tracking, which typically works well when a model of the system—or a template—can be constructed. An added benefit of employing a model is the ability to make predictions about cases that have been observed during the initial calibration phase.

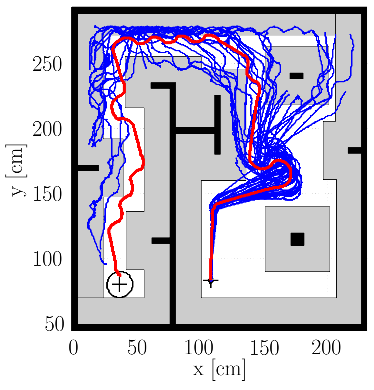
Remark 3 *A large number of paths bring the robot in contact with the workspace boundary. In most of these cases, the robot was able to keep making progress toward its goal. Moreover, there exist cases where the robot-wall interaction was beneficial. For instance, Figure 3.28(a) shows that after the impact, the wall compensated for the accumulated error, and aided STAR in moving closer to the goal. On the contrary, if no walls were present, the robot would have deviated significantly from its predetermined path. Such effects and tradeoffs can be studied with suitable stochastic models.*



(a)



(b)



(c)

Figure 3.28: Experimental implementation of the plans shown in Figure 3.27. (a) Least complex workspace: 4 paths (in green) reach the desired configuration. (b) Medium-complexity workspace: 3 trials reached the goal. (c) No successful trials were recorder for the hardest workspace.

3.5 Discussion

Overall, we emphasize that data must be used to judge the efficacy of templates, and generate suitable instances that capture salient robot motion behaviors. Our experimental results validate this thesis in several distinct case studies. Once constructed, such appropriate templates can be used to ensure that the different levels of the hierarchical framework are compatible with each other.

Unfortunately, compatibility does not always ensure consistency. Our results on miniature legged robots show that despite elaborate efforts to deal with uncertainty at a low-level, policies developed in the high level may not work well. This is primarily due to the fact that we do not know a-priori how the uncertainty may affect the system. For example, implementing both trajectory replanning and control improves significantly the behavior of OctoRoACH, as showcased in Figure 3.22. However, there are still cases in which the robot collides with its environment. These collisions could in principle have been prevented by using a smaller control horizon, at the expense of increasing the computational complexity of the navigation task. Selecting the control horizon is an implementation issue in similar receding-horizon schemes; it ultimately relates to the *amount of uncertainty* the system is subjected to. Similarly, the case study of STAR suggests that collisions may actually be helpful in terms of accommodating for part of the process uncertainty the robot is subjected to (see Figure 3.28). It is thus beneficial to be able to quantify the probability of such events to happen, which in turn can improve the consistency among the tiers of the hierarchical framework.

In particular, to achieve consistency we need to reinforce the mid level with tools that quantify process uncertainty and provide probabilistic guarantees on implementing high (cyber) level actions to the low (physical) level. We do that in the following Chapter 4 by properly quantifying and reproducing within templates the experimentally observed variability.

Chapter 4

A DATA-DRIVEN PROBABILISTIC FRAMEWORK FOR UNCERTAINTY QUANTIFICATION

In the previous chapter, we saw that a combination of templates and motion primitives ensures realizability without oversimplifications. Application to miniature legged robots yields good results, however it may be too restrictive since the approach is purely deterministic. For example, if the volume of obstacles increases, there may not be adequate physical space to permit replanning. Additionally, as Figure 3.22 depicts, there may still be cases that the robot fails to complete a given task. Thus, it is important to have guarantees that high-level policies can be implemented in the physical world despite uncertainty.

In this chapter we develop a methodology that can be used to provide probabilistic guarantees of task accomplishment. The approach reported here allows one to systematically extend an underlying deterministic model to a stochastic regime, and validate the outcome of this procedure against experimental data. In particular, given a model and experimental data, the method provides a way to estimate the magnitude of the uncertainty that needs to be infused in the model in order to capture the range of behaviors observed in experiments, while providing probabilistic guarantees on the validity of the reported model output. Application of the methodology reinforces the mid level by providing *probabilistically-valid templates* that essentially ensure consistency among the levels of the hierarchy despite uncertainty. The stochastically-extended models are capable of capturing and reproducing the variability observed in experimental trials at user-defined levels of fidelity—this is valuable for planning and control in the presence of uncertainty.

4.1 Extending Deterministic Models to Stochastic Regimes

Robot control algorithms are predominantly model-based, and often a large part of the effort prior to deployment is devoted to deriving and validating models that may faithfully represent robot behaviors. By their very nature, robots interact physically with their environment, and these interactions during field deployment become increasingly uncertain. Examples include vehicles operating in partially-known, dynamic environments [10]; legged robots moving on rough terrain [138] or fluidizing ground [121]; quadrotors flying under the influence of uncertain aerodynamic effects [159, 117]; underwater robots affected by uncertain ocean currents [111]; and steerable needles interacting with soft tissue [5]. In many of these examples, deterministic models have limited ability to predict the behavior of the robot as it operates in its environment [150, 149, 27].

The main idea is to parameterize appropriately an otherwise satisfactory deterministic model of the system, to produce an augmented *stochastic* model. Then, randomized algorithms [153, 154, 148] can be used to quantify the extent to which the resulting stochastic model captures the uncertain system-environment interactions. In particular, our method hinges on the concept of checking parameterized *distributions of models* against available experimental data. The probabilistic validation part involves a Monte Carlo simulation for estimating the probability that a random model instantiation is statistically consistent with the measurements. Randomized optimization [153] can then provide approximate near optima for valid model parameters. Thus data variability is integrated within, and can be reproduced by the model. Essentially, data statistics are used to quantify the amount of the uncertainty that the model parameters need to have to capture the variability observed in the experimental data.

4.1.1 Joint Stochastic Model Extension and Probabilistic Validation

The approach combines elements of system identification, model validation and machine learning, and borrows tools from randomized algorithms to render the problem analytically tractable. From a general perspective, system identification techniques focus on learning models and fitting parameters to available data, and offer bounds on the fitting and out-of-sample generalization errors. For instance, linear system identification approaches assign weights to available data and identify their optimal values for linear classification, and linear and logistic regression [2]. If state-space models are required, Linear Time Invariant (LTI) system models can be also obtained [95]. The use of linear models as building blocks supports more powerful nonlinear formulations. For example, cascade products of linear models can generate neural networks [55], and suitable nonlinear transformations give rise to kernel methods [57], such as Volterra models [128, 108]. Genetic algorithms can distill physical laws by selecting nonlinear terms in ODE models [129]; see also [107] for a general overview. However, the models produced typically treat uncertainty as noise, which is either filtered out completely or is used to construct worst-case error bounds.

Model validation [140, 115, 141, 157, 147, 118, 11] uses experimental data, a model of the uncertainty, and a nominal model with its associated error bounds generated by system identification, to report on whether the proposed model can be trusted. These techniques result in *hard* model (in)validation, in the sense that they provide a yes-or-no answer to the question of whether a model is consistent with available data. These methods do not provide sufficient insight on the frequency of the events that result in model invalidation; having this information can be useful for refining the model. Hard model (in)validation can be relaxed in a probabilistic sense by employing tools from statistical learning theory [152, 102]. Some applications involve correlation analysis of residuals [95], prediction error within a robust control framework [49], and computation of relative weighted volumes of convex sets for parametric uncertainty models [90]. Another approach employs a probabilistic model validation methodology to compare a model-generated output probability density function (PDF) with one observed through experiments [53]. That approach relies on the availability of analytic model expressions for uncertainty propagation, and provides sample-complexity bounds for robust validation inference based on randomized algorithms.

Randomized algorithms offer computationally tractable means to tackle problems in control synthesis [83, 153, 154, 22], neural networks [154], and robustness analysis [125, 4]. Typically, deriving worst-case (robust) bounds usually requires a large body of experimental data for theoretical guarantees to hold [153]. However, it has been found that these bounds can be relaxed at the expense of introducing a probabilistic risk, captured by the notion of the *probability of violation* [4, 22, 148]. This concept can be used to allow some design specifications to be violated, albeit with a relatively small probability. In this way, the sample complexity significantly decreases, at the cost of accepting a risk of violation. This idea has been used in system identification to optimally discard sets of small measure from the set of deterministic estimates of design parameters in [32, 33].

Here we employ the notion of probability of violation to turn deterministic models into augmented stochastic models, validating that the latter capture the variability observed in experimental data. The reported approach involves a set-membership characterization of the output PDFs, and applies directly to a wide range of target models, irrespectively of whether they are phenomenological or derived based on first principles. To demonstrate the latter, we apply the framework on two distinct problems: (i) to extend the SFM template to a stochastic regime—see Section 4.3—and (ii) to capture the ground aerodynamic effects in quadrotor ODE models; to keep the development focused on miniature legged robots, the latter is developed in Appendix C.

The reported approach is conceptually related to [54], where an underlying model is used to provide prior information when training a target Gaussian Process model [123] based on the efficient PILCO algorithm [37]. However, the predictive ability of that target model deteriorates significantly when the operating point is shifted even slightly and enters to an area where no data are available. As shown in [70], and reported in Appendix C, the method reported here is more robust in the sense that the resulting stochastic extension can make accurate predictions around different operating points, provided that the induced operating conditions do not change the nature of the mechanisms by which uncertainty affects the system; this is attributed to the fact that our approach makes direct use of a deterministic model that relates to the physics of the underlying process.

4.2 Development of the Framework

The main ingredients of the proposed framework are described in this section. A general account of the method is first presented in Sections 4.2.1 through 4.2.4. A tractable algorithm is then formulated, and made concrete once some assumptions on the underlying statistical distributions are made, in Section 4.2.5. Interspersed between the stages of the conceptual development, are a number of comments that connect the discussion to the examples of Section 4.3 and Appendix C.

4.2.1 Overview

Consider a sample space $\mathcal{W}$ that includes all possible outcomes generated by experiments, where observations are collected from a dynamical process of interest. Each element $w \in \mathcal{W}$ consists of state observations obtained during a single experiment. For example, in Section 4.3 $w \in \mathcal{W}$ will be a motion path for the geometric center of the miniature legged robot OctoRoach [120], when it is implementing a specific low-level controller in open loop.

Suppose that a model $\mathcal{M}$ is available for the dynamical process of interest. The model is parameterized by $\lambda \in \mathbb{N}$ parameters, which are collected in a vector ξ taking values in $\Xi \subset \mathbb{R}^{\lambda}$. For example, in Section 4.3 again, $\mathcal{M}$ takes the form of a stride-to-stride map, while in Appendix C it is a set of differential equations modeling vertical quadrotor flight. As $\xi \in \Xi$ varies, a family of models $\{\mathcal{M}(\xi),\ \xi \in \Xi\}$ is generated; we will refer to each member $\mathcal{M}(\xi)$ obtained for a specific $\xi \in \Xi$ as a *model instantiation* and we will denote $\mathsf{out}(\mathcal{M}(\xi))$ its output.

Typically, given a collection of $I \in \mathbb{N}$ samples $\{w_1, ..., w_I\}$ obtained experimentally, where each $w_i \in \mathcal{W}$, one can compute the value $\bar{\xi} \in \Xi$ of the model parameter vector that results in a model instantiation $\mathcal{M}(\bar{\xi})$, the output $\mathsf{out}(\mathcal{M}(\bar{\xi}))$ of which best reproduces the average of the experimentally observed system behavior. Denoting by

w^{ave} the average of the set of samples w_i for $i \in \{1, .., I\}$, one way to find $\bar{\xi}$ is by solving the least-squares problem

$$\bar{\xi} = \arg\min_{\xi \in \Xi} \sum_{t=1}^{T} \|\mathsf{out}(\mathcal{M}(\xi))_t - w^{\text{ave}}(t)\|^2 \ . \tag{4.1}$$

Here, $t \in 1, \dots, T$ is used to emphasize that both the output of the model instantiation, and the experimental data, are expressed in time series form of length T. Later in Section 4.2.4, this time dependence is made even more explicit.

In many applications, knowing merely the value $\bar{\xi}$ of the parameter vector that results in a best-fit model instantiation $\mathcal{M}(\bar{\xi})$, may not be sufficient. For example, when using a model to plan the motion of a robot in the presence of uncertainty—as is done for instance by [151] and [112]—one needs to know not only the average path behavior, but also the paths' distribution around this average. Only then can one quantify the probability that the robot collides with obstacles.

In this part of the dissertation we provide a new mid-level *tool* that extends deterministic models to a stochastic regime based on experimental data, and provides probabilistic guarantees of validity in doing so. The resulting stochastic model is considered valid when (i) it has low probabilistic risk of producing a response that is not consistent with the experimental data (model fidelity), and (ii) the resulting paths cover as thoroughly as possible of the area marked by the experimental data (model expressiveness). Consequently, this procedure inform us about the range of nominal model parameters, *and* the uncertainty that needs to be infused in the model, so that it can jointly reproduce the experimental data on average and capture the observed variability. The method proposed in this work can be applied to a wide range of models of physical processes, not necessarily expressed in the form of differential equations.

4.2.2 Quantifying Model Fidelity: The Probability of Violation

Associated with the sample space $\mathcal{W}$ is a probability measure $\mathbb{P}_{\mathcal{W}}$, which reflects one's belief regarding how the data of the physical system are distributed in $\mathcal{W}$. For example, in the case of the OctoRoACH implementing a low-level straight-line motion controller in open loop, one expects the majority of the experimentally produced paths to be clustered around a straight line, and hence $\mathbb{P}_{\mathcal{W}}$ should "peak" on this line. It should be emphasized though, that the proposed method does not depend on the specific form of $\mathbb{P}_{\mathcal{W}}$—which is dictated by the physics of the problem—and it can be applied irrespectively of how $\mathbb{P}_{\mathcal{W}}$ is approximated.

Given $\mathcal{W}$ and $\mathbb{P}_{\mathcal{W}}$, a *multisample* $\mathbf{w}$ is defined as a collection of $K \in \mathbb{N}$ independent and identically distributed (i.i.d.) samples w_k, $k \in \{1, ..., K\}$, drawn from $\mathcal{W}$ according to $\mathbb{P}_{\mathcal{W}}$, and is denoted $\mathbf{w} = \{w_1, ..., w_K\}$. Thus, the multisample $\mathbf{w}$ is drawn from the Cartesian product $\mathcal{W}^K = \mathcal{W} \times \cdots \times \mathcal{W}$ (K-times) according to the probability measure $\mathbb{P}_{\mathcal{W}^K}$. In our setting, we generate multisamples by repeating an experiment K times, assuming that each experiment is independent[1] of others and that all experiments are performed under identical conditions. To provide some intuition, Figure 4.1(a) highlights a multisample of $K = 8$ sample paths obtained by implementing a straight-line controller [72] on the miniature legged robot OctoRoACH.

Given a value $\xi \in \Xi$ for the model parameters, we are interested in making a decision as to whether the corresponding model instantiation $\mathcal{M}(\xi)$ will be in agreement, at any time, with the experimentally obtained data. To achieve this, we define a binary-valued decision function $g : \mathcal{W}^K \times \Xi \to \{0, 1\}$ that effectively measures the extent to which the output $\mathsf{out}(\mathcal{M}(\xi))$ of the model instantiation $\mathcal{M}(\xi)$ computed for particular $\xi \in \Xi$ is representative of the data that form the multisample $\mathbf{w} \in \mathcal{W}^K$. To make this statement more precise, we say that the model's output $\mathsf{out}(\mathcal{M}(\xi))$ for $\xi \in \Xi$ is representative of the data forming a multisample $\mathbf{w} \in \mathcal{W}^K$, when this output falls within a prespecified region at level $p \in (0, 1)$ with confidence $\gamma \in (0, 1)$. This region is evaluated based on the data in $\mathbf{w}$ and is centered around the multisample's mean. The

[1] As is often the case, the assumption of independence is difficult to justify in practice. Note though that certain properties that are relevant to our discussion can be extended when the sequence of samples is not i.i.d. but satisfies a "mixing" condition [154, Section 2.5, p. 33]. We will not discuss this issue further, for it requires the introduction of a number of technical results that would shift the focus of this work; the interested reader is referred to [154, Chapter 3].

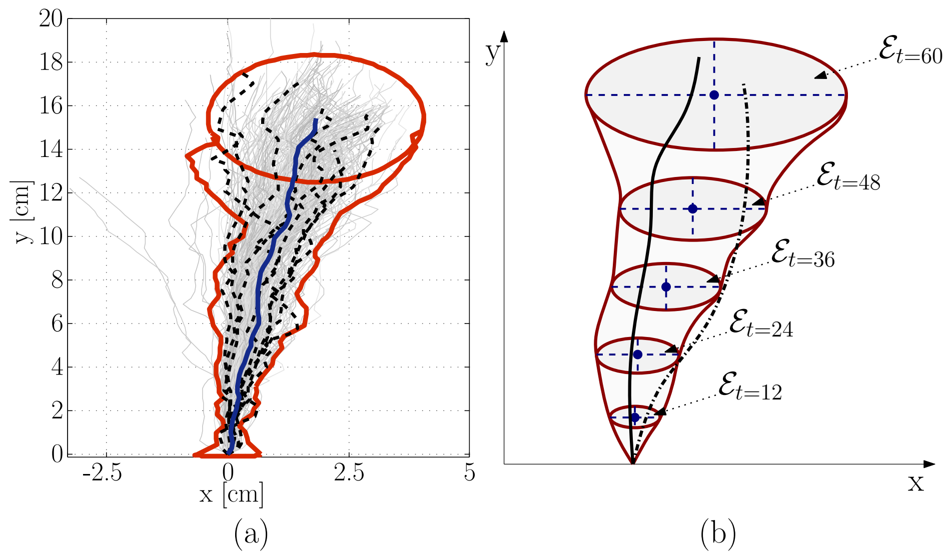


Figure 4.1: (a) A multisample of length $K = 8$. The sample paths of interest are marked with dashed curves, and are superimposed on top of the whole experimental set of paths. The thick solid curve in the center denotes the average of the eight sample paths, while the thick outline denotes the corresponding *cone of data*, explained below in Section 4.2.4.1. (b) A schematic representation for computing the cone of data and the decision function. For each $t \in \{1, \ldots, 60\}$, the *data variability ellipses* $\mathcal{E}_t$ are centered at the sample mean (marked with disks), while their axes are constructed based on sample variances of the multisample; see Sections 4.2.4.1 and 4.2.5. Taking the union of all the ellipses yields the cone of data for a particular multisample. Then, the decision function g reports 0 if a model instantiation never crosses the boundary of the cone of data, as shown with the thick curve, and 1 ("violation") otherwise (as shown with the dashed curve crossing the boundary at $t = 24$).

area covered by the region is called the *cone of data* and is denoted $\mathsf{cone}_{p,\gamma}(\mathbf{w})$. Section 4.2.4.1 below provides details on computing $\mathsf{cone}_{p,\gamma}(\mathbf{w})$ in a general setting, while Section 4.2.5 shows how these computations can become tractable by particularizing on the measure $\mathbb{P}_{\mathcal{W}^K}$ (see Figure 4.1(b)).

The decision function g can now be defined as

$$g(\mathbf{w}, \xi) := \begin{cases} 0, & \text{if } \mathsf{out}(\mathcal{M}(\xi)) \subset \mathsf{cone}_{p,\gamma}(\mathbf{w}) \\ 1, & \text{otherwise} \end{cases} . \tag{4.2}$$

Intuitively, the function g is interpreted as a penalty on a model instantiation $\mathcal{M}(\xi)$ for $\xi \in \Xi$ whenever $\mathcal{M}(\xi)$ produces a behavior that is statistically different from experimental data $\mathbf{w}$. For a given $\xi \in \Xi$, consider the set

$$A_\xi := \left\{\mathbf{w} \in \mathcal{W}^K \mid g(\mathbf{w}, \xi) = 1\right\} \tag{4.3}$$

which contains all the multisamples that violate the condition $\mathsf{out}(\mathcal{M}(\xi)) \subset \mathsf{cone}_{p,\gamma}(\mathbf{w})$. As $\xi \in \Xi$ varies, a collection of sets

$$\mathcal{A} := \{A_\xi,\ \xi \in \Xi\}$$

in $\mathcal{W}^K$ is generated, each of which contains the "bad" multisamples for the corresponding parameter values ξ. The *probability of violation* can then be defined by the function $P : \Xi \to [0,1]$ given by the rule

$$P(\xi) := \mathbb{P}_{\mathcal{W}^K}(A_\xi)\ , \tag{4.4}$$

which provides a measure of the subset $A_\xi \subset \mathcal{W}^K$ of multisamples that are statistically inconsistent with the particular model instantiation $\mathcal{M}(\xi)$. More precisely, for a *given* value $\xi \in \Xi$ of the model's parameters, $P(\xi)$ expresses the likelihood of generating a multisample $\mathbf{w}$ by sampling $\mathcal{W}^K$ according to $\mathbb{P}_{\mathcal{W}^K}$, which—for a desired level $p \in (0,1)$ at confidence $\gamma \in (0,1)$—results in a $\mathsf{cone}_{p,\gamma}(\mathbf{w})$ that does not properly contain the output $\mathsf{out}(\mathcal{M}(\xi))$ of the model instantiation $\mathcal{M}(\xi)$.

The probability of violation (4.4) provides a means of deciding whether a *specific* model instantiation $\mathcal{M}(\xi)$ is probabilistically consistent with experimental data. For a given ξ, a large value for $P(\xi)$ implies that the chance of generating multisamples that are not in agreement with the particular model instantiation $\mathcal{M}(\xi)$ is high, suggesting that the fidelity of $\mathcal{M}(\xi)$ is low.

In addition to quantifying model fidelity, it is also crucial to express multisample variability. This cannot be achieved with a single model instantiation; instead, we need to look at distributions of model instantiations over model parameters.

4.2.3 Distributions of Models and Model Expressiveness

One way in which a model can capture the dispersion of the experimentally generated data is through stochasticity in the model's parameters: their value is the outcome of a random experiment. In particular, consider a sample space Ω_Ξ containing all the possible outcomes of such random experiment and a probability measure $\mathbb{P}_\Xi$ belonging in a family of measures $\mathcal{P}_\Xi$. Then, the parameters of the model form a random vector $\tilde{\xi} : \Omega_\Xi \to \Xi$, the realization of which results in the values $\xi \in \Xi$ that determine the model instantiation $\mathcal{M}(\xi)$. With this construction, for each $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$ a *distribution* of model instantiations $\mathcal{D}_{\mathbb{P}_\Xi} = \{\mathcal{M}, \Xi, \mathbb{P}_\Xi\}$ is defined.

Note that each measure $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$ is assumed here to "peak" around the parameter value $\bar{\xi}$ obtained by the solution of the least-squares optimization problem (3.9). This reflects our intuition that the output of $\mathcal{M}(\bar{\xi})$ is a good representation of the average of the experimental measurements. One can choose the family of measures $\mathcal{P}_\Xi$ in a way that reflects their own beliefs on how the stochasticity enters the nominal model parameters.

The dispersion of the values ξ obtained by sampling the random vector $\tilde{\xi}$ according to different measures in $\mathcal{P}_\Xi$ can be different. As a result, the variability of the outputs produced by the model instantiations $\mathcal{M}(\xi)$ generated by sampling a distribution $\mathcal{D}_{\mathbb{P}_\Xi}$ varies across the collection $\{\mathcal{D}_{\mathbb{P}_\Xi},\ \mathbb{P}_\Xi \in \mathcal{P}_\Xi\}$. The problem now reduces to identifying the distribution $\bar{\mathcal{D}}_{\mathbb{P}_\Xi} \in \{\mathcal{D}_{\mathbb{P}_\Xi},\ \mathbb{P}_\Xi \in \mathcal{P}_\Xi\}$ which captures best the variability in the experimental data without violating a desired specification on the probability of violation of the model instantiations contained in $\bar{\mathcal{D}}_{\mathbb{P}_\Xi}$. The purpose of this section is to make this statement precise.

We begin by providing a way to evaluate the capacity of a distribution of models $\mathcal{D}_{\mathbb{P}_\Xi}$ for capturing the experimental data, given $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$. Consider the set

$$S := \{\xi \in \Xi \mid P(\xi) \geq P_0\} \quad , \tag{4.5}$$

which includes the parameter values ξ that result in model instantiations $\mathcal{M}(\xi)$, each corresponding to a probability of violation exceeding $P_0 \in (0, 1)$. Note that the size of the set S depends on the measure $\mathbb{P}_\Xi$, implying that the likelihood of model instantiations that satisfy (4.5) is different for different distributions $\mathcal{D}_{\mathbb{P}_\Xi}$. Given a desired level $\alpha \in [0, 1)$ we require

$$\mathbb{P}_\Xi(S) \leq \alpha \ . \tag{4.6}$$

Clearly, a lower value of the parameter α corresponds to stricter fidelity standards for the model instantiations $\mathcal{M}(\xi)$ generated by sampling $\mathcal{D}_{\mathbb{P}_\Xi}$. In fact, selecting $\alpha = 0$ implies $P_0 = \sup_{\xi\in\Xi} P(\xi)$, since the probability that a model instantiation $\mathcal{M}(\xi) \in \mathcal{D}_{\mathbb{P}_\Xi}$ results in a probability of violation that exceeds P_0 is required to be zero. Hence, $\alpha = 0$ corresponds to the most conservative way of tuning the behavior expressiveness of a distribution of models $\mathcal{D}_{\mathbb{P}_\Xi}$; that is, $\mathcal{D}_{\mathbb{P}_\Xi}$ is characterized by the model instantiation with the worst performance, in the sense that the corresponding parameters maximize the probability of violation.

In view of (4.5), relaxing α in (4.6) means that we allow a set $S \subset \Xi$ of parameter values to be exceptions to the fidelity rule. The size of S can be explicitly controlled through (4.6) by selecting α so that [153, Section 3]

$$\sup_{\xi\in\Xi\setminus S} P(\xi) \leq P_0 \leq \sup_{\xi\in\Xi} P(\xi) \ ; \tag{4.7}$$

i.e., P_0 is bounded from above by the supremum of the probability of violation over all parameter values (most stringent characterization) and from below by the supremum of the probability of violation over "nearly" all parameter values (turning a blind eye to parameters in S). The implication of (4.7) is that P_0 is a *probable near maximum* (Type 2 near maximum) over the set Ξ of the probability of violation $P(\cdot)$ to the level α [154, Definition 11.2, p. 433]; [153, Section 3]. Note that P_0 depends *both* on the measure $\mathbb{P}_\Xi$ and on the level α; writing $P_0(\mathbb{P}_\Xi, \alpha)$ emphasizes this dependence.

Remark 4 *While the value of the probability of violation (4.4) defined in Section 4.2.2 for a given $\xi \in \Xi$ quantifies the behavior of a single model instantiation $\mathcal{M}(\xi)$, the probable near maximum P_0 to the level α of $P(\cdot)$ over a distribution of models $\mathcal{D}_{\mathbb{P}_\Xi}$ provides a measure of how faithfully $\mathcal{D}_{\mathbb{P}_\Xi}$ captures the experimental data. Indeed, for a* *given level α, the smaller P_0 is for a distribution $\mathcal{D}_{\mathbb{P}_\Xi}$, the more faithful this distribution is in capturing the data.*

We are now ready to provide a precise formulation of the problem described at the beginning of the section: Given (i) a family of model distributions $\{\mathcal{D}_{\mathbb{P}_\Xi},\ \mathbb{P}_\Xi \in \mathcal{P}_\Xi\}$, (ii) a level $\alpha \in [0,1)$, and (iii) a desired fidelity specification $\rho \in [0,1)$, determine the distribution $\bar{\mathcal{D}}_{\mathbb{P}_\Xi}$—or, equivalently, the corresponding measure $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$—that maximizes the dispersion of a random vector $\tilde{\xi}$ of model parameters, provided that the probable near maximum to level α of the probability of violation does not exceed ρ. Mathematically, this translates to finding the measure $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$ that realizes

$$\sup_{\mathbb{P}_\Xi \in \mathcal{P}_\Xi} \operatorname{Tr}\left(\operatorname{Cov}(\tilde{\xi},\tilde{\xi})\right) \tag{4.8}$$

subject to the constraint

$$P_0(\mathbb{P}_\Xi,\alpha) \le \rho \ , \tag{4.9}$$

where $\operatorname{Cov}(\tilde{\xi},\tilde{\xi})$ is the covariance matrix associated with the random vector $\tilde{\xi}$ and $\operatorname{Tr}(\cdot)$ denotes trace.

Implicit here is the assumption that as the measure $\mathbb{P}_\Xi$ changes to make the variance on $\tilde{\xi}$ grow, the model instantiations generated by sampling $\tilde{\xi}$ produce outputs $\mathsf{out}(\mathcal{M}(\xi))$ that are more and more dispersed. Hence, the solution of (4.8)–(4.9) is expected to result in a distribution over the model parameters ξ that allows the corresponding distribution of model instantiations $\mathcal{D}_{\mathbb{P}_\Xi}$ to reproduce, at a given confidence level, as many experimental behaviors observed as possible—not only the average.

4.2.4 A Randomized Approach for Stochastic Model Extension and Probabilistic Validation

This section provides details on computing the decision function g defined by (4.2) in a general context, and proposes a randomized approach for estimating the quantities involved in the implementation of the method via explicit computations.

4.2.4.1 Cone of Data and Decision Function

Consider a multisample $\mathbf{w} = \{w_1, ...w_K\} \in \mathcal{W}^K$ generated experimentally by executing an experiment K times. Each $w_k \in \mathbf{w}$ has the form of a time series

$$w_k = \{(x_{k,1}(t), x_{k,2}(t), ..., x_{k,L}(t))\}_{t \in \{1,...,T\}} \quad ,$$

where $x_{k,\ell}(t)$ is a measurement at time $t \in \{1, ..., T\}$, of the system state indexed $\ell \in \{1, .., L\}$, during the experiment $k \in \{1, ..., K\}$. Time instants $t \in \{1, ..., T\}$ are determined based on the sampling frequency of data collection.

We can associate to each multisample, $\mathbf{w}$, a tolerance interval $\mathcal{I}_{\ell,t}(\mathbf{w})$ to which an experimental trial belongs according to a given probability. A typical way to construct such intervals is to use information about the underlying distribution; for example, in Section 4.2.5 below we assume a normal distribution and use the associated critical values to construct the tolerance intervals [61]. Another possible way is to construct tolerance intervals based on bootstrap [41]. This class of methods relies on resampling of the original data, and can be used to estimate sample distributions of various statistics.

Then, the intervals $\mathcal{I}_{\ell,t}(\mathbf{w})$ provide the basis for constructing the L-dimensional *data variability ellipsoid* $\mathcal{E}_t(\mathbf{w})$. At each time instant $t \in \{1, ..., T\}$, the ellipsoid $\mathcal{E}_t(\mathbf{w})$ is centered at the point $(\bar{x}_1(t), \bar{x}_2(t), ..., \bar{x}_L(t))$, and $\mathcal{I}_{\ell,t}(\mathbf{w})$ for $\ell \in \{1, \ldots, L\}$ are its principal axes (see Figure 4.1(b)). Note that the dependence of $\mathcal{E}_t$ on $\mathbf{w}$ appears explicitly to highlight the fact that these constructions are specific to a given multisample. The cone of data corresponding to a multisample $\mathbf{w} \in \mathcal{W}^K$ at level $p \in (0, 1)$ and confidence $\gamma \in (0, 1)$ is then the union of all L-dimensional ellipsoids

$$\mathsf{cone}_{p,\gamma}(\mathbf{w}) = \bigcup_{t=1}^{T} \mathcal{E}_t(\mathbf{w}) \quad . \tag{4.10}$$

Figure 4.1(b) provides a schematic representation of the cone of data associated with a multisample, for $T = 60$, for the case of the OctoRoACH robot studied in Section 4.3.

To evaluate the decision function g defined by (4.2) for a $\xi \in \Xi$ given $\mathbf{w}$, we need to specify how we check the condition $\mathsf{out}(\mathcal{M}(\xi)) \subset \mathsf{cone}_{p,\gamma}(\mathbf{w})$ for the associated model instantiation $\mathcal{M}(\xi)$. We work element-wise through $\{1, \ldots, T\}$, first by setting

$$\mathsf{out}(\mathcal{M}(\xi)) = \{\mathsf{out}(\mathcal{M}(\xi))_t\}_{t \in \{1,\ldots,T\}} \ ,$$

where for any $t \in \{1, \ldots, T\}$

$$\mathsf{out}(\mathcal{M}(\xi))_t := (x_{\mathcal{M},1}(t), \ldots, x_{\mathcal{M},L}(t)) \ ,$$

and then by defining the indicator function

$$\mathbf{1}_{\mathcal{E}_t(\mathbf{w})}(\mathsf{out}(\mathcal{M}(\xi))_t) := \begin{cases} 1, & \text{if } \mathsf{out}(\mathcal{M}(\xi))_t \in \mathcal{E}_t(\mathbf{w}) \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

that checks the inclusion condition at every $t \in \{1, \ldots, T\}$. In this way the decision function is found as

$$g(\mathbf{w}, \xi) = 1 - \prod_{t=1}^{T} \mathbf{1}_{\mathcal{E}_t(\mathbf{w})}(\mathsf{out}(\mathcal{M}(\xi))_t) \ . \tag{4.12}$$

Note that (4.12) requires the inclusion to hold for all time instants $t \in \{1, ..., T\}$; if at any single t $\mathsf{out}(\mathcal{M}(\xi))_t \notin \mathcal{E}_t(\mathbf{w})$, the decision function is triggered and the model is considered to have violated the fidelity specification.

4.2.4.2 Approximating the Probability of Violation

The probability of violation (4.4) is difficult to compute explicitly, even if the probability measure $\mathbb{P}_{\mathcal{W}^K}$ is analytically available. However, this probability can be effectively approximated empirically [4]. If $\mathbf{W}_M = \{\mathbf{w}_1, ..., \mathbf{w}_M\} \in (\mathcal{W}^K)^M$ is a collection of M multisamples of length K, each drawn from $\mathcal{W}^K$, the empirical probability of violation is

$$\hat{P}(\xi; \mathbf{W}_M) = \frac{1}{M} \sum_{m=1}^{M} g(\xi, \mathbf{w}_m) \ , \tag{4.13}$$

where the dependence of $\hat{P}$ on both the specific collection of multisamples $\mathbf{W}_M = \{\mathbf{w}_1, ..., \mathbf{w}_M\} \in (\mathcal{W}^K)^M$ and on the parameter values $\xi \in \Xi$ that determine the violation set $A_\xi \in \mathcal{A}$ in (4.3) appears explicitly. Note that $\hat{P}(\xi; \mathbf{W}_M)$ is a random variable. For $\varepsilon > 0$, consider [153, Section 4]

$$q(M, \varepsilon, \mathbb{P}_{\mathcal{W}^K}) := \mathbb{P}_{(\mathcal{W}^K)^M} \Big\{ \mathbf{W}_M \in (\mathcal{W}^K)^M : \sup_{\xi \in \Xi} |\hat{P}(\xi; \mathbf{W}_M) - P(\xi)| > \varepsilon \Big\} \ . \tag{4.14}$$

Then, $1 - q(M, \varepsilon, \mathbb{P}_{\mathcal{W}^K})$ is the confidence with which we can say that $\hat{P}(\xi, \mathbf{W}_M)$ is within $\varepsilon > 0$ of the true $P(\xi)$. If $q(M, \varepsilon, \mathbb{P}_{\mathcal{W}^K}) \to 0$ as $M \to \infty$ for any fixed ε, then the empirical probabilities converge uniformly to their true values, implying that the collection of sets $\mathcal{A}$ has the property of uniform convergence of empirical probabilities (UCEP); see [154, Section 3.1, p. 45]. Establishing the UCEP property for the collection of sets $\mathcal{A}$ can be difficult if $\mathcal{A}$ is infinite; but as this turns out to be a finite collection by design, Hoeffding's inequality [154, Lemma 2.7, p. 26] yields

$$q(M, \varepsilon, \mathbb{P}_{\mathcal{W}^K}) \leq 2|\mathcal{A}| \exp\left(-2M\varepsilon^2\right) \ , \tag{4.15}$$

where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$. In fact, since $\{2|\mathcal{A}| \exp\left(-2M\varepsilon^2\right)\}_{M \in \mathbb{N}}$ is summable (see [154, Lemma 2.10, p. 31]), $\hat{P}(\xi; \mathbf{W}_M)$ not only converges uniformly to $P(\xi)$ with M, but also almost surely. The inequality (4.15) can be used to provide bounds for the sample size M that achieves the desired accuracy and confidence specifications.

4.2.4.3 Approximating the Maximum of the Probability of Violation over a Distribution of Models

We have seen in Section 4.2.3 that the expressiveness of a distribution of model instantiations $\mathcal{D}_{\mathbb{P}_\Xi} = \{\mathcal{M}, \Xi, \mathbb{P}_\Xi\}$ for a given probability measure $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$ can be characterized by evaluating a probable near maximum P_0 at level α, of the probability of violation $P(\cdot)$ over $\mathcal{D}_{\mathbb{P}_\Xi}$.

With the probabilistic setting of Section 4.2.3, a collection of $N \in \mathbb{N}$ samples ξ_n, $n \in \{1, ..., N\}$ is drawn according to $\mathbb{P}_\Xi$, thereby resulting in a *parameter multisample* denoted by $\boldsymbol{\xi}_N = \{\xi_1, ..., \xi_N\}$. Note that the parameter multisample $\boldsymbol{\xi}_N$ is drawn from the Cartesian product $\Xi^N = \Xi \times \cdots \times \Xi$ (N-times) according to the measure $\mathbb{P}_{\Xi^N}$. Then, for a given (fixed) *data multisample* $\mathbf{w} \in \mathcal{W}^K$, the probability of violation is a random variable due to its dependence on $\tilde{\xi}$, and $\{P(\xi_1), ..., P(\xi_N)\}$ are the corresponding samples of the probability of violation. Now define

$$\bar{P}_0(\boldsymbol{\xi}_N) := \max_{n \in \{1,..,N\}} P(\xi_n) \ , \tag{4.16}$$

and consider the set

$$\bar{S}_{\boldsymbol{\xi}_N} := \left\{ \xi \in \Xi \mid P(\xi) > \bar{P}_0(\boldsymbol{\xi}_N) \right\} \ , \tag{4.17}$$

which is defined similarly to S in (4.5), only now we have used P_0 instead of $\bar{P}_0(\boldsymbol{\xi}_N)$, computed on the basis of the parameter multisample $\boldsymbol{\xi}_N$. Then, [154, Lemma 11.1, p. 427] asserts that

$$\mathbb{P}_{\Xi^N} \left\{ \boldsymbol{\xi}_N \in \Xi^N \mid \mathbb{P}_{\Xi}(\bar{S}_{\boldsymbol{\xi}_N}) > \alpha \right\} \leq (1-\alpha)^N \ . \tag{4.18}$$

The inequality implies that $\bar{P}_0(\boldsymbol{\xi}_N)$ is an "empirical estimate" of the supremum of the probability of violation $P(\cdot)$ over $\mathcal{D}_{\mathbb{P}_\Xi}$. However, $\bar{P}_0(\boldsymbol{\xi}_N)$ is a different type of estimate compared to $\hat{P}(\xi; \mathbf{W}_M)$ in (4.13), because (4.18) does not require $\bar{P}_0(\boldsymbol{\xi}_N)$ to converge uniformly to the true supremum of $P(\cdot)$. Rather, the claim is that the probability that the violation set $\bar{S}_{\boldsymbol{\xi}_N}$ has small measure, is high. Inequality (4.18) is used to specify the size N of the parameter multisample $\boldsymbol{\xi}_N$.

4.2.5 Algorithm and Implementation

The results described in the previous sections did not assume any particular type of probability distribution. To facilitate computations, however, in what follows, we will assume that $\mathbb{P}_{\mathcal{W}^K}$ corresponds to a (joint) Gaussian distribution.[2] This assumption greatly simplifies the computation of the cone of data associated with a specific multisample $\mathbf{w} \in \mathcal{W}^K$.

[2] We emphasize here that the Gaussianity assumption may not always be the most appropriate choice. In the case of ground mobile robots, for instance, a more appropriate choice for $\mathbb{P}_{\mathcal{W}^K}$ may be the "banana" distribution [97, 149, 27]—essentially a Gaussian distribution in exponential coordinates. The Gaussian assumption is imposed here merely for computational expediency; exploring the different options for constructing the cone data is out of the focus of this work.

Under a Gaussian assumption, each multisample of size K is used to estimate the average of the population of data. For each $\ell \in \{1, ..., L\}$, let

$$\bar{x}_\ell(t) = \frac{1}{K} \sum_{k=1}^{K} x_{k,\ell}(t) \ ,$$

be the sample average at a given time instant $t \in \{1, ..., T\}$, and denote

$$s_\ell(t) = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} \left(x_{k,\ell}(t) - \bar{x}_\ell(t)\right)^2}$$

the corresponding sample standard deviation. Then, the tolerance interval $\mathcal{I}_{\ell,t}(\mathbf{w})$ for the ℓ-th variable associated with the multisample $\mathbf{w}$ at time instant $t \in \{1, ..., T\}$ is

$$\mathcal{I}_{\ell,t}(\mathbf{w}) = [\bar{x}_\ell(t) - k_2\, s_\ell(t),\ \bar{x}_\ell(t) + k_2\, s_\ell(t)] \ . \tag{4.19}$$

The next step is to determine the constant k_2 that defines the two-sided tolerance intervals that cover at least a proportion $p \in (0, 1)$ of the sample size K with confidence level $\gamma \in (0, 1)$. Let $\nu = K - 1$ denote the associated degrees of freedom, $\chi^2_{1-\gamma,\nu}$ the critical value of the chi-square distribution with ν degrees of freedom that is exceeded with probability γ, and $z_{(1-p)/2}$ the critical value of a normal distribution with cumulative probability $(1 - p)/2$. Then, the constant k_2 is given by [61]

$$k_2 = \sqrt{\frac{\nu\left(1 + \frac{1}{K}\right) z^2_{(1-p)/2}}{\chi^2_{1-\gamma,\nu}}} \ . \tag{4.20}$$

The percentiles $\chi^2_{1-\gamma,\nu}$ and $z^2_{(1-p)/2}$ can be found in tables [39] for given values $p \in (0, 1)$ and $\gamma \in (0, 1)$.

To find the appropriate cardinality N of the parameter multisample set ξ_N, we first need to set parameters $p, \gamma, \alpha, \rho > 0$ and $\varepsilon, \delta > 0$, which we collect in Table 4.1. For computational expediency, each measure $\mathbb{P}_\Xi \in \mathcal{P}_\Xi$ is assumed to be associated with a Gaussian random vector $\tilde{\xi}$ with mean $\mathbb{E}[\tilde{\xi}] = \bar{\xi}$ computed by (3.9), and covariance matrix $\mathrm{Cov}(\tilde{\xi}, \tilde{\xi}) = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_\lambda^2)$, where λ is the number of the parameters. Hence, there is an array $\boldsymbol{\sigma} = \{\sigma_1^2, \ldots, \sigma_\lambda^2\}$ parameterizing the family of parameter distributions $\mathcal{P}_\Xi = \{\mathbb{P}_\Xi^{\boldsymbol{\sigma}},\ \boldsymbol{\sigma} \in \mathbb{R}^\lambda\}$. The array is determined by solving the optimization (4.8).

Table 4.1: Key Terminology for Stochastic Model Extension

Description	**Symbol**	**Equation**
Tolerance level for the cone of data	$p \in (0, 1)$	(4.20)
Confidence for tolerance interval	$\gamma \in (0, 1)$	(4.20)
Level of probable near maximum	$\alpha \in [0, 1)$	(4.6)
Model fidelity specification	$\rho \in (0, 1)$	(4.9)
Accuracy level	$\varepsilon \in (0, 1)$	(4.14)
Confidence level	$\delta \in (0, 1)$	(4.21)-(4.22)
Tolerance interval for the cone of data	k_2	(4.20)
Number of parameter multisamples	N	(4.21)
Number of data multisamples	M	(4.22)
Length of each data multisample	K	–
Total number of experimental paths	$I = M \cdot K$	–
Number of state variables	L	–
Number of model parameters	λ	–

Based on (4.18), and for $\delta \in (0, 1)$, N needs to satisfy

$$(1-\alpha)^N \leq \frac{\delta}{2} \iff N \geq \frac{\log \frac{2}{\delta}}{\log \frac{1}{1-\alpha}} \quad , \tag{4.21}$$

so that with confidence $1 - \frac{\delta}{2}$, $\bar{P}_0(\boldsymbol{\xi}_N)$ defined by (4.16) is a probable near maximum of the probability of violation $P(\cdot)$ to a level α. This sampling process results in the finite collection of sets $\mathcal{A} := \{A_{\xi_n},\ \xi_n \in \boldsymbol{\xi}_N\}$ with $|\mathcal{A}| = N$. Then, Hoeffding's inequality (4.15) links the two sample sizes

$$2N \exp\left(-2M\varepsilon^2\right) < \frac{\delta}{2} \iff M \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{4N}{\delta}\right) \quad , \tag{4.22}$$

suggesting that if N parameter multisamples are drawn, then M data multisamples need to be obtained experimentally in order for $\hat{P}(\xi; \mathbf{W}_M)$ to be an empirical estimate of $P(\xi)$ with confidence $1 - \frac{\delta}{2}$. Then, we select a measure $\mathbb{P}_\Xi^{\boldsymbol{\sigma}}$ (through $\boldsymbol{\sigma}$) and generate the parameter multisample $\boldsymbol{\xi}_N = \{\xi_1, ..., \xi_N\}$, sampling $\tilde{\xi}$ according to $\mathbb{P}_\Xi^{\boldsymbol{\sigma}}$.

With M data multisamples $\mathbf{w}_m \in \mathbf{W}_M$, and N parameter multisamples $\xi_n \in \boldsymbol{\xi}_N$ available, the decision function (4.2) is computed explicitly based on (4.11)-(4.12). For each ξ_n, $n \in \{1, \dots, N\}$, (4.13) results in an empirical probability of violation

$$\hat{P}(\xi_n; \mathbf{W}_M) = \frac{1}{M} \sum_{m=1}^{M} g(\xi_n, \mathbf{w}_m) \ . \tag{4.23}$$

Owing to the choice of N and M according to (4.21) and (4.22), we can say with confidence $1 - \delta$ that

$$\hat{P}_0 = \max_{n \in \{1,..,N\}} \hat{P}(\xi_n; \mathbf{W}_M) \tag{4.24}$$

is a *probably approximate near maximum (Type 3 near maximum)* to accuracy ε and level α [154, 153, 83] of the probability of violation in (4.4) over the distribution $\mathcal{D}_{\mathbb{P}_\Xi}$.

The procedure described above relaxes in a probabilistic sense the problem of maximizing the function $P(\cdot)$, through an explicitly computable quantity $\hat{P}_0$ that characterizes the expressiveness of a given model distribution $\mathcal{D}_{\mathbb{P}_\Xi}$. The optimization problem defined by (4.8)–(4.9) is relaxed into the problem of maximizing the variances $\boldsymbol{\sigma} = \{\sigma_1^2, \dots, \sigma_\lambda^2\}$ in the model parameters within a family of Gaussian distributions, which all "peak" at the solution $\bar{\xi}$ of (3.9) while satisfying the constraint

$$\hat{P}_0 \leq \rho \ . \tag{4.25}$$

The output of the algorithm, summarized in Table 4.2, is the largest $\bar{\boldsymbol{\sigma}} = \{\bar{\sigma}_1^2, \dots, \bar{\sigma}_\lambda^2\}$ that $\mathcal{M}(\xi)$ can afford before violating (4.25). Equivalently, the procedure determines the measure $\mathbb{P}_\Xi^{\bar{\boldsymbol{\sigma}}}$ in the form of a Gaussian distribution $\mathcal{N}(\bar{\xi}, \mathrm{diag}\{\bar{\sigma}_1^2, \dots, \bar{\sigma}_\lambda^2\})$, which in turn produces a distribution of models $\mathcal{D}_{\mathbb{P}_\Xi^{\bar{\boldsymbol{\sigma}}}}$ with outputs that cover densely the distribution of experimental data remaining statistically within that cone at level p with confidence γ.

Remark 5 *In the remaining of the chapter, we select $p = 90\%$ and $\gamma = 99\%$. We also set ρ at 35% in order to produce less conservative results. The values of α, ε, and δ directly determine the amount of experimental trials that should be performed,*

Table 4.2: Probabilistically Valid Stochastic Model Extension Algorithm

```
1.  **Require** $p, \gamma, \alpha, \rho > 0, \varepsilon, \delta > 0$, and $K \in \mathbb{N}$.
2.  **Calculate** $k_2$, $N$, and $M$ from (4.20), (4.21), and (4.22), respectively.
3.  **Collect** $M$ data multisamples of size $K$.
4.  **Identify** $\bar{\xi}$ from (3.9) using all available $I = M \cdot K$ data.
5.  **Select** $\boldsymbol{\sigma}$.
6.  **Generate** $\boldsymbol{\xi}_N = \{\xi_1, \xi_2, \ldots, \xi_N\}$ by sampling $\tilde{\xi} \sim \mathbb{P}^{\boldsymbol{\sigma}}_{\Xi}$.
7.  **Calculate** $\mathsf{out}(\mathcal{M}(\xi))_t$ for each $\xi_n, n \in \{1, \ldots, N\}$.
8.  **Construct** $\mathsf{cone}_{p,\gamma}(\mathbf{w}_m)$ for each $\mathbf{w}_m, m \in \{1, \ldots, M\}$
    according to (4.10)
9.  **Calculate** the empirical probability of violation $\hat{P}(\xi_n; \mathbf{W}_M)$
    for each $n \in \{1, \ldots, N\}$ according to (4.23).
10. **Select** the probably approximate near maximum $\hat{P}_0$
    according to (4.24).
11.     **If** $\hat{P}_0 \leq \rho$, increase $\boldsymbol{\sigma}$ and **go to** step 6,
        **else** return $\bar{\boldsymbol{\sigma}}$ and exit.
```

and different combinations yield the same number of trials; we select α, ε so that the experimental trials are kept at a reasonable number (about 250), and out of all possible combinations, we choose the one that maximizes the confidence $1 - \delta$.

4.3 Application to a Miniature Legged Robot

As an example of how the method is used, the developed algorithm is applied to the Switching Four-bar Mechanism (SFM) template.[3] Application of the method extends the SFM to a *probabilistically-valid template* which is shown capable of capturing and reproducing the variability observed in OctoRoACH motion behaviors when the robot navigates at low crawling speeds.

[3] To show that the method is general, a second example that focuses on small-scale aerial vehicles is presented in Appendix C.

4.3.1 Stochastic Extension of the SFM

The SFM template is a step-to-step map. Its state propagates in the body-fixed coordinate frame according to (3.2) or (3.3) depending on the active pair, which can be right or left, respectively. This local state propagation is then mapped to the global frame as Figure 3.2 indicates. To illustrate the effect of infusing uncertainty to different parameter subsets, the model parameterization in this chapter slight differs from the one presented before in Chapter 3. Specifically, we have chosen to use the leg angular velocities instead of fixing the number of steps in a primitive to $N_s = 10$. To facilitate computations, we set $\dot{\phi}_1 = \dot{\phi}_2 = \dot{\phi}_3 = \dot{\phi}_4 = \dot{\phi}$. All touchdown angles attain the same value—i.e. $\phi_1^{\text{td}} = \phi_2^{\text{td}} = \phi_3^{\text{td}} = \phi_4^{\text{td}} = \phi^{\text{td}}$—while liftoff angles of the legs of the same pair take on the same values (that is, $\phi_1^{\text{lo}} = \phi_2^{\text{lo}} = \phi_R^{\text{lo}}$, and $\phi_3^{\text{lo}} = \phi_4^{\text{lo}} = \phi_L^{\text{lo}}$). Note that in CW primitives only the left pair is active necessitating $\phi_R^{\text{lo}} = \phi^{\text{td}}$, and similarly CCW primitives require only the right pair to be active so that $\phi_L^{\text{lo}} = \phi^{\text{td}}$.

According to the aforementioned selections, the parameter space is the 5-tuple

$$\xi = (\phi_R^{\text{lo}}, \phi_L^{\text{lo}}, \dot{\phi}, \phi^{\text{td}}, \theta^{\text{init}}) \ \in \Xi \ .$$

and the step-to-step map is written as $\mathcal{M}(\xi) : \Xi \to \mathbb{R}^2 \times \mathbb{S}$. The output of the SFM is taken here to be the planar position of its geometric center (x_G, y_G), that is

$$\mathsf{out}(\mathcal{M}(\xi)) = (x_G, y_G) \ .$$

To capture the effect of uncertainty in the leg-ground interaction, we extend the SFM to a stochastic setting by randomizing its parameters. In particular, the parameter vector ξ needs to become a random vector $\tilde{\xi}$, drawn according to the multivariate normal distribution

$$\tilde{\xi} \sim \mathcal{N}(\bar{\xi}, \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_5^2)) \ , \tag{4.26}$$

where $\bar{\xi}$ is found by solving the least-squares optimization problem (3.9), and $\sigma_i, \ i = 1, \ldots, 5$ regulate the amount of noise in the model.

Selecting the model parameters over which stochasticity is introduced depends on the nature of the model, and its relation to the physical system it describes. In this particular legged robot example, one source of uncertainty is random leg placement and is associated with randomizing touchdown angles (captured in σ_4). Similarly, randomization of angular velocities (captured in σ_3) is associated with speed irregularities caused by random leg-ground contact, while randomizing liftoff angles (captured in σ_1, and σ_2) suggests that legs lift off the ground randomly—a phenomenon which is attributed to random leg-ground contact, and possible surface irregularities.

4.3.2 Application of the Method

We are now ready to apply the Algorithm of Table 4.2, to estimate the model parameters for the OctoRoach in three distinct modes of motion: (a) *straight line* (SL), (b) 90^o *clockwise turn* (CW), and (c) 90^o *counter-clockwise turn* (CCW).

With respect to Remark 5, we first select $p = 0.90, \gamma = 0.99, \alpha = 0.29, \rho = 0.35, \varepsilon = 0.29$, and $\delta = 0.16$. Based on these selections, $N = 8$, and $M = 31$. Then, we set the length of every multisample equal to $K = 8$, which makes the total number of experimental paths required equal to $I = M \cdot K = 248$, for each mode. Using a VICON motion capture system, we collect experimental measurements of the planar position of the geometric center of the robot (x_G, y_G), and its orientation θ. The collection of K such paths generates a multisample $\mathbf{w}$. The experiments are conducted on a rubber floor mat surface, for a total time of 3 sec at a sampling rate of 20 Hz, yielding $T = 60$. The robot is set into a designated start area with an initial state set at $(x_G,\ y_G,\ \theta) = (0,\ 0,\ 0)$ [cm, cm, deg]. Initial pose errors are shown in Table 4.3, and include measurement noise.

Figure 4.2(a) presents the experimental data. Let $\mathcal{W}_{\text{SL}}$, $\mathcal{W}_{\text{CW}}$, and $\mathcal{W}_{\text{CCW}}$ denote the collections of all planar trajectories of the robot for the SL, CW, and CCW control modes, respectively. The average for each set is marked $w^{\text{ave}}_{\text{SL}}$, $w^{\text{ave}}_{\text{CW}}$, and $w^{\text{ave}}_{\text{CCW}}$, respectively (shown with dashed curves). Dashed outlines represent the cone of data $\mathsf{cone}_{p,\gamma}(\mathcal{W})$ for each case at level $p = 90\%$ with confidence $\gamma = 99\%$. The cone of data is calculated based on $k_2 = 4.147$.[4] The deterministic component of the SFM template

[4] This quantity is found according to (4.20) with 7 degrees of freedom so that the

Table 4.3: Initial pose error statistics

Type	Mean [cm, cm, deg]	Standard Deviation [cm, cm, deg]
CW	$(-0.156, -0.041, 1.23]$	$[0.177, 0.141, 1.37)$
SL	$(-0.007, 0.027, 0.06]$	$[0.234, 0.054, 1.81)$
CCW	$(-0.322, -0.012, 2.50]$	$[0.156, 0.130, 1.23)$

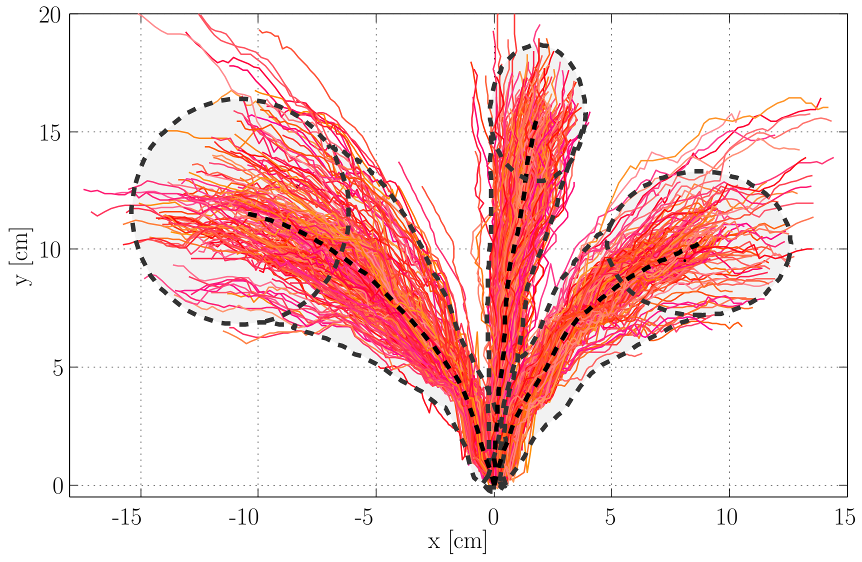
is found by solving (3.9) for each mode of motion; the results are summarized in Table 4.4. With these parameter values, the model produces paths that fit best to the experimental averages (dashed thick curves in Figure 4.2(a)).

Table 4.4: Nominal SFM model parameters

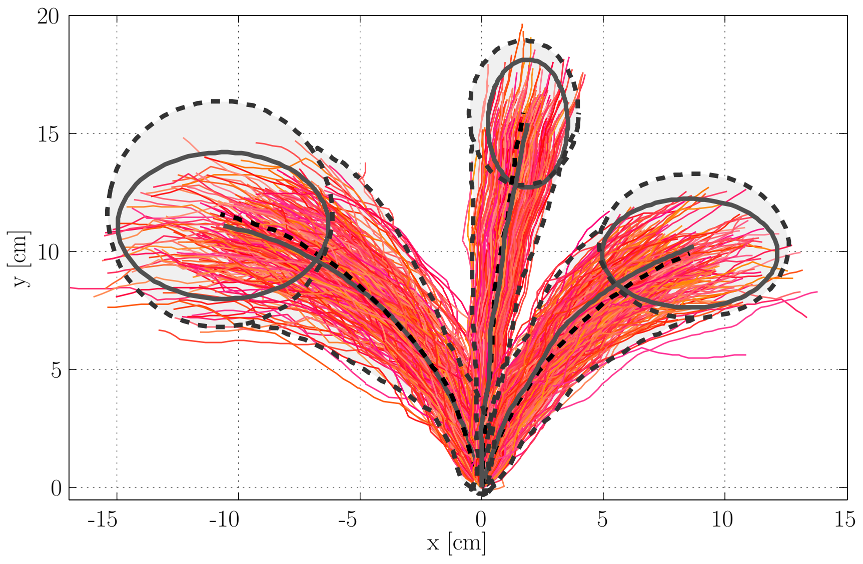
Type	ϕ_R^{lo} [deg]	ϕ_L^{lo} [deg]	$\dot{\phi}$ [deg/sec]	ϕ^{td} [deg]	θ^{init} [deg]
CW	38.54	13.78	4.64	38.54	-16.19
SL	-59.75	-47.19	6.87	0.90	0.00
CCW	14.85	39.59	6.07	39.59	11.86

We investigate here the case where $\sigma_1 = \sigma_2 = \sigma_5 = 0$, assuming that only the leg placement and angular velocity are responsible for the uncertainty observed in the data—in Section 4.3.3 that follows we consider a different model parameter randomization, and compare with the results obtained in this section. Then, we follow the steps 5–11 of the Algorithm in Table 4.2 to estimate $\bar{\sigma}_3$ and $\bar{\sigma}_4$. The result of the procedure is reported in Table 4.5.

lower-tail critical value of the chi-square distribution is equal to $\chi^2_{0.01,7} = 1.239$, and the critical value of the normal distribution is equal to $z_{0.05} = 1.645$.



(a)



(b)

Figure 4.2: (a) Experimental data for the three control models considered. Dashed outlines indicate the *cone of data* for each case, while experimental averages are shown with dashed thick curves in the interior of each cone. (b) Output of the stochastic model, tuned according to the values in Tables 4.4 and 4.5. A set of 248 random model instantiations are plotted over the experimental averages and cones of data. For all cases, the average behavior of the model, marked with a solid curve, remains very close to the experimental average (marked with a dashed curve). The uncertainty ellipses at the final position also match closely.

The estimated normal distribution for the random parameter vector $\tilde{\xi}_{\text{est}}$ for each robot behavior follows from (4.26), with $\bar{\xi}$ shown in Table 4.4, $(x_G^{\text{init}}, y_G^{\text{init}}) = (0,0)$ [cm,cm], and $\bar{\sigma}_i,\ i = 1,\ldots,5$ found in Table 4.5. The associated family of model instantiations $\mathcal{M}(\tilde{\xi}_{\text{est}})$ produces paths that distribute themselves over the area within the marked cone-like boundaries in Figure 4.2(b) to the maximum possible degree, while allowing for a $\rho = 0.35$ probability of leaving at any time t the cone of data created by taking all 248 paths, for each case.

Table 4.5: Probably approximate near maximum SFM model uncertainty

Type	$\bar{\sigma}_1$ [deg]	$\bar{\sigma}_2$ [deg]	$\bar{\sigma}_3$ [deg/sec]	$\bar{\sigma}_4$ [deg]	$\bar{\sigma}_5$ [deg]
CW	0.00	0.00	0.39	17.19	0.00
SL	0.00	0.00	0.62	4.30	0.00
CCW	0.00	0.00	0.13	18.91	0.00

4.3.3 Different Ways to Infuse Stochasticity

In this section we demonstrate the applicability of the proposed framework when a different subset of model parameters is randomized. In particular, we consider the case where $\sigma_3 = \sigma_5 = 0$, assuming this time that the leg placement in both the touchdown and liftoff configurations is responsible for the uncertainty observed in the data. The parameters of the analysis remain the same as before, and we follow the steps 5–11 of the Algorithm in Table 4.2 to estimate $\bar{\sigma}_1,\ \bar{\sigma}_2$ and $\bar{\sigma}_4$. The output of the procedure is reported in Table 4.6.

The random parameter vector $\tilde{\xi}_{\text{est}}$ follows from (4.26), with $\bar{\xi}$ shown in Table 4.4, $(x_G^{\text{init}}, y_G^{\text{init}}) = (0,0)$ [cm,cm], and $\bar{\sigma}_i,\ i = 1,\ldots,5$ tabulated in Table 4.6. The associated family of model instantiations $\mathcal{M}(\tilde{\xi}_{\text{est}})$ produces paths that distribute themselves over the area within the marked cone-like boundaries in Figure 4.3 to the maximum possible degree, while allowing for a $\rho = 0.35$ probability of exiting the cone of data at any time

t. Comparing Figure 4.3 to Figure 4.2, we see that the new randomization produces paths that match the experimentally observed variability as well, albeit the predicted variability in final position for the straight line mode is more conservative. In essence, the method reports that both parameter randomization cases are acceptable solutions. Choosing one over the other ultimately relies on the designer, based on their own beliefs or assumptions on the sources of uncertainty, and which model parameters can best reflect these sources.

Table 4.6: Probably approximate near maximum SFM model uncertainty for the second parametric randomization of Section 4.3.3

Type	$\bar{\sigma}_1$ [deg]	$\bar{\sigma}_2$ [deg]	$\bar{\sigma}_3$ [deg/sec]	$\bar{\sigma}_4$ [deg]	$\bar{\sigma}_5$ [deg]
CW	0.00	8.88	0.00	0.43	0.00
SL	4.58	4.58	0.00	4.58	0.00
CCW	8.60	0.00	0.00	0.51	0.00

Irrespectively of how parameter randomization is performed, the proposed approach comes with probabilistic guarantees of model performance. The guarantees assert that the probability of collecting new experimental paths that will not be captured by the stochastically-extended model, is bounded. The bound is determined by the *user-defined* model fidelity specification ρ. The fidelity specification and the associated probabilistic guarantees may be then be used is support of motion planning under uncertainty. In turn this can help in establishing tradeoffs between risk and task satisfaction under uncertainty within our hierarchical framework. We explore aspects of this topic in the section that follows.

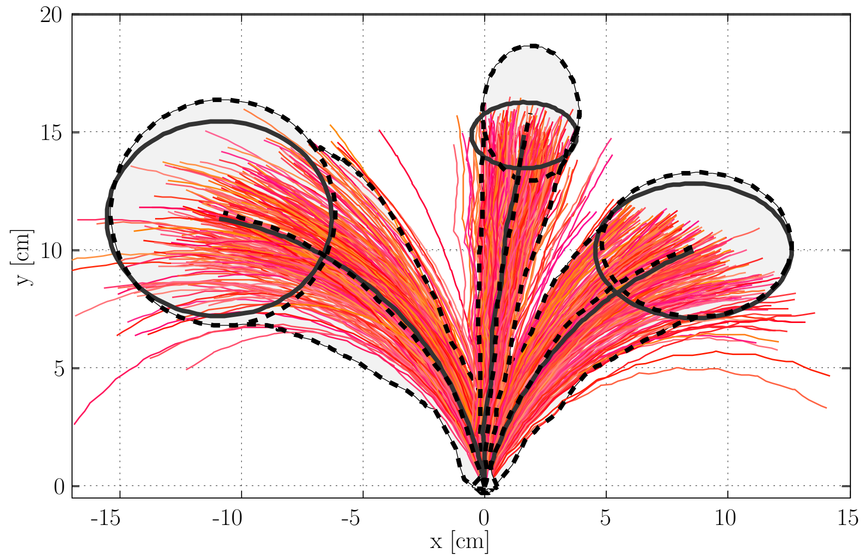


Figure 4.3: Output of the stochastic model, tuned according to the values in Tables 4.4 and 4.6. 248 random model instantiations are plotted over the experimental averages and cones of data. Similarly to Figure 4.2(b), model-predicted statistics follow closely the experimental statistics.

4.4 Outlook: Probabilistically-Valid Templates to Ensure Consistency

The methodology introduced above allows one to *quantify and reproduce* process uncertainty within (stochastic) models or templates. The outcome of the approach defines a probabilistically-valid template which offers system-specific probabilistic guarantees. Having such guarantees is important since they can be used to ensure consistency among the levels of the hierarchical control framework in several ways.

Augmenting obstacles

Probabilistically-valid templates can be useful in augmenting obstacles during robot motion planning. As it was shown earlier in Chapter 3, augmenting the actual obstacles is important so that the motion planner generates safe reference trajectories to reassure that the robot will not collide. A recipe that typically works well in deterministic cases is to augment the obstacles accordingly to the larger dimension of the robot (as in Chapter 3). However this may not be sufficient when the robot is subject to process uncertainty. Instead, the obstacle regions should be augmented by linking safety specifications to the user-defined model fidelity and the associated cone of data predicted by the stochastically-extended model. For example, the work reported previously suggests that the augmented obstacles regions must be larger for the case of the OctoRoach compared to the case of spidar. As a result, the design of reference trajectories is informed by the actual robot behavior observed in experiments, and is biased toward improving the chances of avoiding obstacle collisions.

Estimating probabilities of task accomplishment

Probabilistically-valid templates also relate the design of "uncertainty-informed" reference trajectories to estimating the probability of achieving a desired task. Navigation tasks, for instance, require both obstacle avoidance and goal attainment. To estimate the probability of collisions and goal attainment, it is important to *propagate* the uncertainty with the template at hand. It is customary to propagate the uncertainty numerically through Monte Carlo simulations. Essentially, candidate reference trajectories are run in simulation multiple times to numerically estimate the probability of achieving a desired task—e.g., avoiding obstacles and attaining the desired goal. While effective, this approach tends to become computationally demanding as the number of candidate trajectories, or their length increase. Thus, a useful probabilistically-valid template should allow one to propagate the uncertainty analytically as well.

The Switching Four-bar Mechanism is one such template. Foremost, the geometric equations for step-to-step state propagation[5] are kinematic, and employ only one degree of freedom. Additionally, the availability of analytic expressions renders state propagation direct, fast, and exact. These features speed up computations since, given the initial state, touchdown configuration, and a desired liftoff configuration, the next state is determined through a single function evaluation. A careful selection of where to infuse stochasticity into the model permits analytic uncertainty propagation. For instance, the randomization approaches analyzed before in Section 4.3 render the uncertainty propagation non-trivial since the uncertainty has to be propagated through

[5] See Appendix B for details on the derivations.

the nonlinear state propagation equations (3.2) and (3.3) in the local frame, and then mapped to the global frame, for each step. This limitation can be circumvented by infusing stochasticity at the end of each step. The model progresses deterministically during a step, and the resulting state is perturbed before the next step is initialized (see Figure 3.19). The probabilistic methodology for extending models to stochastic settings provides a measure of how large this stochastic perturbation must be to capture and reproduce the observed uncertain robot behaviors. This last randomization approach may be particularly suitable for estimating the probability of achieving a desired task since it permits a direct, analytic way to propagate uncertainty with the SFM template.

Establishing tradeoffs between risk and task accomplishment

A probabilistically-valid template can also be used to establish tradeoffs between risk and satisfying a required task. This can be particularly helpful when performing more complex tasks, such as visibility-constrained surveillance (see Figure 3.25). Essentially, the probabilistically-valid template quantifies the risk caused by process uncertainty, and uses this information to evaluate reference trajectories based on the tradeoffs among competing constraints of the problem. For example, longer paths typically have less probability of reaching a particular desired state, yet they may offer greater visibility coverage. Another example is balancing speed of reaching a target versus safety; going faster also results in higher levels of uncertainty in robot motion. The probabilistically-valid template then reports back to the high level regarding these tradeoffs, and informs as to whether a reference trajectory should be revised. In turn, this process can consolidate low-level execution with high-level task planning under uncertainty.

4.5 Discussion

To summarize, the probabilistic method presented here focuses on uncertainty quantification, and offers an intuitive and tractable means to capture and reproduce motion uncertainty with stochastically-extended models. Uncertainty is infused in model parameters, and a randomized optimization algorithm estimates the magnitude of the infused uncertainty. The method comes with probabilistic guarantees of model performance that bound the probability of generating new experimental paths that are not consistent with a template. The resulting template is termed *probabilistically-valid template*.

Probabilistic guarantees of model performance are important for making informed choices on the type of appropriate planners and controllers [24]. In addition, once the data variability is captured within the model, analytic tools can be brought to bear for propagating the uncertainty as the physical process evolves [27]. This can be useful in applications such as motion planning in the presence of uncertainty [149, 76, 99, 17, 151, 112], and filtering and estimation [149, 1, 97].

Probabilistically-valid templates are key for consolidating low-level execution with high-level task planning under uncertainty. Foremost, they can be used together with uncertainty propagation to estimate the risk in achieving high-level policies based on the uncertainty exhibited by the real system. They can also be used to establish performance tradeoffs in the presence of uncertainty and instruct as to how to revise a high-level policy to meet competing objectives. These points are discussed through some representative examples that may motivate interesting future research directions.

Last but not least, the reported approach is general and may find application in multiple areas in robotics. Indeed, it can support applications in which data variability plays a significant role, but it is either assumed or provided without any probabilistic guarantees. Robotics applications in this realm that can benefit from probabilistically-valid models (and templates) include stochastic control design [135, 7]; Linear Temporal Logic (LTL) control of uncertain vehicles [29]; model verification [145] and calibration [133]; and needle steering [5], to name a few. Moreover, the method here can supplement stochastic trajectory optimization techniques like STOMP [66], by identifying—based on data—uncertain terms in the optimization function used therein. In Appendix C we present results on applying the method to small-scale aerial vehicles.

Chapter 5
CONCLUSIONS AND FUTURE DIRECTIONS

We conclude with a short summary of the contributions of this work, and highlight some new research directions that have been made possible. Taking the contributions together, we hope that—quoting Aristotle—the "*whole is greater than the sum of its parts*." The contributions of this work may provide the foundations of a general hierarchical framework for multi-robot planning and control under uncertainty. The reported results push forward the state-of-the-art on motion planning and navigation of miniature legged robots, and may also find application in other areas of robotics where making decision under uncertainty is important.

5.1 Hierarchical Control for Uncertainty in Robot Navigation

The overarching idea in this work is that a three-level hierarchical control framework is effective for dealing with uncertainty in navigation; this claim is validated experimentally in the domain of miniature legged robots. In the three levels we find a high level responsible for task planning, a mid level that focuses on trajectory generation, and a low level that performs trajectory tracking control. The reported results flesh out aspects of the mid level of the hierarchy, and essentially provide tools that ensure consistency among the different levels in the presence of uncertainty. A range of different techniques developed in the low and high levels is thus bridged, by employing our *probabilistically-valid templates* which are simple, low-dimensional probabilistic models validated against experimental data, and which facilitate analysis and control.

5.1.1 Selecting Suitable Templates is Key

Appropriate templates can be used to ensure that the different levels of the hierarchical framework are compatible with each other. Such templates facilitate low-level trajectory tracking control, and are amenable to primitives-based motion planning. When this happens, we are able to ensure low-level realizability without oversimplifying the dynamics of the physical system at hand. We show this point by performing real-time navigation with miniature legged robots in obstacle-cluttered environments.

5.1.2 Experimental Data Guarantee Low-Level Implementation of High-Level Policies despite Uncertainty

To ensure that the results based on the proposed approach are not overly restrictive, a data-driven probabilistic framework is also developed. The aim of this framework is to capture the uncertainty that is observed in experimental data, and infuse it back into a model so that the latter can capture the variability observed in the targeted robot behavior it reproduces. The parameters of an appropriate deterministic model are turned into random variables, the statistics of which are identified and validated against experimental data. The resulting stochastic model may then be used to quantify risk and restore performance guarantees in the presence of uncertainty.

5.1.3 Miniature Legged Robots Fit Well in the Framework

Miniature legged robots offer a great testbed to test and develop the aforementioned key tools of the framework. Uncertainty—associated with various manufacturing variabilities, uncertain mechanical properties of structural material, and inherently uncertain leg-ground interactions—affects the behavior of these robots in a natural and immediately visible way. Due to the stringent size and weight specifications that limit reliance on extensive feedback control, noise cannot be fully suppressed, but hopefully can be managed if appropriately quantified. The reported results meet this challenge.

5.2 Dealing with Uncertainty in Robot Planning, Navigation, and Control

The tools provided in this dissertation are key components for a general hierarchical framework for dealing with uncertainty in robot navigation and control; however, further investigation is needed to accomplish the goal.

5.2.1 Extending Low-Level Control to Bio-Inspired Templates

We need to extend low-level control approaches to bio-inspired legged locomotion templates. Indeed, research on control of wheeled vehicles has contributed several important and efficient techniques; it would be beneficial to port these techniques to the domain of small-scale legged robots. However, it is still not clear how existing bio-inspired templates can be adjusted to facilitate the application of wheel-based control techniques (e.g., model predictive navigation [73]). The Switching Four-bar Mechanism (SFM) template developed here is a first step to enabling the application of kinematic control approaches to small-scale legged locomotion. Porting dynamic control approaches remains an open problem.

5.2.2 Dealing with Uncertainty in Perception and the Environment

Robot performance in real-world settings depends on the interactions of action, perception, and the environment (Figure 1.1), all of which introduce different types of uncertainty. The results reported in this dissertation shed some light on how to deal with action uncertainty through *probabilistically-valid templates*. Uncertainty in perception may be tackled through *multi-sensor data fusion*, more delicate *reasoning* on the nature of obstacles, features, and other points of interest, or *learning*. The latter may also be used to deal with uncertainties in dynamically-changing environments, using for example *Grammatical Inference* [36] as in [26, 46]. It is conceivable that we can incorporate all three components into a single, unifying framework.

5.2.3 Uncertainty in Planning, Navigation, and Control of Multi-Robot Systems

A research direction that could be developed in parallel is the extension of the proposed framework to cases that involve multiple, possibly heterogeneous, robots. The interactions among robots add to the complexity of the overall problem, however they may be treated through *game-theoretical* approaches. Adding this research directions to the framework touches on a more general problem of how to tackle uncertainties in Cyber-Physical Systems.

5.3 Uncertainty in Multi-Robot Cyber-Physical Systems

Modern Cyber-Physical Systems (CPSs) that include robots in real-world applications are rapidly gaining momentum (for example in hospitals). The proposed hierarchical approach is anticipated to provide novel modules for harnessing uncertainty in multi-robot CPSs. One such example is drawn from the area of miniature legged robots; robots may act as remote sensors in human-in-the-loop systems for search-and-rescue missions. This dissertation lays foundations for realizing the full potential of miniature legged robots in real-worlds applications.

5.4 Frontiers in Robotics, Control, and Small-Scale Animal Locomotion

This work can be cast within the context of biorobotics [63]. In particular, the findings here suggest that control-oriented approaches may drive new design paradigms for miniature legged robots, and may provide intuition on the mechanisms that uncertainty affects the locomotion of small-scale animals.

REFERENCES

[1] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun. Discriminative training of kalman filters. In *Robotics: Science and Systems*, pages 289–296. MIT Press, 2005.

[2] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning From Data.* AMLBook, 2012.

[3] A. P. Aguiar and J. P. Hespanha. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[4] T. Alamo, R. Tempo, and E. F. Camacho. Randomized Strategies for Probabilistic Solutions of Uncertain Feasibility and Optimization Problems. *IEEE Transactions on Automatic Control*, 54(11):2545–2559, 2009.

[5] R. Alterovitz, M. Branicky, and K. Goldberg. Motion Planning Under Uncertainty for Image-guided Medical Needle Steering. *The International Journal of Robotics Research*, 27(11-12):1361–1374, 2008.

[6] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[7] R. P. Anderson and D. Milutinović. A Stochastic Optimal Enhancement of Feedback Control for Unicycle Formations. *Robotica*, 32(2):305–324, 2014.

[8] S. B. Andersson and D. Hristu. Symbolic feedback control for navigation. *IEEE Transactions on Automatic Control*, 51(6):926–937, 2006.

[9] E. Andrada, C. Rode, and R. Blickhan. Grounded running in quails: Simulations indicate benefits of observed fixed aperture angle between legs before touch-down. *Journal of Theoretical Biology*, 335:97–107, 2013.

[10] G. Aoude, B. Luders, J. M. Joseph, N. Roy, and J. P. How. Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns. *Autonomous Robots*, 35:51–76, 2013.

[11] C. Baier and J. P. Katoen. *Principles of Model Checking.* The MIT Press, 2008.

[12] A. T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, and R. J. Wood. High speed locomotion for a quadrupedal microrobot. *The International Journal of Robotics Research*, 33(8):1063–1082, 2014.

[13] A. T. Baisch, P. Sreetharan, and R. J. Wood. Biologically-inspired locomotion of a 2g hexapod robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5360–5365, 2010.

[14] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic planning and control of robot motion [Grand Challenges of Robotics]. *IEEE Robotics Automation Magazine*, 14(1):61–70, 2007.

[15] M. Binnard and M. R. Cutkosky. A design by composition approach for layered manufacturing. *ASME Journal of Mechanical Design*, 122(1):91–101, 2000.

[16] P. Birkmeyer, K. Peterson, and R. S. Fearing. DASH: A dynamic 16g hexapedal robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2683–2689, 2009.

[17] L. Blackmore, M. Ono, and B. Williams. Chance-Constrained Optimal Path Planning with Obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094, 2011.

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

[19] R. Blickhan and R. J. Full. Locomotion energetics of ghost crab. II. Mechanics of the center of mass during walking and running. *Journal of Experimental Biology*, 130(1):155–174, 1987.

[20] R. Blickhan and R. J. Full. Similarity in Multilegged Locomotion: Bouncing Like a Monopode. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173:509–517, 1993.

[21] C. Y. Brown, D. E. Vogtmann, and S. Bergbreiter. Efficiency and effectiveness analysis of a new direct drive miniature quadruped robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5631–5637, 2013.

[22] G. C. Calafiore, F. Dabbene, and R. Tempo. Research on Probabilistic Methods for Control System Design. *Automatica*, 47(7):1279–1293, 2011.

[23] G. A. Cavagna, N. C. Heglund, and C. R. Taylor. Mechanical work in terrestrial locomotion: two basic mechanisms for minimizing energy expenditure. *American Journal of Physiology- Regulatory, Integrative and Comparative Physiology*, 233:243–261, 1977.

[24] A. Censi, D. Calisi, A. De Luca, and G. Oriolo. A bayesian framework for optimal motion planning with uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1798–1805, 2008.

[25] J. G. Cham, S. A. Bailey, J. E. Clark, R. J. Full, and M. R. Cutkosky. Fast and Robust: Hexapedal Robots via Shape Deposition Manufacturing. *The International Journal of Robotics Research*, 21(10-11):869–882, 2002.

[26] J. Chandlee, J. Fu, K. Karydis, C. Koirala, J. Heinz, and H. G. Tanner. Integrating grammatical inference into robotic planning. In Jeffrey Heinz, Colin de la Higuera, and Tim Oates, editors, *Proceedings of the 11th International Conference on Grammatical Inference*, volume 21, pages 69–83, 2012.

[27] G. S. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications.* Birkhauser, 2012.

[28] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation.* MIT Press, 2005.

[29] I. Cizelj and C. Belta. Control of noisy differential-drive vehicles from time-bounded temporal logic specifications. *The International Journal of Robotics Research*, 33(8):1112–1129, 2014.

[30] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.

[31] N. J. Cowan, M. M. Ankarali, J. P. Dyhr, M. S. Madhav, E. Roth, S. Sefati, S. Sponberg, S. A. Stamper, E. S. Fortune, and T. L. Daniel. Feedback control as a framework for understanding tradeoffs in biology. *Integrative and Comparative Biology*, 54(2):223–237, 2014.

[32] F. Dabbene, M. Sznaier, and R. Tempo. A probabilistic approach to optimal estimation - part I: Problem formulation and methodology. In *Proceedings of the 51th IEEE Conference on Decision and Control*, pages 190–195, 2012.

[33] F. Dabbene, M. Sznaier, and R. Tempo. A probabilistic approach to optimal estimation - part II: Problem formulation and methodology. In *Proceedings of the 51th IEEE Conference on Decision and Control*, pages 196–201, 2012.

[34] N. Dantam and M. Stilman. The motion grammar: Linguistic perception, planning, and control. In *Proceedings of Robotics: Science and Systems*, 2011.

[35] A. De, K. S. Bayer, and D. E. Koditschek. Active sensing for dynamic, non-holonomic, robust visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6192–6198, 2014.

[36] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars.* Cambridge University Press, 2010.

[37] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the International Conference on Machine Learning*, 2011.

[38] F. Delmotte, T. R. Mehta, and M. Egerstedt. A software tool for hybrid control. *IEEE Robotics Automation Magazine*, 15(1):87–95, 2008.

[39] W. J. Dixon and F. J. Massey. *Introduction to Statistical Analysis.* McGraw-Hill Companies, 4th edition, 1984.

[40] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[41] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap.* CRC Press, 1994.

[42] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. 1990.

[43] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):342–352, 2009.

[44] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[45] R. A. Freeman and P. V. Kokotovic. *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques.* Birkhäuser, 1996.

[46] J. Fu, H. G. Tanner, J. N. Heinz, K. Karydis, J. Chandlee, and Koirala C. Symbolic planning and control using game theory and grammatical inference. *Engineering Applications of Artificial Intelligence*, 37:378–391, 2015.

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

[48] F. Garcia Bermudez, R. Julian, D. W. Haldane, P. Abbeel, and R. S. Fearing. Performance analysis and terrain classification for a legged robot over rough terrain. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 513–519, 2012.

[49] M. Gevers, X. Bombois, B. Codrons, G. Scorletti, and B. D. O. Anderson. Model Validation for Control and Controller Validation in a Prediction Error Identification Framework – Part I: Theory. *Automatica*, 39(3):403–415, 2003.

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

[51] D. W. Haldane, C. S. Casarez, J. T. Karras, J. Lee, C. Li, A. O. Pullin, E. W. Schaler, D. Yun, H. Ota, A. Javey, and R. S. Fearing. Integrated manufacture of exoskeletons and sensing structures for folded millirobots. *AMSE Journal of Mechanisms and Robotics*, 7(2):021011–1–021011–19, 2015.

[52] D. W. Haldane, K. Peterson, F. Garcia Bermudez, and R. S. Fearing. Animal-inspired design and aerodynamic stabilization of a hexapedal millirobot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3279–3286, 2013.

[53] A. Halder and R. Bhattacharya. Probabilistic Model Validation for Uncertain Nonlinear Systems. *Automatica*, 50(8):2038–2050, 2014.

[54] J. Hall, C. E. Rasmussen, and J. Maciejowski. Modelling and control of nonlinear systems using gaussian processes with partial model information. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 5266–5271, 2012.

[55] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice-Hall, Upper Saddle River, NJ, 1999.

[56] K. L. Hoffman and R. J. Wood. Towards a multi-segment ambulatory microrobot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1196–1202, 2010.

[57] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel Methods in Machine Learning. *Annals of Statistics*, 36(3):1171–1220, 2008.

[58] P. Holmes, R. J. Full, D. E. Koditschek, and J. Guckenheimer. The Dynamics of Legged Locomotion: Models, Analyses, and Challenges. *SIAM Review*, 48(2):207–304, 2006.

[59] A. M. Hoover, S. Burden, X-Y. Fu, S. Sastry, and R. S. Fearing. Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot. In *Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics*, pages 869–876, 2010.

[60] A. M. Hoover, E. Steltz, and R. S. Fearing. RoACH: An autonomous 2.4g crawling hexapod robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 26–33, 2008.

[61] W. G. Howe. Two-sided tolerance limits for normal populations - some improvements. *Journal of the American Statistical Association*, 64:610–620, 1969.

[62] D. Hristu-Varsakelis, M. Egerstedt, and P. S. Krishnaprasad. On the structural complexity of the motion description language MDLe. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3360–3365, 2003.

[63] A. Ijspeert. Biorobotics: Using robots to emulate and investigate agile animal locomotion. *Science*, 346(6206):196–203, 2014.

[64] D. L. Jindrich and R. J. Full. Many-Legged Maneuverability: Dynamics of Turning in Hexapods. *The Journal of Experimental Biology*, 202:1603–1623, 1999.

[65] B. Johnson and H. Kress-Gazit. Analyzing and revising synthesized controllers for robots with sensing and actuation errors. *The International Journal of Robotics Research*, 34(6):816–832, 2015.

[66] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011.

[67] S. Karaman and E. Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5041–5047, 2013.

[68] K. Karydis, Y. Liu, I. Poulakakis, and H. G. Tanner. A Template Candidate for Miniature Legged Robots in Quasi-Static Motion. *Autonomous Robots*, 38(2):193–209, 2015.

[69] K. Karydis, Y. Liu, I. Poulakakis, and H. G. Tanner. Navigation of miniature legged robots using a new template. In *23rd Mediterranean Conference on Control and Automation*, pages 1112–1117, 2015.

[70] K. Karydis, I. Poulakakis, J. Sun, and H. G. Tanner. Probabilistically Valid Stochastic Extensions of Deterministic Models for Systems with Uncertainty. *The International Journal of Robotics Research*, 34(10):1278–1295, 2015.

[71] K. Karydis, I. Poulakakis, and H. G. Tanner. A switching kinematic model for an octapedal robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 507–512, 2012.

[72] K. Karydis, I. Poulakakis, and H. G. Tanner. Probabilistic validation of a stochastic kinematic model for an eight-legged robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2562–2567, 2013.

[73] K. Karydis, L. Valbuena, and H. G. Tanner. Model predictive navigation for position and orientation control of nonholonomic vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3206–3211, 2012.

[74] K. Karydis, D. Zarrouk, I. Poulakakis, R. S. Fearing, and H. G. Tanner. Planning with the STAR(s). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3033–3038, 2014.

[75] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[76] G. Kewani, G. Ishigami, and K. Iagnamma. Stochastic mobility-based path planning in uncertain environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1183–1189, 2009.

[77] S. Kim, J. E. Clark, and M. R. Cutkosky. iSprawl: Design and Tuning for High-speed Autonomous Open-loop Running. *The International Journal of Robotics Research*, 25(9):903–912, 2006.

[78] E. Klavins. A computation and control language for multi-vehicle systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 4133–4139, 2003.

[79] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[80] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics*, 26(1):48–61, 2010.

[81] D. E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11(4):412–442, 1990.

[82] N. J. Kohut, A. M. Hoover, K. Y. Ma, S. S. Baek, and R. S. Fearing. MEDIC: A legged millirobot utilizing novel obstacle traversal. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 802–808, 2011.

[83] V. Koltchinskii, C. T. Abdallah, M. Ariola, P. Dorato, and D. Panchenko. Statistical learning control of uncertain systems: it is better than it seems. Technical Report EECE-TR-00-001, University of New Mexico, 2000.

[84] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

[85] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal logic based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

[86] M. Lahijanian, S. B. Andersson, and C. Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Transactions on Robotics*, 28(2):396–409, 2012.

[87] B. G. A. Lambrecht, A. D. Horchler, and R. D. Quinn. A small, insect-inspired robot that runs and jumps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1240–1245, 2005.

[88] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006.

[89] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[90] L. H. Lee and K. Poolla. On Statistical Model Validation. *Journal of Dynamic Systems, Measurement, and Control*, 118(2):226–236, 1996.

[91] C. Li, A. M. Hoover, P. Birkmeyer, P. B. Umbanhowar, R. S. Fearing, and D. I. Goldman. Systematic study of the performance of small robots on controlled laboratory substrates. In *Proceedings of the SPIE Conference on Micro- and Nanotechnology Sensors, Systems, and Applications II*, volume 7679, pages 76790Z–13, 2010.

[92] C. Li, A. O. Pullin, D. W. Haldane, H. K. Lam, R. S. Fearing, and R. J. Full. Terradynamically streamlined shapes in animals and robots enhance traversability through densely cluttered terrain. *Bioinspiration & Biomimetics*, 10(4):046003, 2015.

[93] C. Li, T. Zhang, and D. I. Goldman. A terradynamics of legged locomotion on granular media. *Science*, 339:1408–1412, 2013.

[94] D. Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction.* Princeton University Press, 2011.

[95] L. Ljung. *System Identification: Theory for the User.* Prentice-Hall, 1999.

[96] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, volume 1, pages 153–158, 2004.

[97] A. W. Long, K. C. Wolfe, M. Mashner, and G. S. Chirikjian. The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates. In *Robotics: Science and Systems*, 2012.

[98] G. A. D. Lopes and D. E. Koditschek. Visual Servoing for Nonholonomically Constrained Three Degree of Freedom Kinematic Systems. *The International Journal of Robotics Research*, 26(7):715–736, 2007.

[99] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1642–1648, 2010.

[100] P. Martin, E. Johnson, T. Murphey, and M. Egerstedt. Constructing and implementing motion programs for robotic marionettes. *IEEE Transactions on Automatic Control*, 56(4):902–907, 2011.

[101] A. Mathis, J. Russell, T. Moore, J. Cohen, B. Satterfield, N. Kohut, X.-Y. Fu, and R. S. Fearing. Autonomous navigation of a 5 gram crawling millirobot in a complex environment. In *Proceedings of Adaptive Mobile Robotics: 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 121–128, 2012.

[102] S. Mendelson. A Few Notes on Statistical Learning Theory. In S. Mendelson and E. J. Smola, editors, *Advanced Lectures in Machine Learning*, volume 2600, pages 1–40. Springer Verlag, 2003.

[103] R. Merz, F. B. Prinz, K. Ramaswami, M. Terk, and L. E. Weiss. Shape deposition manufacturing. In *Proceedings of the Solid Freeform Fabrication Symposium*, 1994.

[104] R. Milner. *Communication and concurrency.* PHI Series in computer science. Prentice Hall, 1989.

[105] J-M. Mongeau, B. McRae, A. Jusufi, P. Birkmeyer, A. M. Hoover, R. S. Fearing, and R. J. Full. Rapid Inversion: Running Animals and Robots Swing like a Pendulum under Ledges. *PLoS ONE*, 7(6):e38003, 2012.

[106] J. Morrey, B. G. A. Lambrecht, A. Horchler, R. Ritzmann, and R. D. Quinn. Highly mobile and robust small quadruped robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 82–87, 2003.

[107] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[108] T. Ogunfunmi. *Adaptive Nonlinear System Identification: The Volterra and Wiener Model Approaches.* Signal and Communication Technology. Springer-Verlag, New York, NY, 2007.

[109] C. D. Onal, M. T. Tolley, R. J. Wood, and D. Rus. Origami-Inspired Printed Robots. *IEEE/ASME Transactions on Mechatronics*, PP(99):1–8, 2014.

[110] D. Panagou and H. G. Tanner. Modeling of a Hexapod Robot; Kinematic Equivalence to a Unicycle. Technical Report UDMETR-2009-001, University of Delaware, 2009.

[111] A. A. Pereira, J. Binney, G. A. Hollinger, and G. S. Sukhatme. Risk-aware Path Planning for Autonomous Underwater Vehicles using Predictive Ocean Models. *Journal of Field Robotics*, 30(5):741–762, 2013.

[112] M. Pivtoraiko, D. Mellinger, and V. Kumar. Incremental micro-UAV motion replanning for exploring unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2452–2458, 2013.

[113] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[114] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[115] K. Poolla, P. Khargonekar, A. Tikku, James Krause, and K. Nagpal. A Time-domain Approach to Model Validation. *Transactions on Automatic Control*, 39(5):951–959, 1994.

[116] I. Poulakakis and J. W. Grizzle. The Spring Loaded Inverted Pendulum as the Hybrid Zero Dynamics of an Asymmetric Hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009.

[117] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar. Influence of aerodynamics and proximity effects in quadrotor flight. In *Proceedings of the International Symposium on Experimental Robotics*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 289–302, 2012.

[118] Stephen Prajna. Barrier Certificates for Nonlinear Model Validation. *Automatica*, 42(1):117–126, 2006.

[119] J. Proctor and P. Holmes. Steering by Transient Destabilization in Piecewise-Holonomic Models of Legged Locomotion. *Regular and Chaotic Dynamics*, 13(4):267–282, 2008.

[120] A. O. Pullin, N. J. Kohut, D. Zarrouk, and R. S. Fearing. Dynamic turning of 13 cm robot comparing tail and differential drive. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5086–5093, 2012.

[121] F. Qian, T. Zhang, C. Li, P. Masarati, A. M. Hoover, P. Birkmeyer, A. O. Pullin, R. S. Fearing, and D. I. Goldman. Walking and running on yielding and fluidizing ground. In *Robotics: Science and Systems*, pages 345–352, 2012.

[122] M. H. Raibert. *Legged Robots that Balance.* MIT Press, Cambridge, MA, 1986.

[123] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[124] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3):38–52, Jun. 2000.

[125] L. R. Ray and R. F. Stengel. A Monte Carlo Approach to the Analysis of Control System Robustness. *Automatica*, 29(1):229–236, 1993.

[126] P. Roy, P. Tabuada, and R. Majumdar. Pessoa 2.0: a controller synthesis tool for cyber-physical systems. In Marco Caccamo, Emilio Frazzoli, and Radu Grosu, editors, *Hybrid Systems: Computation and Control*, pages 315–316, 2011.

[127] U. Saranli, M. Bühler, and D. E. Koditschek. RHex: A Simple and Highly Mobile Hexapod Robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

[128] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Publishing, Malabar, FL, 2006.

[129] M. Schmidt and H. Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324:81–85, 2009.

[130] J. Schmitt and P. Holmes. Mechanical models for insect locomotion: dynamics and stability in the horizontal plane - I. Theory. *Biological Cybernetics*, 83(6):501–515, 2000.

[131] J. Schmitt and P. Holmes. Mechanical models for insect locomotion: dynamics and stability in the horizontal plane - II. Application. *Biological Cybernetics*, 83(6):517–527, 2000.

[132] W. J. Schwind. *Spring Loaded Inverted Pendulum Running: A Plant Model*. PhD thesis, University of Michigan, 1998.

[133] N. Seegmiller, F. Rogers-Marcovitz, G. Miller, and A. Kelly. Vehicle model identification by integrated prediction error minimization. *The International Journal of Robotics Research*, 32(8):912–931, 2013.

[134] J. E. Seipel, P. J. Holmes, and R. J. Full. Dynamics and stability of insect locomotion: a hexapedal model for horizontal plane motions. *Biological Cybernetics*, 91(2):76–90, 2004.

[135] S. Shah, C. D. Pahlajani, N. Lacock, and H. G. Tanner. Stochastic receding horizon control for robots with probabilistic state constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2893–2898, 2012.

[136] S. Shah, C. D. Pahlajani, and H. G. Tanner. Optimal navigation for vehicles with stochastic dynamics. *IEEE Transactions on Control Systems Technology*, 23(5):2003–2009, 2015.

[137] S. Shah and H. G. Tanner. Bounding the uncertainity in nonlinear robust model predictive control using sphere covering. In *19th Mediterranean Conference on Control and Automation*, pages 807–812, 2011.

[138] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake. Bounding on rough terrain with the LittleDog robot. *The International Journal of Robotics Research*, 30(2):192–215, 2011.

[139] M. Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 2013.

[140] R. S. Smith and J. C. Doyle. Model Validation: a Connection Between Robust Control and Identification. *Transactions on Automatic Control*, 37(7):942–952, 1992.

[141] R. S. Smith and G. E. Dullerud. Continuous-time Control Model Validation using Finite Experimental Data. *Transactions on Automatic Control*, 41(8):1094–1105, 1996.

[142] O. J. Sordalen and O. Egeland. Exponential stabilization of nonholonomic chained systems. *IEEE Transactions on Automatic Control*, 40(1):35–49, 1995.

[143] A. J. Spence, S. Revzen, J. E. Seipel, C. Mullens, and R. J. Full. Insects Running on Elastic Surfaces. *Journal of Experimental Biology*, 213:1907–1920, 2010.

[144] A. Stager, K. Karydis, and H. G. Tanner. A Passively Sprawling Miniature Legged Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3134–3139, 2015.

[145] J. Steinhardt and R. Tedrake. Finite-time regional verification of stochastic nonlinear systems. *The International Journal of Robotics Research*, 31(7):901–923, 2012.

[146] J. J. Stoker. *Differential Geometry.* Wiley-Interscience, New York, NY, 1969.

[147] M. Sznaier and M. C. Mazzaro. An LMI Approach to Control-oriented Identification and Model (In)validation of LPV Systems. *Transactions on Automatic Control*, 48(9):1619–1624, 2003.

[148] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems: With Applications.* Springer Publishing Company, Inc., 2nd edition, 2012.

[149] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005.

[150] A. Timcenko and P. Allen. Modeling dynamic uncertainty in robot motions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 531–536, 1993.

[151] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[152] V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, NY, 1998.

[153] M. Vidyasagar. Randomized Algorithms for Robust Controller Synthesis using Statistical Learning Theory. *Automatica*, 37(10):1515–1528, 2001.

[154] M. Vidyasagar. *Learning and Generalization With Applications to Neural Networks.* Springer-Verlag, London, UK, 2nd edition, 2003.

[155] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.

[156] R. J. Wood, S. S. Avadhanula, R. R. Sahai, E. E. Steltz, and R. S. Fearing. Microrobot Design Using Fiber Reinforced Composites. *ASME Journal of Mechanical Design*, 130(5):1–11, 2008.

[157] D. Xu, Z. Ren, G. Gu, and J. Chen. LFT Uncertain Model Validation with Time- and Frequency-domain Measurements. *Transactions on Automatic Control*, 44(7):1435–1441, 1999.

[158] A. A. Yumaryanto, J. An, and S. Lee. A cockroach-inspired hexapod robot actuated by LIPCA. In *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–6, 2006.

[159] S. Zarovy, M. Costello, A. Mehta, G. Gremillion, D. Miller, B. Ranganathan, J. S. Humbert, and P. Samuel. Experimental study of gust effects on micro air vehicles. In *AIAA Conference on Atmospheric Flight Mechanics*, pages AIAA–2010–7818, 2010.

[160] D. Zarrouk and R. S. Fearing. Compliance-based dynamic steering for hexapods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3093–3098, 2012.

[161] D. Zarrouk, A. O. Pullin, N. J. Kohut, and R. S. Fearing. STAR - a sprawl tuned autonomous robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 20–25, 2013.

[162] T. Zhang, F. Qian, C. Li, P. Masarati, A. M. Hoover, P. Birkmeyer, A. O. Pullin, R. S. Fearing, and D. I. Goldman. Ground fluidization promotes rapid running of a lightweight robot. *The International Journal of Robotics Research*, 7(32):859–869, 2013.

Appendix A
KEY TERMINOLOGY IN LEGGED LOCOMOTION

For the convenience of the reader, we present here a collection of key terms in legged locomotion that are used in this dissertation.

abduction	Motion of the legs on the coronal plane that moves them farther from the center of the body
adduction	Motion of the legs on the coronal plane that brings them closer to the center of the body
contralateral	On opposite sides
coronal	Vertical plane dividing a body into the front and back halves
cycle	Periodic motion of the legs
duty factor	Percentage of the total cycle during which a particular leg touches the ground
gait	Pattern of movement of the legs
ipsilateral	On the same side
protraction	Motion of the legs on the sagittal plane that moves them further away from the center of the body
retraction	Motion of the legs on the sagittal plane that brings them closer to the center of the body
sagittal	Vertical plane dividing a body into a right and left half
stance phase	Portion of the cycle during which a particular leg touches the ground

swing phase	Portion of the cycle during which a particular leg is lifted off the ground and moves forward

Appendix B
DERIVATION OF CLOSED-FORM EXPRESSIONS

We present here the steps to derive the closed-form expressions (3.2) for the right pair; due to symmetry, the expression for the left pair follows directly (see (3.3)).

The target equations are repeated here:

$$\Delta x = |O_1G'| \sin\Big(\omega' - |\Delta\theta|\Big) - |O_1G| \sin\omega \ \ , \tag{B.1a}$$

$$\Delta y = |O_1G'| \cos\Big(\omega' - |\Delta\theta|\Big) - |O_1G| \cos\omega \ \ , \tag{B.1b}$$

$$\Delta\theta = \phi_1^{\mathrm{lo}} - \phi_1^{\mathrm{td}} + \angle AO_1A' \ \ . \tag{B.1c}$$

where for clarity of presentation we have dropped the dependence on ϕ_1, and $|O_1G'| \triangleq r(\phi_1), |O_1G| \triangleq r(\phi_1^{\mathrm{td}}), \omega' = \omega(\phi_1)$, and $\angle AO_1A' \triangleq \chi(\phi_1)$.

With reference to Figure B.1, the unprimed quantities correspond to locations in the configuration of the mechanism at the beginning of a left pair step (touchdown), and the primed quantities mark the new locations at the end of the step (liftoff). In this context, the vector of displacement of the geometric center of the model in the body frame, GG', as well as the difference in the orientation of the long body axis between beginning and end of step, $\Delta\theta$, can be expressed in terms of $\phi_1 \in [\phi_1^{\mathrm{td}}, \phi_1^{\mathrm{lo}}]$ as follows:

Derivation of (B.1a) **and** (B.1b)

Non-primed symbols correspond to touchdown and primed ones to liftoff. With reference to Figure B.1, the vector-loop equation in triangle $\triangle A'O_1G'$ using complex notation yields

$$\begin{aligned}\boldsymbol{R}_{GG'} &= \boldsymbol{R}_{O_1G'} - \boldsymbol{R}_{O_1G} \\ &= (\boldsymbol{R}_{O_1A'} + \boldsymbol{R}_{A'G'})\,\mathrm{e}^{\mathrm{j}|\Delta\theta|} - (\boldsymbol{R}_{O_1A} + \boldsymbol{R}_{AG}) \\ &= \left[|O_1G'|\,\mathrm{e}^{\mathrm{j}(\frac{\pi}{2}-\omega')}\right]\mathrm{e}^{\mathrm{j}|\Delta\theta|} - |O_1G|\,\mathrm{e}^{\mathrm{j}(\frac{\pi}{2}-\omega)}\ . \end{aligned} \tag{B.2}$$

The displacement along the x and y axes is then found by substituting $\mathrm{e}^{\pm j\theta} = \cos\theta \pm j\sin\theta$ and taking the real and imaginary parts of (B.2), respectively, that is

$$\begin{aligned}\Delta x &= |O_1G'|\Big[\cos(|\Delta\theta|)\sin(\omega') - \sin(|\Delta\theta|)\cos(\omega')\Big] - |O_1G|\sin(\omega)\,, \\ \Delta y &= |O_1G'|\Big[\cos(|\Delta\theta|)\cos(\omega') + \sin(|\Delta\theta|)\sin(\omega')\Big] - |O_1G|\cos(\omega)\,.\end{aligned}$$

To evaluate the above quantities, we need to calculate $|O_1G|$, ω, $|O_1G'|$, ω', as well as $\Delta\theta$. First, let $|O_1A| = |O_1A'| = l$, and $|AG| = |A'G'| = d/2$ (see Figure 3.1(a)). Then, application of the cosine and sine laws to $\triangle AO_1G$ gives the quantities $|O_1G|$ and ω, while $|O_1G'|$ and ω' are found by applying the trigonometric laws to $\triangle A'O_1G'$. We have

$$\begin{aligned}|O_1G| &= \sqrt{l^2 + \frac{d^2}{4} - l\,d\cos\left(\frac{\pi}{2} - \phi_1^{\mathrm{td}}\right)}\ , \\ \omega &= \arcsin\left(\frac{l\sin(\frac{\pi}{2} - \phi_1^{\mathrm{td}})}{|O_1G|}\right)\ , \\ |O_1G'| &= \sqrt{l^2 + \frac{d^2}{4} - l\,d\cos\left(\frac{\pi}{2} - \phi_1^{\mathrm{lo}}\right)}\ , \\ \omega' &= \arcsin\left(\frac{l\sin(\frac{\pi}{2} - \phi_1^{\mathrm{lo}})}{|O_1G'|}\right)\ .\end{aligned}$$

Derivation of (B.1c)

The expression for $\Delta\theta$ is

$$\begin{aligned}\Delta\theta &= \angle A'C'O_1 - \angle ACO_1 \\ &= \left(\frac{\pi}{2} + \phi_1^{\mathrm{lo}} - \angle A'O_1C'\right) - \left(\frac{\pi}{2} + \phi_1^{\mathrm{td}} - \angle AO_1C\right) \\ &= \phi_1^{\mathrm{lo}} - \phi_1^{\mathrm{td}} + (\angle AO_1O_2 - \angle A'O_1O_2) \\ &= \phi_1^{\mathrm{lo}} - \phi_1^{\mathrm{td}} + \angle AO_1A'\ ,\end{aligned}$$

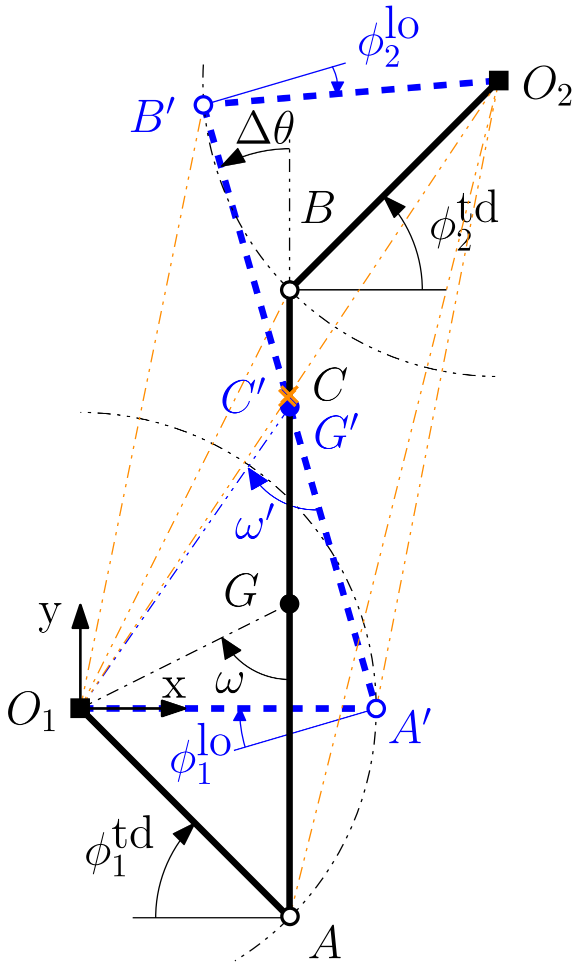


Figure B.1: Geometric analysis of SFM. Bold thick lines and unprimed symbols annotate the right pair configuration at touchdown, while dashed thick lines and primed letters describe the mechanism at the liftoff configuration. Thin dashed-dotted lines outline the various triangles we refer to in text. Due to the no-slip assumption, the mechanism pivots around the touchdown points, O_1 and O_2. Points C and C' denote the intersection of the torso of the model and the segment O_1O_2.

which requires knowledge of $\angle AO_1A'$. To find the latter, we first calculate the length $|O_1O_2|$ and the angle ϕ_2^{lo}.

The vector-loop equation for the mechanism is

$$\boldsymbol{R}_{O_1O_2} = \boldsymbol{R}_{AB} + \boldsymbol{R}_{BO_2} - \boldsymbol{R}_{AO_1} \ ,$$

which can be expressed in complex number notation as

$$|O_1O_2|\,\mathrm{e}^{j(\angle AO_1O_2-\phi_1^{\mathrm{td}})} = d\,\mathrm{e}^{j(\pi/2)} + l\,\mathrm{e}^{j(\phi_2^{\mathrm{td}})} - l\,\mathrm{e}^{j(\pi-\phi_1^{\mathrm{td}})} \ .$$

Then, substituting $\mathrm{e}^{\pm j\theta} = \cos\theta \pm j\sin\theta$ and separating real and imaginary parts, we write the above equation as

$$\begin{aligned}|O_1O_2|\,\cos(\angle AO_1O_2-\phi_1^{\mathrm{td}}) &= l\cos(\phi_2^{\mathrm{td}}) + l\cos(\phi_1^{\mathrm{td}})\\ |O_1O_2|\,\sin(\angle AO_1O_2-\phi_1^{\mathrm{td}}) &= d + l\sin(\phi_2^{\mathrm{td}}) - l\sin(\phi_1^{\mathrm{td}}) \ .\end{aligned}$$

Squaring both equations, adding them, and then solving for $|O_1O_2|$ yields

$$\begin{aligned}&|O_1O_2| =\\ &\sqrt{2l^2+d^2+2l\Big(l\cos(\phi_1^{\mathrm{td}}+\phi_2^{\mathrm{td}})+d(\sin(\phi_2^{\mathrm{td}})-\sin(\phi_1^{\mathrm{td}}))\Big)} \ .\end{aligned}$$

To find the angle ϕ_2^{lo} we turn our attention to $\triangle O_1B'A'$. Applying the laws of sines and cosines produces

$$\begin{aligned}|O_1B'| &= \sqrt{l^2+d^2-2l\,d\cos\left(\frac{\pi}{2}-\phi_1^{\mathrm{lo}}\right)} \ ,\\ \angle O_1B'A' &= \arcsin\left(\frac{l\sin(\frac{\pi}{2}-\phi_1^{\mathrm{lo}})}{|O_1B'|}\right) \ .\end{aligned}$$

From the cosine law applied to $\triangle O_1B'O_2$ we have

$$\angle O_1B'O_2 = \arccos\left(\frac{|O_1B'|^2+l^2-|O_1O_2|^2}{2|O_1B'|l}\right) \ .$$

Then, ϕ_2^{lo} is calculated as

$$\phi_2^{\mathrm{lo}} = \angle O_1B'O_2 - \angle O_1B'A' - \frac{\pi}{2} \ . \tag{B.3}$$

We are now ready to calculate $\angle AO_1A'$ which is needed for finding $\Delta\theta$ in (B.1c). We can write

$$\begin{aligned}\angle AO_1A' &= \operatorname{arctan2}\left(\frac{\sin(\angle AO_1A')}{\cos(\angle AO_1A')}\right)\\ &= \operatorname{arctan2}\left(\frac{\sin(\angle AO_1O_2-\angle A'O_1O_2)}{\cos(\angle AO_1O_2-\angle A'O_1O_2)}\right) \ ,\end{aligned}$$

or equivalently

$$\angle AO_1A' = \operatorname{arctan2}\left(\frac{\sin(\angle AO_1O_2)\cos(\angle A'O_1O_2) - \cos(\angle AO_1O_2)\sin(\angle A'O_1O_2))}{\cos(\angle AO_1O_2)\cos(\angle A'O_1O_2) + \sin(\angle AO_1O_2)\sin(\angle A'O_1O_2))}\right) \ .$$

Application of the trigonometric laws to $\triangle AO_1O_2$ and $\triangle A'O_1O_2$ gives

$$\begin{aligned}
\cos(\angle AO_1O_2) &= \frac{|O_1O_2|^2 + l^2 - |AO_2|^2}{2|O_1O_2|l} \ , \\
\sin(\angle AO_1O_2) &= \frac{|AO_2|}{|O_1O_2|}\sin(\angle O_1AO_2) \ , \\
\cos(\angle A'O_1O_2) &= \frac{|O_1O_2|^2 + l^2 - |A'O_2|^2}{2|O_1O_2|l} \ , \\
\sin(\angle A'O_1O_2) &= \frac{|A'O_2|}{|O_1O_2|}\sin(\angle O_1A'O_2) \ .
\end{aligned}$$

The last quantities remaining to be found are $|AO_2|, \angle O_1AO_2, |A'O_2|$, and $\angle O_1A'O_2$. We first write

$$\begin{aligned}
\angle O_1AO_2 &= \frac{\pi}{2} - \phi_1^{\mathrm{td}} + \angle BAO_2 \ , \\
\angle O_1A'O_2 &= \frac{\pi}{2} - \phi_1^{\mathrm{lo}} + \angle B'A'O_2 \ .
\end{aligned}$$

Then, application of the trigonometric laws to $\triangle ABO_2$ and $\triangle A'B'O_2$ finally gives

$$\begin{aligned}
|AO_2| &= \sqrt{l^2 + d^2 - 2l\,d\cos\left(\frac{\pi}{2} + \phi_2^{\mathrm{td}}\right)} \ , \\
\angle BAO_2 &= \arcsin\left(\frac{l\sin(\frac{\pi}{2} + \phi_2^{\mathrm{td}})}{|AO_2|}\right) \ , \\
|A'O_2| &= \sqrt{l^2 + d^2 - 2l\,d\cos\left(\frac{\pi}{2} + \phi_2^{\mathrm{lo}}\right)} \ , \\
\angle B'A'O_2 &= \arcsin\left(\frac{l\sin(\frac{\pi}{2} + \phi_2^{\mathrm{lo}})}{|A'O_2|}\right) \ ,
\end{aligned}$$

where ϕ_2^{lo} is found by (B.3), while ϕ_1^{td} and ϕ_2^{td} are known.

In sum, (B.1a), (B.1b), and (B.1c) determine the local state progression as a function of the input angle. For the case of the left pair, (B.1a) and (B.1c) are negated, and (B.1b) is the same.

Appendix C

UNCERTAINTY QUANTIFICATION IN SMALL-SCALE AERIAL VEHICLES

This appendix provides an example of applying the reported method for probabilistic model validation and stochastic extension to a system represented by a set of differential equations. The example considered here is the steady-state response of a small-scale quadrotor during hover. The performance of small-scale rotorcraft vehicles while operating in close proximity to rigid surfaces (e.g., ground, ceiling, etc.) can be considerably affected by uncertain aerodynamic effects that are difficult to incorporate in low-dimensional models such as those typically used for control [117].

For clarity, here only a very small fraction of the dynamics of the physical system is excited, and thus the associated analysis allows no direct generalizations to other flying regimes; yet the case still provides an adequate example of how our method can be applied to nominal deterministic models of dynamical systems that come in the form of differential equations. Further, this example shows that the method can also work if the stochasticity is injected through an exogenous stochastic disturbance input, rather than through the model's parameters. Merely calculating a sample variance and using it directly to estimate the stochastic disturbance does not provide probabilistic guarantees of fidelity.

C.1 The Nominal Model

In applying the method to the altitude control of small scale quadrotor, we isolate the vertical dynamics component and characterize the observed data variability at steady state. Constraining the motion along the vertical direction is achieved by the support structure of Figure C.1.

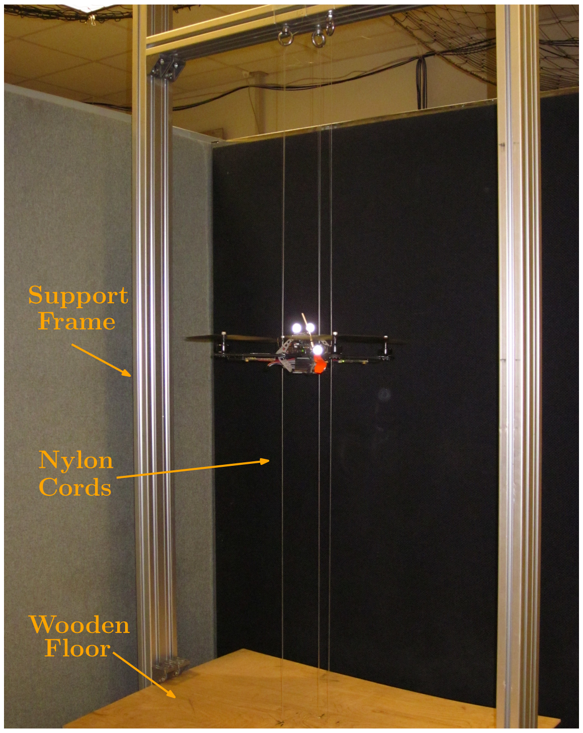


Figure C.1: The experimental setup. Our support structure aids in restraining the motion along the vertical direction only. The driving strings are composed of nylon cords; they provide strong support and minimal friction, while minimizing the fluctuations on the normal to the motion plane. We also added a wooden floor to artificially generate the ground effect.

A simplified quadrotor model for motion on the vertical plane and ignoring the actuator dynamics, can be expressed as [99]

$$\begin{aligned} m\,\ddot{y} &= -f\sin\phi \\ m\,\ddot{z} &= f\cos\phi - m\,g \\ I\,\ddot{\phi} &= \tau \end{aligned} \tag{C.1}$$

where f and τ are the thrust and pitch moments, respectively, m and I are the mass and moment of inertia of the vehicle, and g is the acceleration of gravity. The support structure forces $\phi = 0$, and with the addition of a gravity compensation term

$$f = m\,g + m\,u \ ,$$

the model (C.1) reduces to a double integrator

$$\ddot{z} = u \ . \tag{C.2}$$

The input u is then determined by a proportional-integral-derivative (PID) controller

$$u = K_P\, e + K_I \int e\, \mathrm{dt} + K_D\, \dot{e} \ , \tag{C.3}$$

where $e = (r - z)$ is the position error, and r is the desired hovering height. For the purposes of this work, we consider four distinct altitudes shown in Table C.1. The PID gains (K_P, K_I, K_D) are selected empirically and their values are given in Table C.2. In order to avoid unrealistic control efforts, the PID output is saturated within the region $-2 \leq u \leq 7$; the same saturation interval is used in our experiments as well.

Table C.1: Quadrotor hovering altitudes

Case	I (Low)	II (Mid-low)	III (Mid-high)	IV (High)
r [m]	0.02	0.11	0.20	0.50

Table C.2: PID gains

K_P	K_I	K_D
4.00	4.50	5.00

Figure C.2(a) depicts the model-predicted closed-loop trajectories plotted against collected experimental data. It can be verified that the deterministic closed-loop control model is able to predict quite accurately *on average* the experimentally-observed steady-state response of the system in all four cases. In the following sections we will employ the Algorithm of Table 4.2 to show how the deterministic closed-loop system can be extended to a stochastic setting to capture the data variability in steady state.

C.2 Stochastic Extension

Contrary to the case study of Section 4.3, here we introduce stochasticity through an exogenous random excitation term that acts as stochastic perturbations on the nominal closed-loop dynamics (C.2). This shows that the method is also applicable when a system is perturbed by noise, and can be used to identify such noise terms in practice, while providing probabilistic guarantees on the reported outcome.

The closed-loop stochastic model of the system is now

$$\ddot{z} + \tilde{\xi} = u \ ,$$

where u is calculated by the same PID controller as in (C.3) and then saturated in the region $-2 \leq u \leq 7$, with $\tilde{\xi} \sim \mathcal{N}(0, \sigma_{\xi}^2)$, and the variance σ_{ξ}^2 to be determined by the Algorithm of Table 4.2. In essence, $\tilde{\xi}$ here corresponds to a zero-mean Gaussian process corrupting the control effort, and the task of the method is to find the variance σ_{ξ}^2 that captures the data variability in steady state under prespecified fidelity specifications. Figure C.3 summarizes schematically the closed-loop control stochastic model we consider here.

C.3 Application of the Method

In this section we apply the proposed methodology to determine σ_{ξ}^2 at *steady state* for each of the four cases shown in Table C.1. We select the same problem parameters as in the previous case study: $p = 0.90, \gamma = 0.99, \alpha = 0.29, \rho = 0.35, \varepsilon = 0.29$, and $\delta = 0.16$, giving $N = 8$, $M = 31$ with $K = 8$, and $k_2 = 4.147$ to construct the cone of data for each multisample. We collect closed-loop altitude measurement data from a total of 248 experimental trajectories, for each height. Trials for the first three cases last 15 sec; for the fourth, 20 sec, because the system enters its steady state at the end of the 15th second. The feedback loop refreshes at 30 Hz, at the same frequency as our motion capture system. The initial pose errors are not significant since the support structure ensures that the start position remains unchanged in all trials.

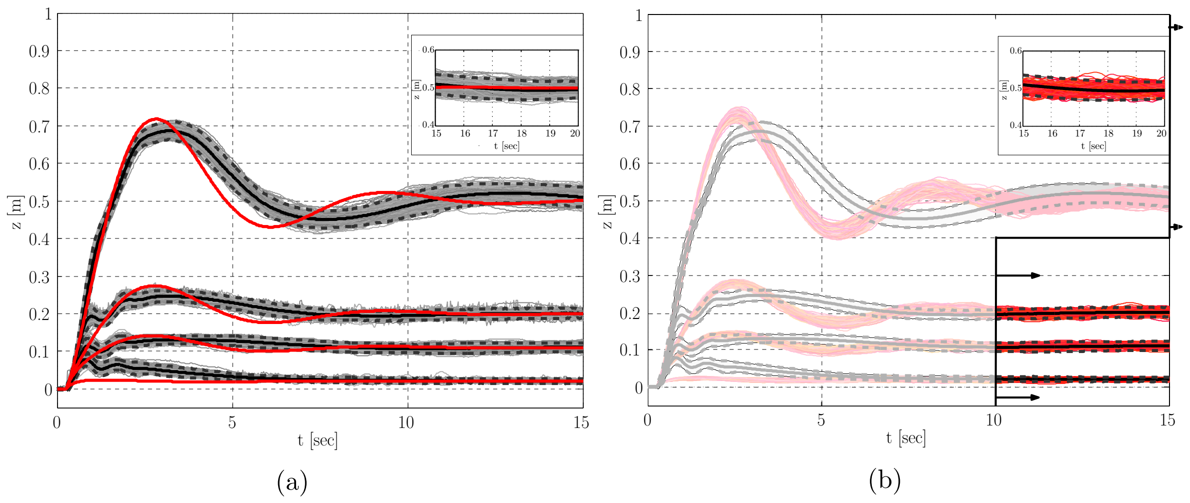


Figure C.2: (a) Experimentally collected paths for a quadrotor aerial vehicle tasked to hover at four distinct altitudes (0.02 m, 0.11 m, 0.20 m, and 0.50 m, respectively). Individual paths are enclosed within the respective cone-of-data areas (marked with the dashed curves), while the experimental averages are shown in solid thick curves. For the fourth case, the steady state is achieved at $t = 20$ sec. For clarity purposes, we show the last 5 sec in the add-on window on the top right corner of the figure. The superimposed solid thick curves correspond to the model-predicted outputs according to (C.3). (b) The output of the stochastically perturbed control architecture depicted in Figure C.3, where the values of σ_ξ for each case have been estimated by the proposed framework, and are shown in Table C.3. A set of 248 random model instantiations are plotted on top of the experimental averages and cones of data. We are now able to capture the data variability during the *steady-state* response.

Figure C.2(a) presents the experimentally observed trajectories. Let $\mathcal{W}_{\mathrm{l}}$, $\mathcal{W}_{\mathrm{ml}}$, $\mathcal{W}_{\mathrm{mh}}$, $\mathcal{W}_{\mathrm{h}}$ indicate the collections of the trajectories for the low, mid-low, mid-high, and high hovering altitude, respectively. Let w denote an element in these sets. The average for each case is denoted by $w_{\mathrm{l}}^{\mathrm{ave}}$, $w_{\mathrm{ml}}^{\mathrm{ave}}$, $w_{\mathrm{mh}}^{\mathrm{ave}}$, $w_{\mathrm{h}}^{\mathrm{ave}}$, respectively, shown in solid thick curves. Dashed curves mark the cone of data for each case. The additional 5 sec for the highest altitude case are shown as an add-on figure on top of Figure C.2(a).

The results of the application of the proposed method are summarized in Table C.3. Figure C.2(b) depicts 248 randomly generated model paths for each case, parameterized according to the values in Tables C.2 and C.3. Focusing on the steady-state response of the system only, we can verify that the stochastically extended control scheme of Figure C.3 is capable of capturing the data variability, and that the resulting paths match closely their experimental counterparts. We highlight in Figure C.2(b) the steady-state part of system responses to emphasize that our focus is in this particular regime. The induced ground effect is evident in the behavior of the quadrotor closest to the ground, where it appears as if the generated airflow creates an aerodynamic "cushion" below the platform. This has a stabilizing effect on the platform, as indicated by the reduction in the amplitude of the vehicle's residual oscillations at steady state.

Table C.3: Probably approximate near maximum quadrotor model uncertainty

Case	I (Low)	II (Mid-low)	III (Mid-high)	IV (High)
$\bar{\sigma}_\xi$	0.08	0.13	0.17	0.30

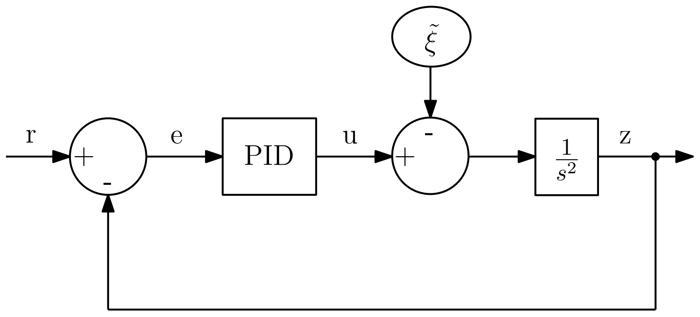


Figure C.3: Schematic representation of the closed-loop control stochastic model for the steady-state vertical dynamics component during quadrotor hovering. The input r denotes the desired hovering height, while the gains of the PID controller have been tuned a-priori, and remain the same with those used in our data collections. In $\tilde{\xi} \sim \mathcal{N}(0, \sigma_\xi^2)$, the variance σ_ξ^2 is estimated using the proposed algorithm.

Note that the control architecture of Fig C.3 does not capture the ripples during the transient phases. These ripples could be attributed, at least in part, to the nonlinear coupling between the pitch angle ϕ and the actuator input f (cf. (C.1)) which are currently ignored by assuming that $\phi \equiv 0$. Thus the simplified deterministic model is incapable of reproducing this transient, irrespectively of how stochasticity is infused. Using a full nonlinear model such as (C.1) may enable one to capture this transient behavior, however doing so falls outside the scope of the present paper.

C.4 Discussion

This section discusses an alternative way to perturb the nominal closed-loop dynamics (C.2), and provides some insight on the predictive ability of the augmented stochastic architecture shown in Figure C.3.

We first perturb the output of the model with an exogenous random term—modeled again as a zero-mean Gaussian process process—shown in Figure C.4.[1] Then, we apply the proposed method with all parameters retaining the same values as in Section C.3. The results are summarized in Table C.4, and Figure C.5(a) depicts 248 randomly generated model paths for each case, parameterized according to the values found in Tables C.2, and C.4. The modified stochastic extension also captures the data variability once the steady state has been reached, but system responses are somewhat different (qualitatively) from the experimental data due to "chattering" (see Figure C.5(a)). Overall, however, the methodology does not promote any one option over another; instead, it reports that both are possible solutions. The designer has then to select which solution to keep, based on their own beliefs or assumptions on the possible sources of uncertainty.

Next, we examine how the stochastic architecture of Figure C.3, with variance parameter σ_ξ^2 specified by data corresponding to a single desired hovering altitude, can be used to predict the steady-state response of the vehicle both temporally, and across

[1] Infusing stochasticity in this way may also have practical significance in the case of uncertain or noisy state measurements.

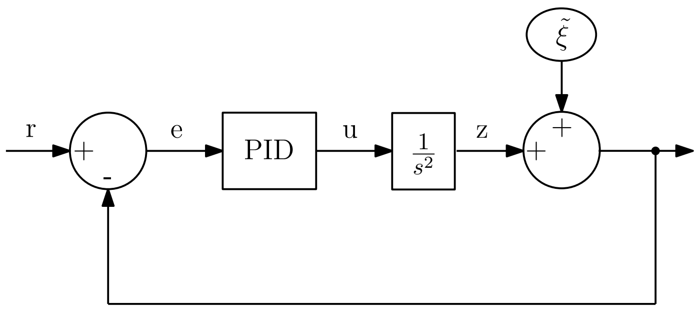


Figure C.4: Schematic representation of the modified closed-loop control architecture. As before, the input r denotes the desired hovering height, and the PID gains have been tuned a-priori.

Table C.4: Probably approximate near maximum quadrotor model uncertainty when the excitation term affects the model output

Case	I (Low)	II (Mid-low)	III (Mid-high)	IV (High)
$\bar{\sigma}_\xi$	0.002	0.004	0.005	0.007

different operating points. In detail, we use as training data those corresponding to case II (i.e., $z_{\text{train}} = 0.11$ m), with $\bar{\sigma}_{\xi,\text{train}} = 0.13$ (from Table C.3). Then, we use the same model, with the same statistics for randomized exogenous forces, but with different inputs (desired hovering altitudes) to predict the steady-state responses in the remaining cases of Table C.1.

Figure C.5(b) presents the results. As expected, the data variability observed in steady state is captured faithfully when the desired reference point does not change too much (i.e., $z_{\text{pred}} = 0.2$ m). However, when the desired reference point induces different operating conditions in which the nature of the environmental interaction with the physical system changes, the resulting stochastic model may offer a rough prediction, albeit collecting more experimental data for these set points is recommended. For example, when the set point is $z_{\text{pred}} = 0.02$ m, we get a more conservative picture of what is happening since the ground effect becomes evident, manifesting itself as a form of a stabilizing aerodynamic cushion under the vehicle (compare with Figure C.2).

The same setup is also used to make temporal predictions that extend for additional 15 s for cases I – III, and 10 s for case IV. As before, provided that the induced operating conditions do not vary significantly, the stochastically extended architecture is able to make accurate temporal predictions, with the caveat that these predictions come with no probabilistic guarantees.

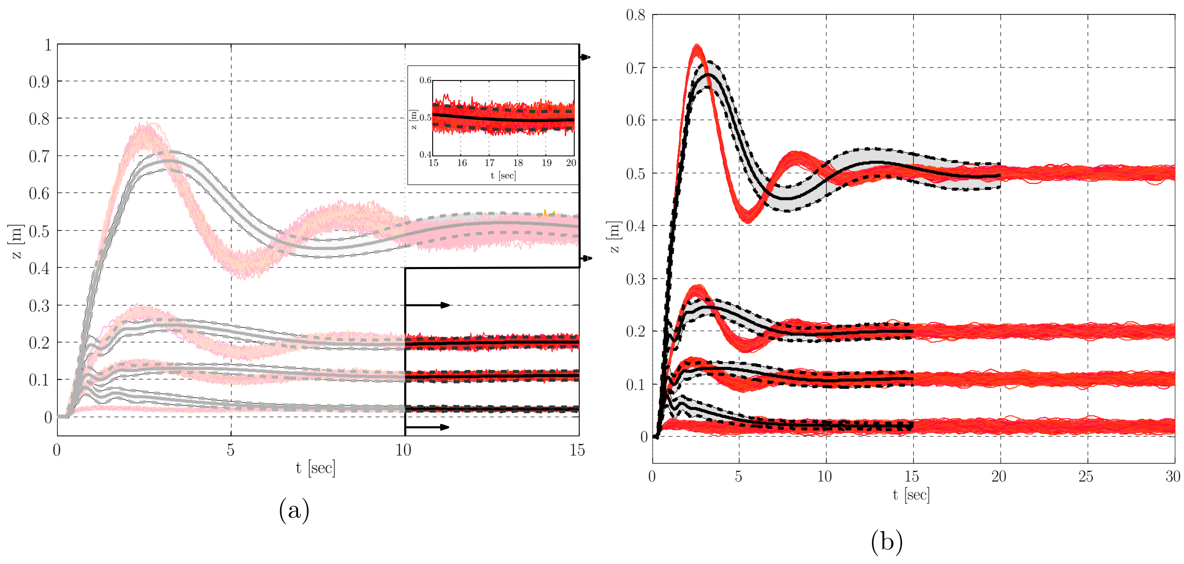


Figure C.5: (a) The output of the stochastically perturbed control architecture depicted in Figure C.3, where the values of σ_ξ for each case have been estimated by the proposed framework, and are shown in Table C.3. A set of 248 random model instantiations are plotted against the experimental averages and cones of data. (b) Training data and exogenous excitation statistics correspond to the second case only ($z_{train} = 0.11$ m, and $\bar{\sigma}_{\xi,train} = 0.13$). The stochastically extended control architecture (Figure C.3) is then used to predict the steady-state response of the system under different inputs, as well as extend these predictions temporally. Provided that the operating conditions do not vary significantly, the stochastic extension is able to make accurate predictions, both temporal, and for different reference points.

Appendix D
REPRINT PERMISSIONS

This dissertation includes results published by the author in conference proceedings [73, 71, 72, 74, 69] and journals [68, 70]. Reprint permissions to reuse material from these works in this present dissertation have been granted by the respective publishers, and are attached in the following pages.

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Model predictive navigation for position and orientation control of nonholonomic vehicles
Conference Proceedings:	Robotics and Automation (ICRA), 2012 IEEE International Conference on
Author:	Karydis, K.; Valbuena, L.; Tanner, H.G.
Publisher:	IEEE
Date:	14-18 May 2012

Copyright © 2012, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE Requesting permission to reuse content from an IEEE publication

Title:	A switching kinematic model for an octapedal robot
Conference Proceedings:	Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on
Author:	Karydis, K.; Poulakakis, I.; Tanner, H.G.
Publisher:	IEEE
Date:	7-12 Oct. 2012

Copyright © 2012, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Probabilistic validation of a stochastic kinematic model for an eight-legged robot
Conference Proceedings:	Robotics and Automation (ICRA), 2013 IEEE International Conference on
Author:	Karydis, K.; Poulakakis, I.; Tanner, H.G.
Publisher:	IEEE
Date:	6-10 May 2013

Copyright © 2013, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE
Requesting permission to reuse content from an IEEE publication

Title:	Planning with the STAR(s)
Conference Proceedings:	Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on
Author:	Karydis, K.; Zarrouk, D.; Poulakakis, I.; Fearing, R.S.; Tanner, H.G.
Publisher:	IEEE
Date:	14-18 Sept. 2014

Copyright © 2014, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Navigation of miniature legged robots using a new template
Conference Proceedings:	Control and Automation (MED), 2015 23th Mediterranean Conference on
Author:	Karydis, K.; Yan Liu; Poulakakis, I.; Tanner, H.G.
Publisher:	IEEE
Date:	16-19 June 2015

Copyright © 2015, IEEE

LOGIN

If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already **a RightsLink user** or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

SPRINGER LICENSE
TERMS AND CONDITIONS

Nov 22, 2015

This is a License Agreement between Konstantinos Karydis ("You") and Springer ("Springer") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3754410309190
License date	Nov 22, 2015
Licensed content publisher	Springer
Licensed content publication	Autonomous Robots
Licensed content title	A template candidate for miniature legged robots in quasi-static motion
Licensed content author	Konstantinos Karydis
Licensed content date	Jan 1, 2014
Volume number	38
Issue number	2
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Order reference number	None
Title of your thesis / dissertation	A Data-Driven Hierarchical Framework for Planning, Navigation, and Control of Uncertain Systems: Applications to Miniature Legged Robots
Expected completion date	Dec 2015
Estimated size(pages)	166
Total	0.00 USD

Terms and Conditions

Introduction
The publisher for this copyrighted material is Springer Science + Business Media. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).
Limited License
With reference to your request to reprint in your thesis material on which Springer Science

Home | Account Info | Help | Live Chat

Title: Probabilistically valid stochastic extensions of deterministic models for systems with uncertainty:

Author: Konstantinos Karydis, Ioannis Poulakakis, Jianxin Sun, Herbert G. Tanner

Publication: International Journal of Robotics Research

Publisher: SAGE Publications

Date: 05/28/2015

Copyright © 2015, © SAGE Publications

Logged in as:
Konstantinos Karydis

LOGOUT

Gratis Reuse

- Without further permission, as the Author of the journal article you may:
  - post the accepted version (version 2) on your personal website, department's website or your institution's repository. You may NOT post the published version (version 3) on a website or in a repository without permission from SAGE.
  - post the accepted version (version 2) of the article in any repository other than those listed above 12 months after official publication of the article.
  - use the published version (version 3) for your own teaching needs or to supply on an individual basis to research colleagues, provided that such supply is not for commercial purposes.
  - use the accepted or published version (version 2 or 3) in a book written or edited by you. To republish the article in a book NOT written or edited by you, permissions must be cleared on the previous page under the option 'Republish in a Book/Journal' by the publisher, editor or author who is compiling the new work.
- When posting or re-using the article electronically, please link to the original article and cite the DOI.
- All other re-use of the published article should be referred to SAGE. Contact information can be found on the bottom of our 'Journal Permissions' page.

BACK | CLOSE WINDOW